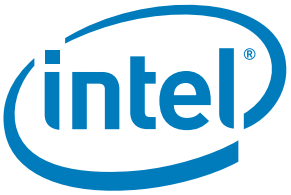
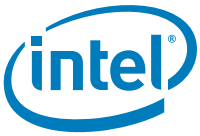
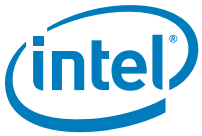


Intel® Ethernet Controller XL710 Datasheet

Networking Division (ND)

Revision: 2.0
July 2014

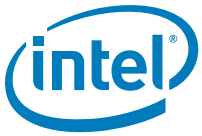




Revision History

Revision	Date	Notes
2.0	July 2014	Initial Release (Intel Public).

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



1.0 Introduction

This document describes the external architecture (including device operation, pin descriptions, register definitions, etc.) for the XL710, a dual-port 40 Gigabit Ethernet (GbE) or quad-port 10 GbE Network Interface Controller. It reflects the silicon device capability while the *Intel® Ethernet Controller XL710 Feature Support Matrix* reflects the features and interfaces actually supported in the NVM and software.

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 XL710.

The XL710 combines standard Ethernet stateless Network Interface Card (NIC) and Fibre Channel over Ethernet (FCoE) block storage acceleration functionality into a single silicon device. It is designed to address the target markets listed in Table 1-1.

Table 1-1. XL710 target markets

Description
Enterprise networking — The XL710 strengths in this market are networking performance, energy efficiency, support for OEM-specific requirements, broad Operating System (OS) and Virtual Machine Monitor (VMM) support, automation (including resource provisioning and monitoring, and workload balancing), converged networking, and emerging standards. For example: Data Center Bridging (DCB) and Virtual Bridging (VEB).
Cloud networking — In the emerging cloud networking market, where computing infrastructure and software are sold as services, and where the large data centers of Internet portal companies such as Google*, Microsoft* and Amazon* drive unique requirements, the XL710 has these strengths: networking performance, energy efficiency, automation (including resource provisioning and monitoring, and workload balancing), sophisticated packet header parsing, and quality open source drivers. In the case where computing infrastructure is sold as a service, the XL710 features important in Enterprise networking and HPC can also be important to the cloud.
Embedded — The embedded market includes security appliances, networking and telecom products, storage targets, etc. The XL710 strengths in this market are performance, broad OS support, quality open source drivers, and Intel product support.

As shown in Figure 1-1, the XL710 is targeted for use in rack mounted or pedestal servers, where it can be deployed as an add-in NIC or LAN on Motherboard (LOM). Some types of Ethernet cables, such as SFP+ direct attach, can be driven directly by the XL710, while other types, such as 10GBASE-T, require the external Physical Layer (PHY) component(s) shown. The XL710 can connect up to four Ethernet ports or it can be configured to connect two 40 Gb/s Ethernet ports. The XL710 is also targeted for use in blade servers, where it can be deployed as a mezzanine card or LOM. The XL710 supports direct connection to backplanes that support the following signalling standards: 40GBASE-KR4 (up to two ports), 1000BASE-KX (up to four ports supported).

Blade backplanes typically connect Ethernet controllers in a dual-redundant star to two separate Ethernet switches. In this configuration, the XL710 can be connected with up to 2 x 10G ports per switch or 1 x 40G port per switch.

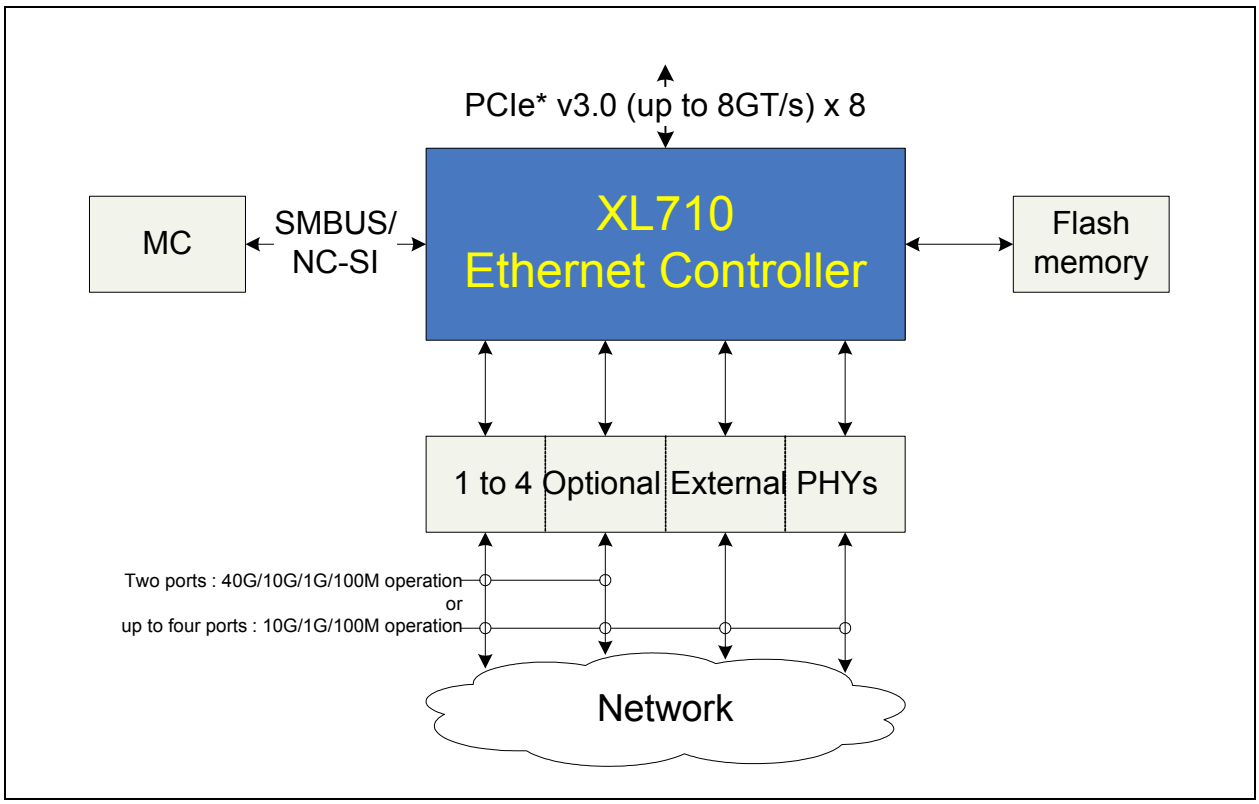


Figure 1-1. Typical rack / pedestal system configuration

As shown in Figure 1-3 the XL710 is also targeted for use in blade servers, where it can be deployed as a mezzanine card or LOM. The XL710 supports direct connection to backplanes that support the following signalling standards: 40GBASE-KR4,10GBASE-KR (up to four ports supported), 10GBASE-KX4 (up to two ports supported), 1000BASE-KX (up to four ports supported).

Blade backplanes typically connect Ethernet controllers in a dual-redundant star to two separate Ethernet switches as shown in Figure 1-3. In this configuration, the XL710 can be connected with two ports to each Ethernet switch or can be connected with only one port to each Ethernet switch, leaving two ports unused.

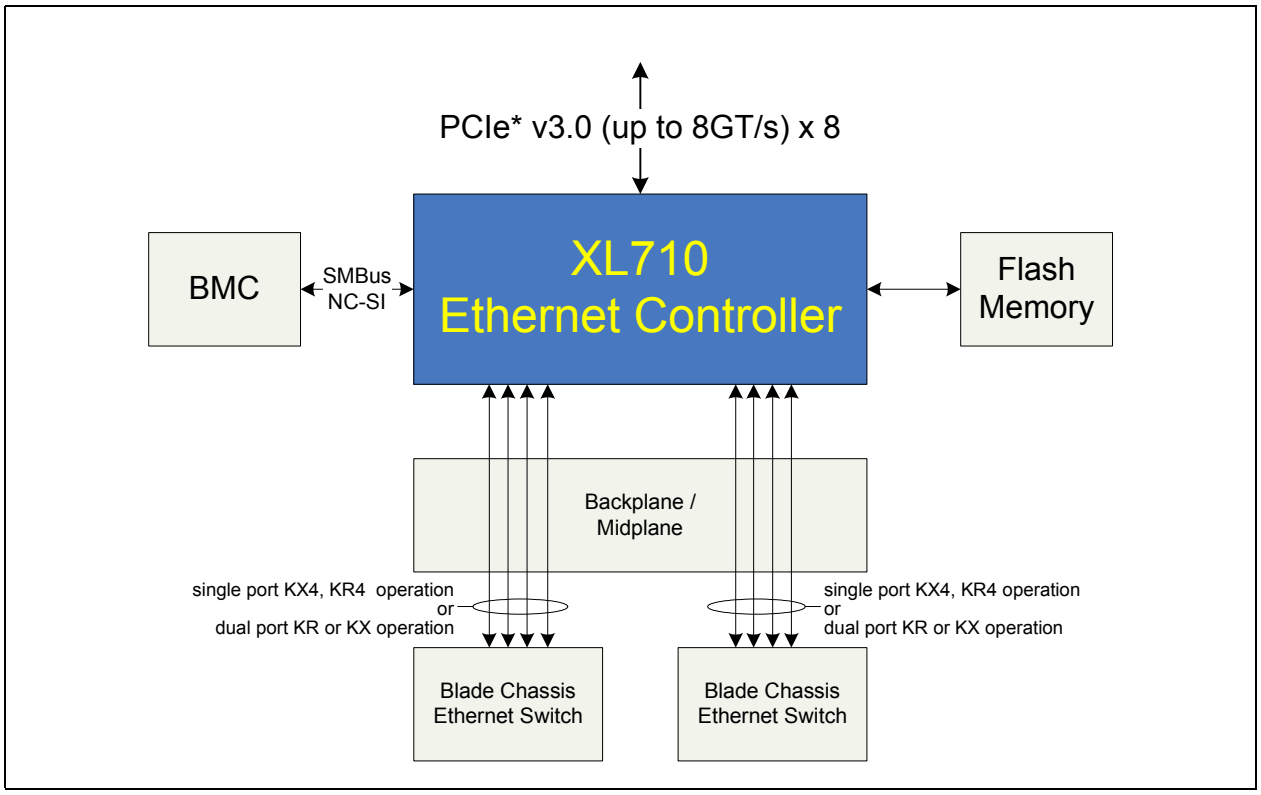


Figure 1-2. Typical blade system dual-redundant star configuration

The total throughput supported by the XL710 is 40 Gb/s, even when connected via two 40 Gb/s connections.

1.1 Block diagram

Figure 1-4 shows a diagram of the XL710's block architecture. This section also provides an overview of the XL710 external interfaces and top-level internal blocks.

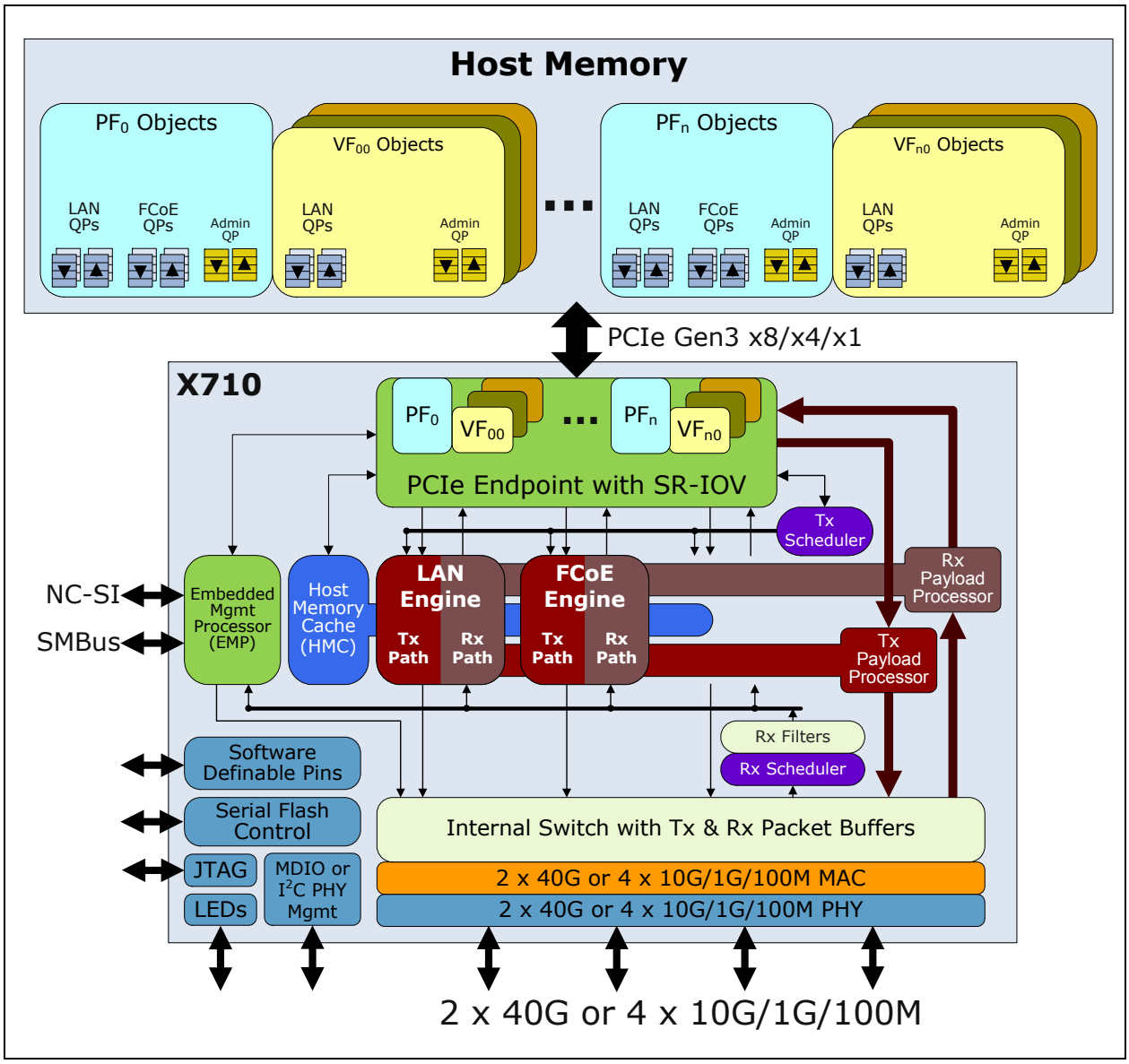


Figure 1-5. XL710 block diagram

1.1.1 PCIe* with Single Root I/O Virtualization (SR-IOV)

The XL710 implements a PCIe v3.0 x8 host interface, which operates at up to 8GT/s or 64 Gb/s. See Section 2.2.1 for a full pin description and Section 14.6.6 for interface timing characteristics.

The XL710's PCIe host interface implements up to 16 Physical Functions (PFs), and up to 128 Virtual Functions (VFs). More details on the XL710's PCIe features are provided in Section 1.2. Section 12.0 describes the PCIe programming interface.

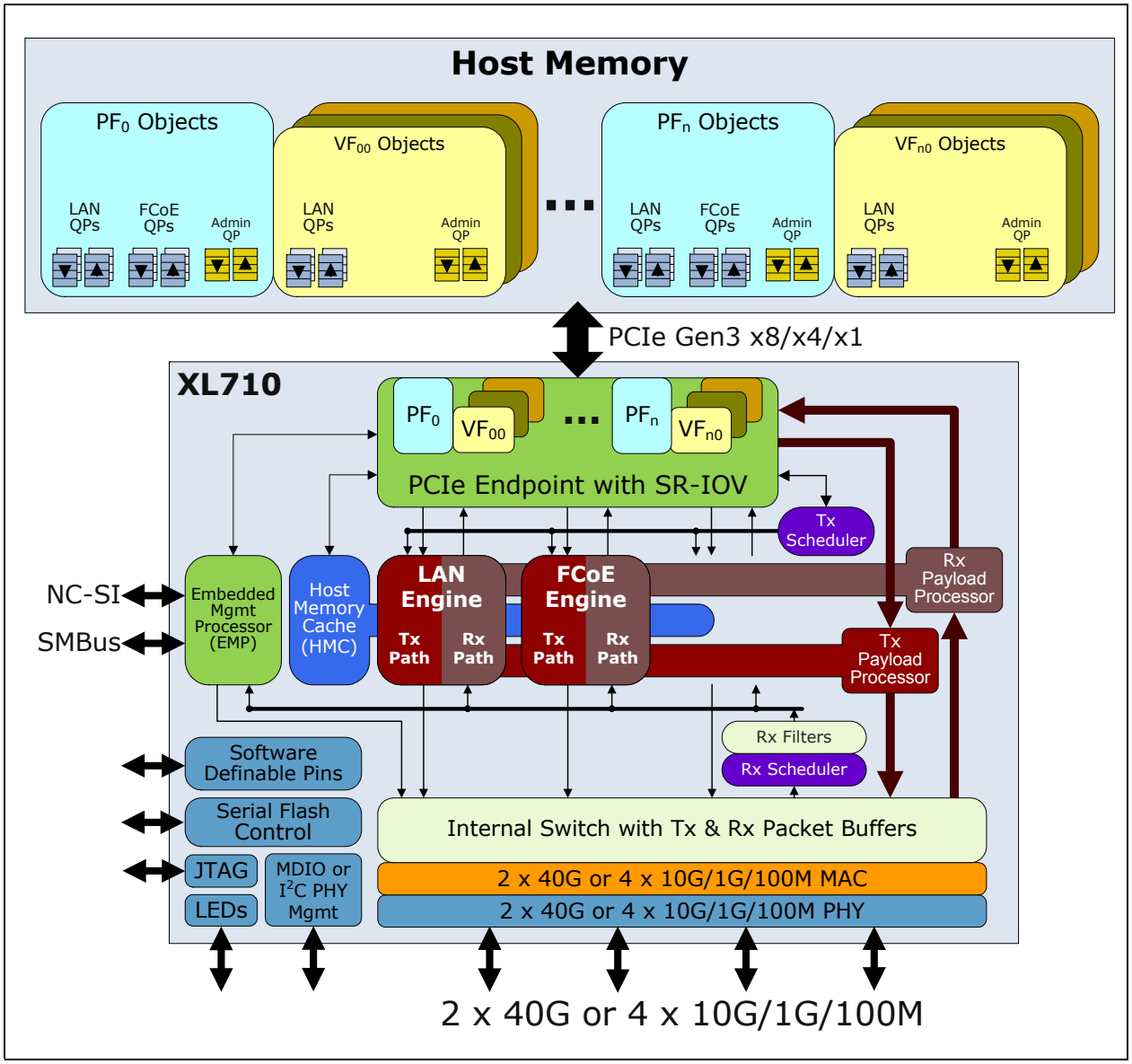


Figure 1-6. XL710 block diagram

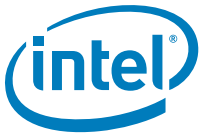
1.1.2 Host memory objects

The XL710 operating system drivers set up a wide variety of host memory objects that are comprehended and manipulated by the XL710. All objects are set up in the context of a PCI function. This is important for at least two reasons:

- Platform security. For example, many types of the XL710 host memory objects are privileged, and are only allowed to be set up in the context of a PF. For example, they are dis-allowed in the context of a Virtual Function, which operates at a lower privilege level than a PF.
- Reliability. If the operating system resets a PCI function, that function's host memory objects are lost, but the host memory objects of other PCI functions survive.

Only 3 types of host memory objects are available. Note that there are many more defined in this Datasheet. They are:

- LAN Queue Pairs (LQPs). These are ring buffers (one transmit, one receive) for submitting commands to the Local Area Network (LAN) engine or FCoE engine. Commands take the form of packets/data to be transmitted, descriptors for empty host memory buffers to be filled with received packets/data, etc. LQPs are typically mapped into OS kernel space. In a virtualized server, they can be assigned either to the VMM, or to VMs using SR-IOV. The XL710 supports up to 1536 LQPs that can be assigned to PFs or VFs as needed. The LQPs assigned to a particular PCI function can be used in these important ways:
  - To segregate FCoE traffic from other LAN traffic.
  - 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 XL710 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.
  - For assigning Traffic Class (TC). Transmit queues assigned to different TCs are serviced at different rates by the XL710 transmit scheduler. Receive queues assigned to different TCs can be serviced at different rates by a Quality of Service (QOS0-enabled operating system and its software device drivers.
- FCoE DDP Queues (DDP). These are a linear list of receive buffers. These DDP buffers are aimed for direct data placement of FCoE exchange payload to the application buffers. There are up to 4 KB DDP queues per PF.
- Admin Queue Pairs. Each PCI function maps an Admin Queue Pair (AQP): one Admin Send Queue (ASQ) and one Admin Receive Queue (ARQ). The ASQ is a ring buffer used by the host driver for submitting commands to configure the XL710. Commands submitted on the ASQ are serviced by the XL710's Embedded Management Processor (EMP). Some examples are commands to reconfigure: the Tx scheduler, an Ethernet link, power management states like EEE, various internal switch and DCB settings, etc. The ARQ conveys events from the EMP to host driver that are not an immediate result of an ASQ command. The host driver posts empty buffers to the ARQ and the EMP fills them with events. Note that commands submitted on a VF ASQ are typically not directly serviced by the EMP, but rather are redirected to the associated PF ARQ. This enables the PF driver to inspect and authorize all VF ASQ commands which are often of a privileged nature. For more details on AQPs.



1.1.3 Ethernet Media Access Controller (MAC) and PHY

The XL710 integrates four IEEE Std 802.3 compliant Ethernet MACs. MAC 0 and 1 operate at 100 Mb/s, 1GbE, 10 GbE, 40 GbE, while MACs 2 and 3 operate at 100 Mb/s, 1GbE, and 10 GbE. All XL710 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 for details.

The XL710 supports up to four active Ethernet ports. It can be configured to support different levels of Ethernet PHY integration using various IEEE standard interfaces that follow.

To connect a XL710 port to the Ethernet media using an external PHY / optical module, the XL710 supports these options:

- Up to two XLAUI interface for connection to an external 40 Gb/s PHY
- Up to two XLPPI interface for connection to an external 40 Gb/s optical module
- Up to four KR interfaces for direct connection to external 10 Gb/s PHYs
- Up to two XAUI interfaces for connection to external 10 Gb/s PHYs
- Up to four SFI interfaces for connection to external SFP+ optical modules or direct attach twin-ax copper cables
- Up to four SGMII interfaces for connection to external Gb/s PHYs

The XL710 supports integrated PHYs for some configurations such as backplane and direct attached copper. To direct-connect a XL710 port to the Ethernet media via Medium Dependent Interface (MDI) with no external PHY, the XL710 supports these options:

- Up to two CR4 interface for connection to direct attach twin-ax copper cable
- Up to two KR4 interface for direct connection to a 40 Gb/s backplane
- Up to two KX4 interfaces for direct connection to a 10 Gb/s backplane
- Up to four KR interfaces for direct connection to a 10 Gb/s backplane
- Up to four KX interfaces for direct connection to a Gb/s backplane

More details on these options, including allowed combinations, can be found in Section 3.2.2.

The XL710 also implements either four independent Management Data Input/Output (MDIO) interfaces or four independent Inter-integrated Circuit (I^2C) interfaces for connection to external PHYs. These enable host software or the XL710 firmware to control connected external PHYs, including the ability to read and write PHY registers. Details on how they are typically connected and used are described in Section 3.2.2.7 through Section 3.2.2.9.

1.1.4 Transmit scheduler

The XL710 provides management interfaces that allow each LAN/FCoE transmit queue (as shown in Figure 1-6) to be placed into a *queue set*. A *queue set* is a list of transmit queues that belong to the same TC and are treated equally by the XL710 transmit scheduler. The XL710 supports a maximum of 384 Virtual Station Interfaces (VSIs), and an average of two queue sets per VSI: two for use by LAN/FCoE. There is no limit to the number of transmit queues that can belong to a queue set.

Typically, one or more queue sets are assigned to a PCI function/VSI, according to the number of TCs it uses. For LAN or FCoE, the queue sets are often only one or a small handful of transmit queues, corresponding to the number of host CPUs assigned to generate traffic on that TC.

The XL710 transmit scheduler supports independent programming of a wide variety of controls that affect queue set behavior. Each queue set can be programmed with an independent static rate limit. Each group of queue sets assigned to a PCI function/VSI can be programmed with DCB Enhanced Transmission Selection (ETS) settings, group static rate limit, and uplink bandwidth share.

The XL710 transmit scheduler also implements controls similar to those previously described for the various internal resources that comprise the XL710's hierarchy of internal switching components. These internal resources include Virtual Ethernet Bridge (VEB), Virtual Ethernet Port Aggregator (VEPA) and SComp v-ports, as well as the physical Ethernet ports of the device.

For more detail on the XL710 transmit scheduler features and operation, see Section 7.8.

1.1.5 Host memory cache

The XL710 LAN engine and FCoE 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. Here are some examples:

- The LAN engine uses two HMC context objects per Queue Pair (QP), one for the Transmit Queue (TQ) and one for the Receive Queue (RQ). Parameters in the TQ context include a Transmit Segmentation Offload (TSO) state. Parameters in the RQ context include a queue base address and associated pointers.
- The FCoE engine optionally uses an HMC context object for each Fibre Channel (FC) command. These include *Direct Data Placement (DDP) context* objects. These objects define a sink host buffer for an individual FC read or write command. Their use is defined in Section 9.4.1.

General information on HMC operation and configuration is provided in Section 7.9.

1.1.6 LAN engine

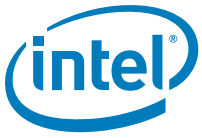
The LAN engine implements the host programming interface for traditional LAN traffic in both virtualized and non-virtualized scenarios. The XL710 implements 384 VSIs (virtual-NICs) used to distribute traffic to PCI physical functions and virtual functions. These VSIs can connect to the host via 1536 LAN QPs.

The LAN engine also implements all of the XL710's performance optimizations for traditional LAN traffic. This includes packet checksum offloads, Transmission Control Protocol/User Datagram Protocol (TCP/UDP Segmentation Offload), RSS and others.

1.1.7 FCoE engine

FC is the predominant protocol used in Storage Area Networks (SAN). FCoE enables a connection between an Ethernet storage initiator and legacy FC storage targets or a complete Ethernet connection between a storage initiator and a device.

As opposed to existing FC Host Bus Adapters (HBAs) that fully offload the FC protocol, the XL710 is focused on offloading the main data path of Input/Output I/O Read and Write commands.



1.1.8 System management

The XL710 participates in system management by providing networking services to platform management controllers (also called Baseboard Management Controller - BMC). The XL710 is also accessible to these devices to be managed as any platform resource that is managed by the system.

Networking services are provided through the Pass-Through (PT) functionality described in Section 10.1. Several sideband channels are provided to connect to a Management Controller (MC):

- System Management Bus (SMBus)
- Network Controller Sideband Interface (NC-SI)
- PCIe; together with Management Component Transport Protocol (MCTP)

The XL710 also supports communication between local Software (SW) agents and the local MC through an OS to Baseboard management Controller (BMC) capability described in Section 10.4.

1.1.9 EMP

The Embedded Management Processor unit (EMP) handles all management duties that cannot be performed by the XL710's device drivers, and must be carried out on-chip. This includes performing parts of the XL710 power-on sequence, handling AQ commands, initializing the XL710 Ethernet ports, participating in various fabric configuration protocols such as DCBX and other Link Layer Discovery Protocol (LLDP) protocols, fielding configuration requests received on one of the XL710's BMC management interfaces such as NC-SI, and handling special configuration requests received off an Ethernet port.

1.1.10 Internal switch

The internal switch handles the XL710 packet buffering and also implements all its internal Ethernet switching capabilities. The internal switch can logically support up to 16 VLAN-aware bridges, connecting up to 384 VSIs out to the XL710 Ethernet ports via optional EVB S-Components. The internal switch implements the filtering and forwarding behaviors expected by 802.1Q VLAN-aware bridges.

1.1.11 Receive scheduler

The receive scheduler prioritizes received packets according to the ETS settings programmed at each Ethernet port.

1.1.12 Receive filters

The XL710 receive filters implement all of the XL710's logic for queue selection. For each received packet, the forwarding process carried out by the internal switch selects which VSI the packet is associated with. The receive filters then determine which engine(s) handle the packet (EMP, LAN engine, FCoE engine) and which selected VSI RQs the packet is associated with. Receive filters include flow director, RSS filters, etc.

1.1.13 Various interfaces

1.1.13.1 Serial Flash interface

The XL710 provides an external Serial Peripheral Interface (SPI) serial interface to connect a Flash device. The XL710 supports serial Flash devices with up to 64 Mb (8 MB) of memory.

The Flash device is required for storage of device firmware, device configuration parameters, identifiers that vary per adapter (like MAC addresses), and register overrides that autoload automatically after reset.

More information on the Serial Flash interface is available in Section 3.3.

1.1.13.2 SMBus interface

SMBus is an optional interface for pass-through and/or configuration traffic between an external BMC and the XL710. The XL710's SMBus interface supports standard SMBus at 100 KHz and extensions up to a frequency of 1MHz. Refer to Section 2.2.4 for the pin descriptions, Section 10.5 for SMBus programming, and Section 14.6 for the timing characteristics.

1.1.13.3 NC-SI interface

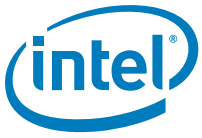
NC-SI is an optional interface for pass-through and/or configuration traffic between an external BMC and the XL710. Refer to Section 2.2.3 for the pin descriptions, Section 10.6 for NC-SI programming, and Section 14.6 for the timing characteristics.

1.1.13.4 Software Definable Pins (SDPs)

SDPs are typically used to exchange information with or to control external devices (such as PHY devices) under the XL710 software driver control. See Section 3.4.2 for more details.

1.1.13.5 Light-emitting Diodes (LEDs)

The XL710 implements eight output drivers intended for driving external LED circuits. The XL710 can be configured so that either two of these outputs are allocated per port or four outputs are allocated per port (this later configuration is used when two ports are disabled). 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 3.4.1 for more information.

1.2 Features

This section lists the XL710's feature set.

Note: Refer to the *Intel® Ethernet Controller XL710 Feature Support Matrix* for a list of features and interfaces supported by the XL710.

Table 1-2. PCIe host interface features[1]

Description
PCIe v3.0 (8GT/s or 5GT/s or 2.5GT/s)
Number of lanes: x8, x4, x1
Supports optional PCIe link lane reversal
Requester Features • 64-bit address support for systems with more than 4 GB of physical memory • Maximum of 112 outstanding requests, allocated fairly as needed amongst PCI Functions • Maximum payload size supported: 2 KB • Maximum read request size supported: 4 KB • All requests use TC TC0/VC0, the best effort service class for general purpose I/O • Optimized support for relaxed Ordering transaction attribute • Optimized support for TLP Processing Hints (TPH) • Optimized support for ID-based Ordering (IDO)

Table 1-2. PCIe host interface features[1]

Description
Completer Features • Up to 16 Peripheral Component Interconnect (PCI) PFs using Alternative Routing-ID Interpretation (ARI) and up to 8 PFs using legacy Routing-ID. Device can be configured to disable/hide any number of these PFs. The XL710 is a Multi-function Per Port (MFP) device, meaning that it can support multiple PFs sharing a single Ethernet port. The XL710 can be configured with up to 8 PFs sharing one Ethernet port. • Each PF implements these Base Address Registers (BARs) — Memory BAR for direct access to most Internal Registers. This BAR can be configured to define either a 32-bit or 64-bit base address. It's size is 256 KB or larger. — MSI-X BAR for direct access to MSI-X-related structures. This BAR can be disabled/hidden. It can be configured to define either a 32-bit or 64-bit base address and it's size is 32 KB. — Expansion ROM BAR for direct host access to one of the XL710's option ROM images. This BAR can be disabled/hidden. It defines a 32-bit base address and it's size is 64 KB or larger. — I/O BAR for indirect access to most Internal Registers. This BAR can be disabled/hidden. It defines a 32-bit base address and it's size is 32 bytes. • Per-PF capabilities list (some of these can be disabled/hidden) — PCI Power Management — MSI — MSI-X — PCIe — Vital Product Data (VPD) Each PF can access a single shared instance of VPD storage, with a size up to 256 bytes — Advanced Error Reporting (AER) — Device Serial Number — Alternative RID Interpretation (ARI) — SR-IOV — TPH Requester — Access Control Services (ACS) — Secondary PCIe
Reliability • AER • Support for optional End-to-End CRC (ECRC) generation and checking • Recovery from data poisoning • Completion timeout
Power Management • Supports the PCI Power Management specification and section 5 of the PCI Express Base Specification • Active state power management (L0s and L1 states) • Does not support D1 or D2 power management states • Does support D0uninitialized, D0active, D3hot, and D3cold power management states • D3hot transition to D0 does preserve configuration context (as indicated by the RO PCI configuration bit No_Soft_Reset=1b)
Interrupt Functionality • INTx: Supports four INTx interrupts conveyed using the PCIe INTx virtual wire signalling mechanism. A given PF can only map a single INTx. • MSI: Supports one Message-Signaled Interrupt (MSI) per PF. • MSI-X: Supports up to 1168 MSI-X vectors, shared among PFs and VFs. Up to 129 MSI-X interrupts can be assigned to a single PF and up to 17 MSI-X interrupts can be assigned to a single VF, depending on the number of enabled functions. • Sophisticated interrupt moderation using Interrupt Throttling (ITR) and Interrupt Rate Limiting (INTRL).

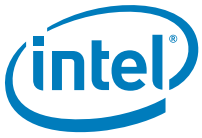


Table 1-2. PCIe host interface features[1]

Description
I/O Virtualization • Supports PCI-SIG SR-IOV Specification with up to 128 VFs. VFs can be flexibly assigned to PFs, although reassignment requires re-enumeration of the PCI bus hierarchy. • Each VF implements these BARs — Memory BAR for restricted access to Internal Registers. This BAR can be configured to define either a 32-bit or 64-bit base address. It's size is 16KB or larger. — MSI-X BAR for direct access to MSI-X-related structures. This BAR can be configured to define either a 32-bit or 64-bit base address. It's size is 16KB. • Per-VF Capabilities list (some of these can be disabled/hidden) — MSI-X — PCIe — AER — ARI — TPH requester — ACS

1. The total throughput supported by XL710 is 40 Gb/s, even when connected via two 40 Gb/s connections.

Table 1-3. Link layer Ethernet port features

Description
Ethernet Speeds and Interfaces 40 Gb/s: 2 ports of XLAUI, XLPPI, KR4, or CR4 10 Gb/s: 2 ports of XAUI or KX4, or 4 ports of KR or SFI 1 Gb/s: 4 ports of KX or SGMII 100 Mb/s: 4 ports of SGMII
The XL710'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), etc.
Full-duplex operation at all supported speeds
Integrates support for IEEE Std 802.3 Clause 73 Auto-Negotiation for Backplane Ethernet, including Clause 73 extensions for 40 Gb/s speed defined in IEEE Std 802.3ba.

Table 1-4. Transmit and receive scheduling

Description
Queue Sets • Supports up to 1024 LAN/FCoE queue sets (768 typically available) • Each queue set is assigned to the TC of a particular VSI • Round Robin (RR) bandwidth distribution between TQs assigned to same queue set • RR bandwidth distribution between LAN/FCoE queue sets belonging to same TC of VSI
Quanta: • Transmit work is scheduled in units of configurable per chip Quanta bytes. • Quanta can be configured to be 1KB, 2KB, 4KB, 8KB, 16KB, 32KB or 64KB.

Table 1-4. Transmit and receive scheduling

Description
VSI: • Supports up to 384 VSIs with average of two TCs allocated per VSI • Independent ETS/SLA configuration per VSI • Configurable bandwidth limit per VSI.
VEB/VEPA: • Supports up to 16 VEB/VEPA switching components • Configurable bandwidth allocation among VEB/VEPA VSIs • Configurable bandwidth Limit of VEB/VEPA egress port
S-components: • Supports up to 4 S-components, one per physical port • Configurable bandwidth allocation among S-channels • Independent ETS configuration per S-component egress port • Configurable bandwidth limit of S-component egress port
Bandwidth Distribution: • Supports two bandwidth allocation modes — Relative bandwidth allocation - used for ETS, and bandwidth allocation between ingress ports of the switching component — Best effort bandwidth allocation - used for SLA • Supports three arbitration schemes — Weighted RR — Weighted Strict Priority (WSP) — Combination of WSP and weighted RR • On any given virtual link, minimum bandwidth allocation is 1% of the virtual link bandwidth
Bandwidth Limit: • Configurable bandwidth limit in range of 40 Gb/s - 1 Mb/s with increment of 1 Mb/s • Configurable maximum bandwidth accumulation with a maximum of 200%

Table 1-5. LAN engine features

Description
VSI Support For each of the XL710's 384 allocated VSIs, the LAN engine allocates the following independent resources: • The LAN engine supports a total of 1536 LAN QPs (LQPs) — A PF VSI can allocate and use up to 1536 LQPs — A VF VSI can allocate and use up to 16 LQPs • Statistics: One set of IEEE Std 802.3 Clause 30 hardware statistics counters per VSI • TSO context: For each TQ, the LAN engine allocates storage for TSO context like TCP sequence numbers and other header fields changed by the TSO process. This enables each TQ to make independent progress on its TSO operations. The following resources are notably not allocated per VSI: • Interrupts: Interrupt vectors are typically allocated per CPU core. Software controls how a single vector is shared amongst interrupt causes (like LAN queues). Typically sharing is limited to a small number of LAN queues to minimize polling performance loss. • RSS: RSS logic is implemented per-PCI Function (both PFs and VFs), so there are 144 instances. Any PF or VF allocated multiple VSIs must share its RSS logic and programming among the VSIs. •

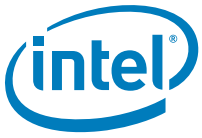


Table 1-5. LAN engine features

Description
Programming Interface. • LAN TQ descriptors: There are different LAN TQ descriptors that define: packet data, transmit context (such as TSO information), flow director filter programming, and various types of FCoE context programming. All TQ descriptors are 16 bytes. The packet data descriptor defines one fragment. If a packet or TSO is comprised of multiple fragments, packet data descriptors can be chained, up to a limit of eight per transmitted Ethernet packet (TSOs are allowed more). TQ completions are signalled either by descriptor writeback, or by updating a software head pointer in host memory. Software controls this signalling mechanism on a per-TQ basis. Software controls signalling frequency on a per-descriptor basis. • LAN RQ descriptors: Each LAN RQ descriptor defines up to two fragments, one for the packet, and one for the header (used when optional header splitting is enabled). For a given RQ, the fragment lengths are fixed and programmed into RQ context. Maximum size of the packet fragment is 16 KB and maximum size of the header fragment is 2 KB. A received packet is allowed to span multiple RQ descriptors, up to a limit of five. When the LAN engine has filled a LAN RQ descriptor, it writes completion status back to the descriptor. RQ descriptors can be configured per queue to be either 16 or 32 bytes in size, depending on the amount of completion status detail desired. • LAN Q properties (TQ and RQ): LAN Qs are mapped into host memory as physically contiguous ring buffers. Maximum LAN Q depth is configurable on a per-Q basis. Supported values range from 8 to 8 KB entries. Maximum LAN TQ size is 8 KB entries x 16 bytes = 128 KB. Maximum LAN RQ size is 8 KB entries x 32 bytes = 256 KB. Each LAN Q can be individually disabled.
Performance Optimizations (available to every VSI) • Packet checksum offloads: Calculates IP, UDP, TCP, and Stream Control Transmission Protocol (SCTP) checksums for insertion into transmitted packets and for integrity checking on received packets. Includes support for UDP and TCP checksums in both IPv4 and IPv6 datagrams. • TCP/UDP segmentation offload, also referred to Large Send Offload (LSO) for transmitted IPv4 and IPv6 packets. The maximum size of a TSO operation is 256 KB. • Received packet header splitting. This feature enables a received packet's header to be placed in a different host buffer than the packet payload. Each RQ can be configured independently to perform header splitting at a specified header layer: none, IP, TCP, UDP, etc. • RSS: PF instances of the RSS logic implement 256 entry indirection tables, supporting up to 64 RQs/CPUs. VF instances of the RSS logic implement 64 entry indirection tables, supporting up to 16 RQs/CPUs.
Intel Virtual Machine Device Queues (VMDq) Functionality VMDq defines the hardware offloads that support Virtual Bridges (VBs) implemented in software (usually in a VMM). The XL710 supports both VMDq version 1 and version 2. • Some of the major features of VMDq version 1 — Layer 2 filters for sorting packets based on MAC address — Layer 2 filters for sorting packets based on VLAN tags — 1536 LAN QPs that can be flexibly allocated to the PFs for VMDq flows — Default queue mechanism for non-matching packets — MSI-X interrupt per queue • Some of the major features of VMDq version 2 (VMDq2) — Internal switching from a TQ to a RQ — Broadcast and multicast replication — Ability to sort packets based on a combination VLAN tag and MAC address filter — Anti-spoofing transmit filters (VLAN and MAC)
Preboot Execution Environment (PXE) Supports the PXE remote boot standard. Also supports Internet Small Computer System Interface (iSCSI) boot via expansion ROM firmware.
Flow director filters: 8 K perfect-match filters stored on-chip

Table 1-6. FCoE engine features

Description
Fibre Channel Protocol (FCP) Transmit Offloads • Generate FC-CRC • Insert FC padding bytes • Supports FCoE TSO that enables an initiator or target to issue a sequence of FCP_DATA packets (up to 256 KB-1 byte) using a single command. • Supports FCoE Enhanced Transmit Segmentation Offload (ETSO) that enables a target to issue a sequence of FCP_DATA packets (as in TSO) plus an FCP_RSP packet using a single command.
FCP Receive Offloads • Validate FC-CRC • Supports FCoE Direct Data Placement (DDP) that enables an initiator or target to offload the handling of all received FCP_DATA packets within an exchange flow. This offload includes automatic transfer of each received FCP_DATA payload directly into buffers pre-allocated by the FCoE stack for this exchange flow. FCoE DDP reduces CPU overhead by eliminating buffer copies and by reducing the number of packets processed by the FCoE stack per exchange flow. • Strip FC padding bytes from packets posted to the DDP
FCP Exchanges Supported The number of outstanding FCP exchanges supported is an often-reported feature of FC HBAs. XL710 DDP offloads use a shared context table whose maximum size per PCI Function is 4096 entries. Therefore, the XL710 supports up to 4096 outstanding FCP Exchanges per PF with DDP offload.
Features Implemented in Host Software Standard FC HBAs typically implement the following features in adapter hardware or firmware. The XL710 FCoE architecture implements these features in host software *with no adapter hardware- or firmware-imposed limits*. The following descriptions are included for informative purposes only, and are subject to change. • Number of FCoE targets (PLOGI) supported per Ethernet port: 1024 for Windows, 256 for Linux* • Number of fabric logins (FLOGI) supported per Ethernet port: 1 for Windows, multiple for Linux • Support for FC tape devices?: No for Windows, No for Linux • N_Port ID Virtualization (NPIV) such as the number of N_Port IDs supported per Ethernet port: not supported for Windows, 255 for Linux (this is a FC architecture limit)
Supports FCoE Boot via Expansion ROM
FCoE-specific Packet Filtering • Direct received FCoE packets to a matching LAN receive queue by Rx exchange ID. Support multiple LAN receive queues for CPU load balance. • Direct received FCoE packet to a matching DDP context using exchange ID, optional VLAN (if header present), FC destination ID, source MAC address
FCoE Statistics Implements ~20 FCoE-specific hardware statistics counters per VSI, including FCoE Tx/Rx packet count, FCoE Tx/Rx Dword count, FC CRC error count, etc.

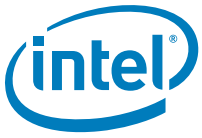


Table 1-7. Internal switching features

Description
VSI Support The XL710 supports a total of 384 VSIs. VSI assignment is flexible, but the choice to support 384 VSIs is motivated by the following usage example: • 256 VSIs for VFs or VMDq2 • 32 VSIs for PFs • 24 control ports (16 for VEBs/VEPAs, 4 for Ethernet ports, 4 for S-comp/E-comp) • 16 mirror ports • 4 for EMP • 20 extras
EVB and Bridge Port Extension (BPE) Functionality • Supports EVB as defined in the IEEE P802.1Qbg specification • The XL710 supports up to four S-components or four eBridge port extenders (one per port), up to 512 S-tags, and up to 384 S-channels • The XL710 supports up to 16 internal VEBs or VEPAs • A single Ethernet port can connect up to 384 S-channels. Connecting to these S-channels can be VEBs, VEPAs, or port extenders. The XL710 must be configured for either VEPA or port extenders (both are not supported simultaneously). Given these rules, plus the global constraints described in the preceding bullets, a single Ethernet port can connect either: — Up to 16 VEBs or VEPAs. This case only requires up to 16 S-channels. — Up to 384 EBPs or VEBs. Of the 384, only up to 16 can be VEBs. • Each VEB is compliant with the IEEE 802.1Q Ethernet VLAN-aware bridge specification • Supports MAC address forwarding table with 1536 entries • Supports MAC and VLAN address forwarding table with 2048 pairs • Supports VLAN insertion and removal for up to 512 VLAN tags located anywhere in the 4K VLAN space • Supports Private VLANs • Supports port mirroring • Storm control - option to limit multicast or broadcast traffic to prevent DoS attacks • Each VSI can be programmed for promiscuous reception of unicast, broadcast and multicast packets
Internal switching operates independently of the state of the LAN ports (also when LAN ports are down)

Table 1-8. System manageability features

Description
Sideband Interfaces for connection to an external BMC • SMBus operating at up to 1 Mb/s • DMTF-compliant NC-SI Interface at 100 Mb/s • PCIe (together with MCTP)
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 to 7 IPv4/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.
Statistics For each Ethernet port, the XL710 implements the statistics defined in the following standards: — IEEE Std 802.3 Clause 30 — RFC 2819 - RMON Ethernet statistics group For each VSI, the XL710 implements the statistics defined in the following standard: IEEE Std 802.3 Clause 30.

Table 1-9. Power management features

Description
Supports the PCIe power management features defined in the "Power Management" entry in Table 1-2.
Energy Efficient Ethernet (EEE) Supports EEE as defined in the IEEE P802.3az specification. EEE is supported with both internal backplane Ethernet PHYs and with select external PHYs. • Internal backplane Ethernet PHY modes that support EEE: KR, KX4, KX • Interfaces that support EEE using an external PHY: KR, XAUI, SFI, SGMII
Low Power Link Up (LPLU) In a backplane Ethernet environment, when the XL710 is entering a low power state like $D3_{COLD}$, the XL710's LPLU logic attempts to re-autonegotiate its backplane Ethernet ports to their lowest possible link speed.
Advanced Power Management (APM) Wake Up Supports APM wake up per-PF: • Magic packet filter. When a magic packet arrives, and if the Magic packet filter is enabled, this can trigger a wake up.

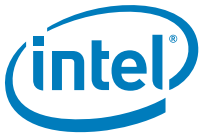


Table 1-10. General features

Description
Serial Flash Interface
Configurable LED operation for software or Original Equipment manufacturer (OEM) customization of LED displays. Default Configuration by Non-volatile Memory (NVM) for all LEDs for pre-driver functionality
Device disable capability
Package Size 25 x 25 mm
Watchdog timer
Time Sync (IEEE 1588)
Firmware and Expansion ROM Management • All firmware and expansion ROM code is stored in flash memory and is field upgradable • Supports field upgrades via host software utility or BMC program • Supports independent update of these objects: EMP firmware and Expansion ROM code • Supports secure authentication of during an update procedure: EMP firmware Expansion ROM code and protected with a digital signature produced using SHA256 hash and 2048b RSA encryption. • The firmware and expansion ROM field upgrade procedure can be carried out while the XL710 is in normal operation. The XL710 does not load and use the new content until a reset occurs. • Failure of the firmware or expansion ROM field upgrade procedure (due to interrupting of Flash programming, or authentication failure) never prevents a reattempt of the field upgrade procedure. Normally the XL710 reverts to the previously saved firmware or expansion ROM content when a field upgrade fails.
SDPs • 22 total SDPs: Four groups containing 4 pins, plus 6 groups containing a single pin. Each group can be configured for ownership by a different XL710 software driver, and can be independently programmed. • SDPs are typically used to exchange information with or to control external devices (such as PHY devices) under XL710 software driver control. • Each SDP can be configured as an input, output, or interrupt source.
Device / Port / PCI Function Disable The XL710 implements two strapping pins that, together with NVM settings, enable the entire XL710, or selected Ethernet ports and their associated PCI functions or selected PCI functions to be disabled. Disabling occurs before PCI enumeration, so the disabled resources are completely hidden.

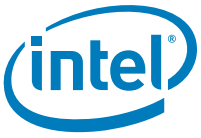
1.3 Conventions

1.3.1 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).

1.3.2 Byte ordering

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



1. Network Order - Ethernet always transmits multiple-bytes fields with the Most Significant (MS) byte first
2. 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:
   a. Little Endian (LE) notation — The MS byte of the logical entity is mapped to the highest byte address of the structure
   b. Big Endian (BE) notation — The MS byte of the logical entity is mapped to the lowest byte address of the structure

A few examples follow:

Example 1: A 32-bit counter is equal to 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 - lower byte address

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

0x67 - highest byte address

0x45

0x23

0x01 - lower byte address

Example 2: An L2 type register that holds the value of IPv4 header is equal to 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 - lower byte address

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

0x00 - highest byte address

0x08 - lower byte address

Example 3: A 48-bit Ethernet MAC address equals to 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

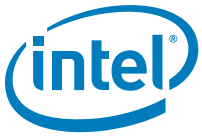
0x11

0x23

0x48

0xA9

0xBE - lower byte address



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

0xBE - highest byte address

0xA9

0x48

0x23

0x11

0x00 - lower byte address

The following rules determine the Endian-ness of the XL710 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, FCoE
- 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)
- 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

1.4 Overview of standards

Table 1-11 lists standards relevant to the XL710.

Table 1-11. Standards supported by the XL710

Category	Description
base	Title: Computing the Internet Checksum 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: A TCP/IP Tutorial 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: Implementing the Internet Checksum in Hardware. Document: http: //www.ietf.org/rfc/rfc1936.txt Description: Techniques for efficiently implementing the Internet checksum in hardware.
Ethernet	Title: "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications" (IEEE Std 802.3-2008) Document: available from standards.ieee.org/getieee802 Description: Specifies the Ethernet MAC and PHY layers up to 10 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), etc.
Ethernet	Title: "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications – Amendment 4: Media Access Control Parameters, Physical Layers and Management Parameters for 40 Gb/s and 100 Gb/s Operation" (IEEE Std 802.3ba-2010) Document: available from standards.ieee.org/getieee802 Description: Defines Ethernet MAC and PHY layers for 40 and 100 Gb/s.
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 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 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 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 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.

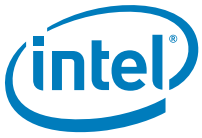


Table 1-11. Standards supported by the XL710 (Continued)

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 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: "Virtual Bridged Local Area Networks — Amendment 18: Enhanced Transmission Selection for Bandwidth Sharing Between Traffic Classes" (IEEE P802.1Qaz-2011) Document: available from 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	Title: "Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks — Amendment 21: Edge Virtual Bridging" (IEEE P802.1Qbg-2012) Document: available from 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 XL710. This includes definition of things like S-channels, which enable the multiplexing of multiple virtual channels on a single physical LAN, VSIs, VEBs, VEPAs, etc. 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 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 standards.ieee.org/getieee802 Description: Defines Link Layer Discovery Protocol (LLDP) that enables a server to advertise it's 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 standards.ieee.org Description: Defines a protocol that enables precise synchronization of clocks in systems communicating via packet networks.
Ethernet	Title: "SFF-8436 Specification for QSFP+ Copper and Optical Modules, rev 3.1, 4/22/2009" Document: ftp://ftp.seagate.com/pub/sff/SFF-8436.PDF (see www.sffcommittee.com) Description: Defines the Quad Small Form Factor Pluggable (QSFP+) module mechanicals, electrical interface and pinout, etc.
Ethernet	Title: "SFF-8431 Specifications for Enhanced Small Form Factor Pluggable Module SFP+, rev 4.1, 7/6/2009" Document: ftp://ftp.seagate.com/sff/SFF-8431.PDF (or see www.sffcommittee.com) Description: Defines the Simple Firmware interface (SFI) high speed electrical interface to a Small Form-factor Pluggable (SFP+) optical module. Also defines the SFP+ management interface.

Table 1-11. Standards supported by the XL710 (Continued)

Category	Description
Ethernet	Title: RMII Specification, rev 1.2, 3/20/1998 Description: Reduced pin count interface used in place of IEEE standard Media Independent Interface (MII). The XL710 has an Reduced Media Independent Interface (RMII) 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, ver 0.3, 11/17/2009 Document: EA-Ethernet Jumbo Frames v0 3.pdf Title: Extended Frame Sizes for Next Generation Ethernets Document: AlteonExtendedFrames_W0601.pdf Title: Extended Ethernet Frame Size Support Document: draft-kaplan-isis-ext-eth-02.txt Description: Documents describing jumbo frames. Included for informative purposes only.
test	Title: IEEE Standard Test Access Port and Boundary-Scan Architecture (IEEE Std 1149.1-2001) Document: available from standards.ieee.org Description: Defines a standard interface through which instructions and test data are communicated to an integrated circuit for component- and circuit board-level testing.
Ethernet/ IP	Title: Standard for the Transmission of IP Datagrams over Ethernet Networks Document: http://www.ietf.org/rfc/rfc894.txt Description: This specifies the method for transmitting IP datagrams over Ethernet.
Ethernet/ IP	Title: Standard for the Transmission of IP Datagrams over IEEE 802 Networks Document: http://www.ietf.org/rfc/rfc1042.txt Description: This specifies the method for transmitting IP datagrams over IEEE 802.3 networks.
iARP	Title: Address Resolution Protocol (ARP) Document: http://www.ietf.org/rfc/rfc0826.txt Description: Protocol to convert IP addresses to Ethernet addresses
IP	Title: Internet Protocol Document: http://www.ietf.org/rfc/rfc0791.txt Description: Base specification for IPv4.
IP	Title: IP Datagram Reassembly Algorithms Document: http://www.ietf.org/rfc/rfc0815.txt
IP	Title: Internet Protocol, Version 6 Document: http://www.ietf.org/rfc/rfc2460.txt Description: Base specification for IPv6.
IP	Title: Neighbor Discovery for IP version 6 (IPv6) 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.

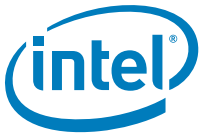


Table 1-11. Standards supported by the XL710 (Continued)

Category	Description
IP	Title: Multicast Listener Discovery (MLD) for IPv6 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: Multicast Listener Discovery Version 2 (MLDv2) for IPv6 Document: http://www.ietf.org/rfc/rfc3810.txt Description: Updates RFC 2710. This is an important standard for power management.
IP	Title: TCP Processing of the IPv4 Precedence Field Document: http: //www.ietf.org/rfc/rfc2873.txt Description: Corrects a conflict between TCP as defined in RFC793 and DiffServ in handling of the IPv4 precedence field.
TCP	Title: Transmission Control Protocol Document: http://www.ietf.org/rfc/rfc0793.txt Description: Base specification for TCP.
TCP	Title: Window and Acknowledgement Strategy in TCP Document: http://www.ietf.org/rfc/rfc813.txt
TCP	Title: Congestion Control in IP/TCP Internetworks 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 Acknowledgements (ACKs) outstanding.
TCP	Title: TCP Extensions for High Performance 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).
TCP	Title: Known TCP Implementation Problems 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.
TCP	Title: TCP Congestion Control Document: http://www.ietf.org/rfc/rfc5681.txt Description: TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms.
TCP	Title: The NewReno Modification to TCP's Fast Recovery Algorithm 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.
TCP	Title: TCP Problems with Path MTU Discovery 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.

Table 1-11. Standards supported by the XL710 (Continued)

Category	Description
TCP	Title: Computing TCP's Retransmission Timer 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: Increasing TCP's Initial Window 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 4KB.
TCP	Title: TCP Congestion Control with Appropriate Byte Counting (ABC) 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.
UDP	Title: User Datagram Protocol Document: http://www.ietf.org/rfc/rfc0768.txt Description: Base specification for UDP.
Tunnelling	Title: NVGRE: Network Virtualization using Generic Routing Encapsulation Document: http://datatracker.ietf.org/doc/draft-sridharan-virtualization-nvgre/?include_text=1 Description: MAC in Generic Routing Encapsulation (GRE) over IP encapsulation
Tunnelling	Title: VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks Document: http://datatracker.ietf.org/doc/draft-mahalingam-dutt-dcops-vxlan/?include_text=1 Description: MAC in UDP encapsulation
Storage	Title: Fibre Channel Framing and Signaling-2 (FC-FS-2), v1.01, 8/8/2006, INCITS 424-2007 Document: available from http://www.incits.org/ Description: Defines the framing and signaling interface of a FC serial link, which supports various FC-4 mapping layers associated with upper level protocols (such as SCSI, IP, SBCCS, VI).
Storage	Title: Fibre Channel Backbone - 5 (FC-BB-5), v2.0, 6/4/2009, INCITS 462-2010 Document: available from http://www.incits.org/ Description: Defines mappings for transporting FC over different network technologies, including FC over Ethernet.
Storage	Title: Fibre Channel Link Services (FC-LS), v1.62, 12/4/2006, INCITS 433-2007 Document: available from http://www.incits.org/ Description: Defines FC link services control frames used for fabric setup and messaging (like fabric login).
Storage	Title: Fibre Channel Protocol for SCSI - Third Version (FCP-3), 11/25/2009, INCITS 416-2006 Document: available from http://www.incits.org/ Description: Defines an FC-4 FC mapping layer for transmitting SCSI command, data, and status information using FC-FS-2 services.
Storage	Title: SCSI Block Commands - 3 (SBC-3), 11/25/2009, T10/1799-D Document: drafts available from http://www.t10.org/ Description: Defines the SCSI Block Commands - 3 (SBC-3) command set, which maintains a high degree of compatibility with the SBC-2 command set, INCITS 405-2005. Of particular interest to the XL710, SBC-3 defines protection information (also known as *Data Integrity Field* or DIF), an industry standard for data integrity that protects data between a block storage initiator and the storage device.

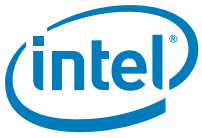


Table 1-11. Standards supported by the XL710 (Continued)

Category	Description
Mgmt	Title: System Management Bus (SMBus) Specification, v2.0, 8/3/2000” Document: www.smbus.org/specs/smbus20.pdf Description: A two-wire interface for communication of management information, based on the principles of operation of I^2C.
Mgmt	Title: Network Controller Sideband Interface (NC-SI) Spec, v1.0.1, 1/10/2013 Document: dmtf.org/sites/default/files/standards/documents/DSP0222_1.0.pdf Description: Standardizes the sideband communication interface between a NIC (the XL710) and a BMC.
Mgmt	Title: Management Component Transport Protocol (MCTP) Base Specification, rev 1.1.0, 4/22/2010 Document: dmtf.org/sites/default/files/standards/documents/DSP0236_1.1.0.pdf Description: Specifies MCTP protocol.
Mgmt	Title: Management Component Transport Protocol (MCTP) SMBus/I2C Transport Binding Specification, rev 1.0.0, 7/28/2009 Document: dmtf.org/sites/default/files/standards/documents/DSP0237_1.0.0.pdf Description: Describes the binding of MCTP over SMBus.
Mgmt	Title: Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding Specification, rev 1.0.1, 12/11/2009 Document: dmtf.org/sites/default/files/standards/documents/DSP0238_1.0.1.pdf Description: Describes the binding of MCTP over PCIe.
Mgmt	Title: Management Component Transport Protocol (MCTP) IDs and Codes, rev 1.1.0, 11/3/2009 Document: dmtf.org/sites/default/files/standards/documents/DSP0239_1.1.0.pdf Description: Describes constants used by MCTP specifications.
Mgmt	Title: NC-SI Over MCTP Specification, rev 0.1, 8/17/2010 Document: dmtf.org/sites/default/files/standards/documents/DSP0261_1.0.pdf Description: Describes the encapsulation of NC-SI packets in MCTP.
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, rev 1.2, 3/3/2004 Document: available from www.pcisig.com Description: Defines a standard set of PCI peripheral power management hardware interfaces and behavioral policies.
Power Mgmt	Title: “Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications – Amendment: Media Access Control parameters, Physical Layers and management parameters for Energy-Efficient Ethernet” (IEEE P802.3az-2010) Document: available from standards.ieee.org/getieee802 Description: EEE defines a mechanism to reduce power consumption during periods of low link use.
Power Mgmt	Title: proxZZZy for sleeping hosts, February 2010 (ECMA-393) 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.

Table 1-11. Standards supported by the XL710 (Continued)

Category	Description
Statistics	Title: The Interfaces Group MIB Document: http://www.ietf.org/rfc/rfc2863.txt Description: Describes objects used for managing network interfaces.
Statistics	Title: Remote Network Monitoring MIB Document: http://www.ietf.org/rfc/rfc2819.txt Description: Defines objects for managing remote network monitoring devices. Includes the popular packet size histogram counters.
Statistics	Title: Microsoft NDIS 6.0 OID_GEN_STATISTICS Document: msdn.microsoft.com/en-us/library/ff569640(VS.85).aspx Description: Adapter statistics counters defined by Microsoft for NDIS 6.0.
Statistics	Title: MIB for the Internet Protocol (IP) Document: http://www.ietf.org/rfc/rfc4293.txt Description: IP version-independent IP MIB. Obsoletes RFC 2011, 2465, 2466.
Statistics	Title: MIB for the Transmission Control Protocol (TCP) Document: http://www.ietf.org/rfc/rfc4022.txt Description: IP version-independent TCP MIB. Obsoletes RFC 2012, 2452.
Statistics	Title: MIB for the User Datagram Protocol (UDP) Document: http://www.ietf.org/rfc/rfc4113.txt Description: Objects for managing implementations of UDP. Obsoletes RFC 2013.
PCI	Title: PCI Local Bus Specification, rev 3.0, 2/3/2004 Document: available from www.pcisig.com Description: Compliant with select sections, such as Appendix I Vital Product Data.
PCI	Title: PCI Hot-Plug Specification, rev 1.1, 6/20/2001 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, rev 3.0, 6/20/2005 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, rev 3.0, 11/10/2010 Document: available from www.pcisig.com Description: Contains the technical details of the PCIe architecture, protocol, link layer, physical layer, and software interface. Also defines some important power management features, including Optimized Buffer Flush/Fill (OBFF).
PCI	Title: PCI Express Card Electromechanical Specification, rev 2.0, 4/11/2007 Document: available from www.pcisig.com Description: Mechanical and electrical specifications for an Evo card form factor.

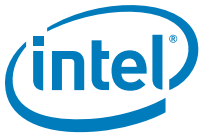


Table 1-11. Standards supported by the XL710 (Continued)

Category	Description
PCI-IOV	Title: Single Root I/O Virtualization and Sharing Specification, rev 1.1, 1/20/2010 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.
software	Title: Microsoft specification for "Receive Side Scaling" (RSS) from MSDN. Document: msdn.microsoft.com/en-us/library/ff567236%28v=VS.85%29.aspx Description: RSS is a network driver technology that enables the efficient distribution of network receive processing across multiple CPUs in multiprocessor systems.
IOSF	Title: Intel On-Chip System Fabric (IOSF) Specification. August 31, 2012 Revision 1.1 Document: TBD Description: Defines the interface between the IP and the IOSF fabric.

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

2.0 Pin interface

2.1 Pin descriptions

This section provides detailed descriptions of the XL710 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	Reserved
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.	
B	Input BIAS.	
CML-in	Current Mode Logic (CML) input signal.	
NCSI-in	NC-SI input signal.	
NCSI-out	NC-SI output signal.	
Pu	Internal pull-up resistor.	
Pd	Internal pull-down resistor.	

2.2 Pin assignment and description

The XL710 is packaged in a 25 mm x 25 mm 576-pin Flip-Chip Ball Grid Array (FCBGA) package. The following sections provide the signal names, pin/ball assignments and signal descriptions. The electrical specifications for the signals are described in Section 14.0.

2.2.1 PCIe interface pins

This section provides the pin assignment for PCIe interface signals. The AC/DC specifications for the PCIe interface signals are defined in Section 14.6.

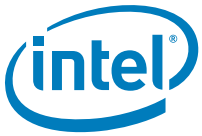


Table 2-2. PCIe interface signals

Reserved	Signal	Ball #	Type	Reserved	Description
	PE_CLK_p PE_CLK_n	AB23 AB24	A-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	Y23 Y24	A-out		PCIe Serial Data Output. A serial differential output pair running at 8 Gb/s or 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 8 GHz or 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end.
	PET_1_p PET_1_n	V23 V24	A-out		Same as previous.
	PET_2_p PET_2_n	T23 T24	A-out		Same as previous.
	PET_3_p PET_3_n	P23 P24	A-out		Same as previous.
	PET_4_p PET_4_n	J23 J24	A-out		Same as previous.
	PET_5_p PET_5_n	G23 G24	A-out		Same as previous.
	PET_6_p PET_6_n	E23 E24	A-out		Same as previous.
	PET_7_p PET_7_n	C23 C24	A-out		Same as previous.
	PER_0_p PER_0_n	AC20 AC21	A-in		PCIe Serial Data Input. A serial differential input pair running at 8 Gb/s or 5 Gb/s or 2.5 Gb/s. An embedded clock present in this input is recovered along with the data.
	PER_1_p PER_1_n	AA20 AA21	A-in		Same as previous.
	PER_2_p PER_2_n	U20 U21	A-in		Same as previous.
	PER_3_p PER_3_n	R20 R21	A-in		Same as previous.
	PER_4_p PER_4_n	K20 K21	A-in		Same as previous.
	PER_5_p PER_5_n	H20 H21	A-in		Same as previous.
	PER_6_p PER_6_n	D20 D21	A-in		Same as previous.
	PER_7_p PER_7_n	B20 B21	A-in		Same as previous.
	PE_WAKE_N	AA18	o/d, In		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.
	PE_RST_N	AD18	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. The AC/DC specifications for the Ethernet interface signals are defined in Section 14.6.

Table 2-3. Ethernet interface pins

Reserved	Signal	Ball #	Type	Reserved	Description
	REFCLKIN_p_XTAL_IN REFCLKIN_n_XTAL_OUT	P2 P1	CML-in		External Reference Clock Input or Crystal Oscillator Input/Output. These pins might be driven by a 25 MHz crystal or an external clock oscillator (25 MHz ±0.01%). The OSC_SEL pin is used to choose between a crystal source or an external clock oscillator.
	RXA_L3_p RXA_L3_n	B4 A4	A-in		Serial Data Input for Ethernet interface Group A. A serial differential input pair running at up to 10.3125 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 in KX, KR, and SFI modes for 4-port 10 Gb/s serial configuration. This lane is also used as one of the receive pairs in XAUI, KX4, KR4, XLAUI or XLPPI modes for 4-lane 10 Gb/s or 40 Gb/s configurations.
	RXA_L2_p RXA_L2_n	D4 D5	A-in		Same as previous.
	RXA_L1_p RXA_L1_n	F4 F5	A-in		Same as previous.
	RXA_L0_p RXA_L0_n	H4 H5	A-in		Same as previous.
	TXA_L3_p TXA_L3_n	C1 C2	A-out		Serial Data Output for Ethernet Interface Group A. A serial differential output pair running at up to 10.3125 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 in KX, KR, and SFI modes for 4-port 10 Gb/s serial configuration. This lane is also used as one of the transmit pairs in XAUI, KX4, KR4, XLAUI or XLPPI modes for 4-lane 10 Gb/s or 40 Gb/s configurations.
	TXA_L2_p TXA_L2_n	E1 E2	A-out		Same as previous.
	TXA_L1_p TXA_L1_n	G1 G2	A-out		Same as previous.
	TXA_L0_p TXA_L0_n	J1 J2	A-out		Same as previous.
	RXB_L3_p RXB_L3_n	U4 U5	A-in		Serial Data Input for Ethernet Interface Group B. A serial differential input pair running at up to 10.3125 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 in KX, KR, and SFI modes for 4-port 10 Gb/s serial configuration. This lane is also used as one of the receive pairs in XAUI, KX4, KR4, XLAUI or XLPPI modes for 4-lane 10 Gb/s or 40 Gb/s configurations.
	RXB_L2_p RXB_L2_n	W4 W5	A-in		Same as previous.
	RXB_L1_p RXB_L1_n	AA4 AA5	A-in		Same as previous.
	RXB_L0_p RXB_L0_n	AC4 AD4	A-in		Same as previous.

Reserved	Signal	Ball #	Type	Reserved	Description
	TXB_L3_p TXB_L3_n	T1 T2	A-out		Serial Data Output for Ethernet Interface Group B. A serial differential output pair running at up to 10.3125 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 in KX, KR, and SFI modes for 4-port 10 Gb/s serial configuration. This lane is also used as one of the transmit pairs in XAUI, KX4, KR4, XLAUI or XLPPI modes for 4-lane 10 Gb/s or 40 Gb/s configurations.
	TXB_L2_p TXB_L2_n	V1 V2	A-out		Same as previous.
	TXB_L1_p TXB_L1_n	Y1 Y2	A-out		Same as previous.
	TXB_L0_p TXB_L0_n	AB1 AB2	A-out		Same as previous.

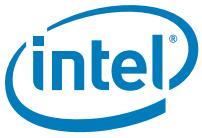


Table 2-4. External Ethernet PHY control - MDIO / I²C interface signals

Reserved	Signal	Ball #	Type	Reserved	Description
	MDIO0_SDA0	AD12	T/s, o/d		Management Data, when configured as an MDIO interface. Bi-directional signal for serial data transfers between the XL710 and the PHY management registers for port 0. **Note:** Tri-state buffer, requires an external pull-up device. I²C Data, when configured as 2-wire management interface for port 0. 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.
	MDC0_SCL0	AC12	O, o/d		Management Clock, when configured as an MDIO interface. Clock output for accessing the PHY management registers for port 0. MDC clock frequency is proportional to link speed. At 10 Gb/s Link speed, MDC frequency can be set to 2.4 MHz (default) or 24 MHz. I²C Clock, when configured as 2-wire management interface for port 0. One clock pulse is generated for each data bit transferred. **Note:** This I/O operates as an open drain buffer, and therefore requires an external pull-up device.
	MDIO1_SDA1	AC17	T/s, o/d		Management Data, when configured as an MDIO interface. Bi-directional signal for serial data transfers between the XL710 and the PHY management registers for port 1. **Note:** Tri-state buffer requires an external pull-up device. I²C Data, when configured as 2-wire management interface for port 1. 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.
	MDC1_SCL1	AB18	O, o/d		Management Clock, when configured as an MDIO interface. Clock output for accessing the PHY management registers for port 1. MDC clock frequency is proportional to link speed. At 10 Gb/s Link speed, MDC frequency can be set to 2.4 MHz (default) or 24 MHz. I²C Clock, when configured as 2-wire management interface for port 1. One clock pulse is generated for each data bit transferred. **Note:** This I/O operates as an open drain buffer, and therefore requires an external pull-up device.
	MDIO2_SDA2	AA12	T/s, o/d		Management Data, when configured as an MDIO interface. Bi-directional signal for serial data transfers between the XL710 and the PHY management registers for port 2. **Note:** Tri-state buffer, requires an external pull-up device. I²C Data, when configured as 2-wire management interface for port 2. 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.

Reserved	Signal	Ball #	Type	Reserved	Description
	MDC2_SCL2	AB12	O, o/d		Management Clock, when configured as an MDIO interface. Clock output for accessing the PHY management registers for port 2. MDC clock frequency is proportional to link speed. At 10 Gb/s Link speed, MDC frequency can be set to 2.4 MHz (default) or 24 MHz. I^2C Clock, when configured as 2-wire management interface for port 2. One clock pulse is generated for each data bit transferred. **Note:** This I/O operates as an open drain buffer, and therefore requires an external pull-up device.
	MDIO3_SDA3	AC18	T/s, o/d		Management Data, when configured as an MDIO interface. Bi-directional signal for serial data transfers between the XL710 and the PHY management registers for port 3. **Note:** Tri-state buffer requires an external pull-up device. I^2C Data, when configured as 2-wire management interface for port 3. 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.
	MDC3_SCL3	Y16	O, o/d		Management Clock, when configured as an MDIO interface. Clock output for accessing the PHY management registers for port 3. MDC clock frequency is proportional to link speed. At 10 Gb/s Link speed, MDC frequency can be set to 2.4 MHz (default) or 24 MHz. I^2C Clock, when configured as 2-wire management interface for port 3. One clock pulse is generated for each data bit transferred. **Note:** This I/O operates as an open drain buffer, and therefore requires an external pull-up device.

Note: If the I^2C is disconnected, an external pull-up should be used for the clock and data pins.

2.2.3 NC-SI interface pins

This section provides the pin assignment for NC-SI signals. The AC/DC specifications for the NC-SI interface signals are defined in Section 14.6.

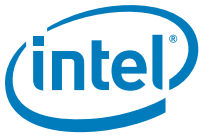


Table 2-5. NC-SI interface signals

Reserved	Signal	Ball #	Type	Reserved	Description
	NCSI_CLK_IN	AC11	NCSI-In		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	AB11	NCSI-Out		Carrier Sense/Receive Data Valid (CRS/DV).
	NCSI_RXD_0 NCSI_RXD_1	AA11 AC10	NCSI-Out		Receive Data. Data signals to the BMC.
	NCSI_TX_EN	AB10	NCSI-In		Transmit Enable.
C	NCSI_TXD_0 NCSI_TXD_1	AA10 AD11	NCSI-In		Transmit Data. Data signals from the BMC.
	NCSI_ARB_IN	Y8	NCSI-In		NC-SI Hardware Arbitration Input. If GL_MNG_HWARB_CTRL.NCSI_ARB_EN is cleared, this pin is internally pulled up.
	NCSI_ARB_OUT	Y10	NCSI-Out		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. The AC/DC specifications for the SMBus interface signals are defined in Section 14.6.

Table 2-6. SMBus interface signals

Reserved	Signal	Ball #	Type	Reserved	Description
	SMBCLK	E8	o/d		SMBus Clock. One clock pulse is generated for each data bit transferred. 3.3V tolerant.
	SMBD	E10	o/d		SMBus Data. Stable during the high period of the clock (unless it is a start or stop condition). 3.3V tolerant.
	SMBALRT_N	E14	o/d		SMBus Alert. Acts as an interrupt pin of a slave device on the SMBus. 3.3V tolerant.

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. The AC/DC specifications for the serial Flash memory interface signals are defined in Section 14.6.

Table 2-7. Serial Flash memory interface signals

Reserved	Signal	Ball #	Type	Reserved	Description
	FLSH_SI	B6	t/s		Serial data output that should be connected to the Serial Input (SI) of the SPI serial Flash memory.
	FLSH_SO	A7	In Pu		Serial data input that should be connected to the Serial Output (SO) from the SPI serial Flash memory.
	FLSH_SCK	A8	t/s		Flash serial clock operates at 12.5 MHz.
	FLSH_CE_N	B7	t/s		Flash chip select output.

2.2.6 General Purpose I/O (GPIO) pins

This section provides the pin assignment for GPIO signals. The XL710 has a total of 30 GPIO pins that can be configured as Software Definable Pins (SDP)s, LED drivers or dedicated hardware functions for connecting to external PHYs or IEEE 1588 auxiliary devices. The GPIO pins can also be associated with any of the physical ports. The following sections show the default configuration for GPIO pins that are reserved for specific use and hence named by default as SDP, LED or GPIO signals. However, the XL710 offers the flexibility to configure any of the GPIO pins (irrespective of the name) to different modes and associated with different ports as described in Section 3.4.

The AC/DC specifications for the GPIO signals are defined in Section 14.6.

2.2.6.1 LED interface pins

This section provides the pin assignment for LED interface signals. These are GPIO signals that are configured by default as LED interface pins associated with physical ports 0-3. The mode and port association for these pins can be configured (or changed) as described in Section 3.4.

The AC/DC specifications for the GPIO/LED signals are defined in Section 14.6.

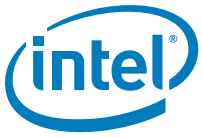


Table 2-8. LED interface signals

Reserved	Signal	Ball #	Type	Reserved	Description
	LED0_0	AD14	O		Port 0 LED0. Programmable LED indicates link up (default).
	LED0_1	AC14	O		Port 0 LED1. Programmable LED indicates 10 Gb/s (default).
	LED1_0	AD13	O		Port 1 LED0. Programmable LED indicates link up (default).
	LED1_1	AC13	O		Port 1 LED1. Programmable LED indicates 10 Gb/s (default).
	LED2_0	AB14	O		Port 2 LED0. Programmable LED indicates link up (default).
	LED2_1	AA14	O		Port 2 LED1. Programmable LED indicates 10 Gb/s (default).
	LED3_0	AB13	O		Port 3 LED0. Programmable LED indicates link up (default).
	LED3_1	AA13	O		Port 3 LED1. Programmable LED indicates 10 Gb/s (default).

2.2.6.2 SDP interface pins

This section provides the pin assignment for SDP interface signals. These are GPIO signals configured by default as SDP pins associated with physical ports 0-3. The mode and port association for these pins can be configured (or changed) as described in Section 3.4.

The AC/DC specifications for the GPIO/SDP signals are defined in Section 14.6.

Table 2-9. SDP interface pins

Reserved	Signal	Ball #	Type	Reserved	Description
	SDP0_0 SDP0_1 SDP0_2 SDP0_3	AD8 AC8 AB8 AA8	t/s Pu		Port 0 SDPs. General purpose 3.3V I/Os. Can be used to connect IEEE1588 auxiliary devices, 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.
	SDP1_0 SDP1_1 SDP1_2 SDP1_3	AC16 AB16 AB17 AA17	t/s Pu		Port 1 SDPs. General purpose 3.3V I/Os. Can be used to connect IEEE1588 auxiliary devices, 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.
	SDP2_0 SDP2_1 SPD2_2 SDP2_3	AD7 AC7 AB7 AA7	t/s Pu		Port 2 SDPs. General purpose 3.3V I/Os. Can be used to connect IEEE1588 auxiliary devices, 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.
	SDP3_0 SDP3_1 SDP3_2 SDP3_3	AA16 AC15 AB15 AA15	t/s Pu		Port 3 SDPs. General purpose 3.3V I/Os. Can be used to connect IEEE1588 auxiliary devices, 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.

2.2.6.3 Global GPIO interface pins

This section provides the pin assignment for Global GPIO interface signals. These are GPIO signals that are not assigned for a specific use or port. The mode and port association for these pins can be configured as described in Section 3.4.

The AC/DC specifications for the GPIO signals are defined in Section 14.6.

Table 2-10. Global GPIO pins

Reserved	Signal	Ball #	Type	Reserved	Description
	GPIO_0 GPIO_1 GPIO_2 GPIO_3	E18 E16 B18 A18	t/s Pu		General purpose 3.3V I/Os. Can be configured to be associated with any of the ports 0 to 3. Can be used as LED interface or SDP or to connect IEEE1588 auxiliary devices, low speed optical module interfaces, external PHY control or other similar usages. The pins can also be configured for use as external interrupt sources.
	GPIO_4 GPIO_5	Y12 Y14	t/s Pu		General purpose 3.3V I/Os. Can be configured to be associated with any of the ports 0 to 3. Can be used as LED interface or SDP or to connect IEEE1588 auxiliary devices, low speed optical module interfaces, external PHY control or other similar usages. The pins can also be configured for use as external interrupt sources.

2.2.7 Miscellaneous pins

This section provides the pin assignment for other miscellaneous signals. The AC/DC specifications for the miscellaneous signals are defined in Section 14.6.

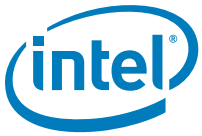


Table 2-11. Miscellaneous pins

Reserved	Signal	Ball #	Type	Reserved	Description
	LAN_PWR_GOOD	A14	In Pu		LAN Power Good. A 3.3V input signal. A transition from low-to-high initializes the XL710 into operation. If not used (POR_BYPASS = 0b), an internal Power-on-Reset (POR) circuit triggers the XL710 power-up. If the internal POR circuit is used to trigger device power-up, this signal should be connected to 3.3V. By default, internal POR should be used.
	POR_BYPASS	D19	In Pu		Bypass indication as to whether or not to use the internal POR or the LAN_PWR_GOOD pin. When set to 1b, the XL710 disables the internal POR circuit and uses the LAN_PWR_GOOD pin as a POR indication. When set to 0b, the XL710 uses the internal POR circuit. By default, the internal POR should be used unless the power supply sequencing timing requirements, as defined in Section 14.0,could not be met.
	OSC_SEL	AA9	t/s Pu		Defines the input clock connected to the REFCLKIN_p_XTAL_IN/REFCLKIN_n_XTAL_OUT pins: 0b = XTAL clock. 1 = OSC clock. This pin is a strapping option latched at LAN_PWR_GOOD.
	AUX_PWR	AB9	t/s		Auxiliary Power Available. When set, indicates that auxiliary power is available and the XL710 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	AC9	In		Main Power OK. Indicates that platform main power is up. Must be connected externally.

Reserved	Signal	Ball #	Type	Reserved	Description
	PCI_DIS_N	AD20	t/s Pu		This pin is a strapping pin latched while LAN_PWR_GOOD or PE_RST_N or In-band PCIe reset are asserted. If this pin is not connected or driven high during initialization, then all PCI functions as configured from NVM are enabled. If this pin is asserted/driven low during initialization, then all PCI functions that are allowed to be disabled as configured in NVM are disabled (see Section 4.3 for details).
	DEV_DIS_N	AD21	t/s 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 NVM configuration (See Section 4.3 for details). If this pin is not connected or driven high during initialization, then all the LAN ports and associated functions as configured from NVM are enabled for normal operation. If this pin is asserted/driven low during initialization then the LAN ports and associated functions as configured from NVM 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 4.3 for details).
	RBIAS RSENSE	M24 N24	B		BIAS. A 4.75 KΩ ±0.1% resistor should be connected between the RBIAS and RSENSE pins. Connect a resistor as close as possible to the chip. A resistor is used for internal impedance compensation and BIAS current generation circuitry.

2.2.8 Testability and debug pins

This section provides the pin assignment for JTAG testability interface signals. The AC/DC specifications for the JTAG testability signals are defined in Section 14.6.

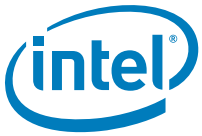


Table 2-12. Testability and debug pins

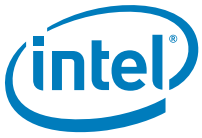
Reserved	Signal	Ball #	Type	Reserved	Description
	JTCK	B16	In		JTAG Clock Input.
	JTDI	A13	In Pu		JTAG Data Input.
	JTDO	B15	o/d		JTAG Data Output.
	JTMS	B13	In Pu		JTAG TMS Input.
	JRST_N	B14	In Pu		JTAG Reset Input. Active low reset for the JTAG port.

2.2.9 Reserved and no-connect pins

This section provides the pin assignment for reserved and no-connect pins.

Table 2-13. Reserved and no-connect pins

Reserved	Signal	Ball #	Type	Reserved	Description
	RSVDA11_NC RSVDA12_NC RSVDA17_NC RSVDB10_NC RSVDB11_NC RSVDB12_NC RSVDB17_NC	A11 A12 A17 B10 B11 B12 B17			RSVD* pins.
	RSVDB8_NC RSVDB9_NC RSVDC10_NC RSVDC11_NC RSVDC12_NC RSVDC13_NC RSVDC14_NC RSVDC15_NC RSVDC16_NC	B8 B9 C10 C11 C12 C13 C14 C15 C16			RSVD* pins.
	RSVDC17_NC RSVDC18_NC RSVDC7_NC RSVDC8_NC RSVDC9_NC RSVDD10_NC RSVDD11_NC RSVDD12_NC RSVDD13_NC	C17 C18 C7 C8 C9 D10 D11 D12 D13			RSVD* pins.
	RSVDD14_NC RSVDD15_NC RSVDD16_NC RSVDD17_NC RSVDD18_NC RSVDD7_NC RSVDD8_NC RSVDD9_NC	D14 D15 D16 D17 D18 D7 D8 D9			RSVD* pins.
	RSVDE11_VSS	E11			RSVD* pin.
	RSVDE13_VSS RSVDE9_VSS	E13 E9			RSVD* pins.
	RSVDM1_NC RSVDM2_NC RSVDN20_NC RSVDN21_NC RSVDN4_NC	M1 M2 N20 N21 N4			RSVD* pins.
	RSVDN5_NC RSVDW20_NC RSVDW21_NC	N5 W20 W21			RSVD* pins.



Reserved	Signal	Ball #	Type	Reserved	Description
	RSVDY11_VSS RSVDY13_VSS RSVDY15_VSS RSVDY18_NC RSVDY17_NC RSVDH7_NC RSVDJ7_NC	Y11 Y13 Y15 Y18 Y17 H7 J7			RSVD* pins.
	NC_U7 NC_C19 NC_B19 NC_P4 NC_L4 NC_L2 NC_L1 NC_F20 NC_AC6 NC_J6 NC_L23 NC_L24 NC_N1 NC_N2 NC_T6 NC_T7 NC_AC19 NC_AB19 NC_AA19 NC_A20 NC_A21	U7 C19 B19 P4 L4 L2 L1 F20 AC6 J6 L23 L24 N1 N2 T6 T7 AC19 AB19 AA19 A20 A21			NC pins.
	RSVDE15_VSS RSVDY9_VSS RSVDF21_VSS	E15 Y9 F21			RSVD* pins.
	RSVDV16_VSS RSVDW16_VSS RSVDE17_VSS	V16 W16 E17			RSVD* pins.
	RSVDW7_VSS	W7			RSVD* pins.

Note: Connect RSVD pins based on naming convention:

Note: NC – pin is not connected in the package

Note: RSVD_NC – reserved pin. Should be left unconnected.

Note: RSVD_VSS – reserved pin. Should be connected to GND.

2.2.10 Integrated Voltage Regulator (SVR) pins

This section provides the pin assignment for SVR pins. The electrical specifications of the SVR pins are defined in Section 14.6.

Table 2-14. Integrated SVR pins

Reserved	Signal	Ball #	Type	Reserved	Description
	SVR_IND	AD17			Internal node of the integrated SVR. Connect to the analog power supply (VCCA) rail through an external inductor.
	VCC3P3_SVR	AD15	3.3V		Power supply input to integrated SVR (analog voltage).
	VSS_SVR	AD16	0V		Integrated SVR GND (analog voltage).
	VSS_S	V17, W17, W18	0V		Integrated SVR GND (analog voltage). Internal shield around the power switch.
	EXT_SVR_CONFIG	E12	Output	Digital	Configuration signal for the external SVR. Signal indicates what is the voltage level that the external SVR should provide (VCC-High or VCC-Low). 0b = External SVR should provide VCC-Low. 1b = External SVR should provide VCC-High. Default value = 1b. The value of this pin is define by the voltage-scaling-efuse.
	EXT_SVR_SENSE_P	M21	A-out	NA	Differential sense output for external SVR Dig.
	EXT_SVR_SENSE_N	M20	A-out	NA	Differential sense output for external SVR Dig.
	RSVDU16_VCCD	U16		NA	RSVD* pin.

2.2.11 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 14.6.

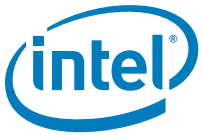


Table 2-15. Power supply pins

Reserved	Signal	Ball #	Type	Description
	VCC3P3	A10, A15, A19, AD10, AD6, A6	3.3V	Digital power supply.
	VCC3P3_A	AD19	3.3V	Analog power supply.
	VCC3P3_TX	E7, L5, P5, Y7	3.3V	Power supply for high speed SerDes transmit interfaces.
	VCCD	F11, F14, F16, F9, G11, G14, G16, G9, H11, H14, H16, H9, J11, J14, J16, K11, K12, K13, K14, K16, L11, L14, M11, M14, N11, N14, P11, P14, R11, R12, R13, R14, T11, T14, U11, U14, U9, V11, V14, V9, W11, W14, W9	0.85V	Digital power supply.
	VCCD_A	R16, T16	0.85V	Analog power supply.
	VCCA	H18, J18, J9, K18, K7, K9, L16, L18, L7, L9, M16, M18, M7, M9, N16, N18, N7, N9, P16, P18, P7, P9, R18, R7, R9, T18, T9, U18	0.935V	Analog power supply.
	VSSA	A1, A2, A22, A23, A24, A3, A5, AA1, AA2, AA22, AA23, AA24, AA3, AA6, AB20, AB21, AB22, AB3, AB4, AB5, AB6, AC1, AC2, AC22, AC23, AC24, AC3, AC5, AD1, AD2, AD22, AD23, AD24, AD3, AD5, B1, B2, B22, B23, B24, B3, B5, C20, C21, C22, C3, C4, C5, C6, D1, D2, D22, D23, D24, D3, D6, E19, E20, E21, E22, E3, E4, E5, E6, F1, F19, F2, F22, F23, F24, F3, F6, G18, G19, G20, G21, G22, G3, G4, G5, G6, G7, H1, H17, H19, H2, H22, H23, H24, H3, H6, H8, J17, J19, J20, J21, J22, J3, J4, J5, J8, K1, K17, K19, K2, K22, K23, K24, K3, K4, K5, K6, K8, L17, L19, L20, L21, L22, L3, L6, L8, M17, M19, M22, M23, M3, M4, M5, M6, M8, N17, N19, N22, N23, N3, N6, N8, P17, P19, P20, P21, P22, P3, P6, P8, R1, R17, R19, R2, R22, R23, R24, R3, R4, R5, R6, R8, T17, T19, T20, T21, T22, T3, T4, T5, T8, U1, U17, U19, U2, U22, U23, U24, U3, U6, U8, V18, V19, V20, V21, V22, V3, V4, V5, V6, V7, W1, W19, W2, W22, W23, W24, W3, W6, Y19, Y20, Y21, Y22, Y3, Y4, Y5, Y6	0V	Analog power supply ground.
	VSS	A16, AD9, F10, F12, F13, F15, F17, F18, F7, F8, G10, G12, G13, G15, G17, G8, H10, H12, H13, H15, J10, J12, J13, J15, K10, K15, L10, L12, L13, L15, M10, M12, M13, M15, N10, N12, N13, N15, P10, P12, P13, P15, R10, R15, T10, T12, T13, T15, U10, U12, U13, U15, V10, V12, V13, V15, V8, W10, W12, W13, W15, W8, A9	0V	Digital power supply ground.

2.2.12 Pull-up and pull-down resistors

Internal pull-up values:

- Min: 54 KΩ
- Typ: 74 KΩ
- Max: 110 KΩ

Table 2-16. Pull-up and pull-down resistors

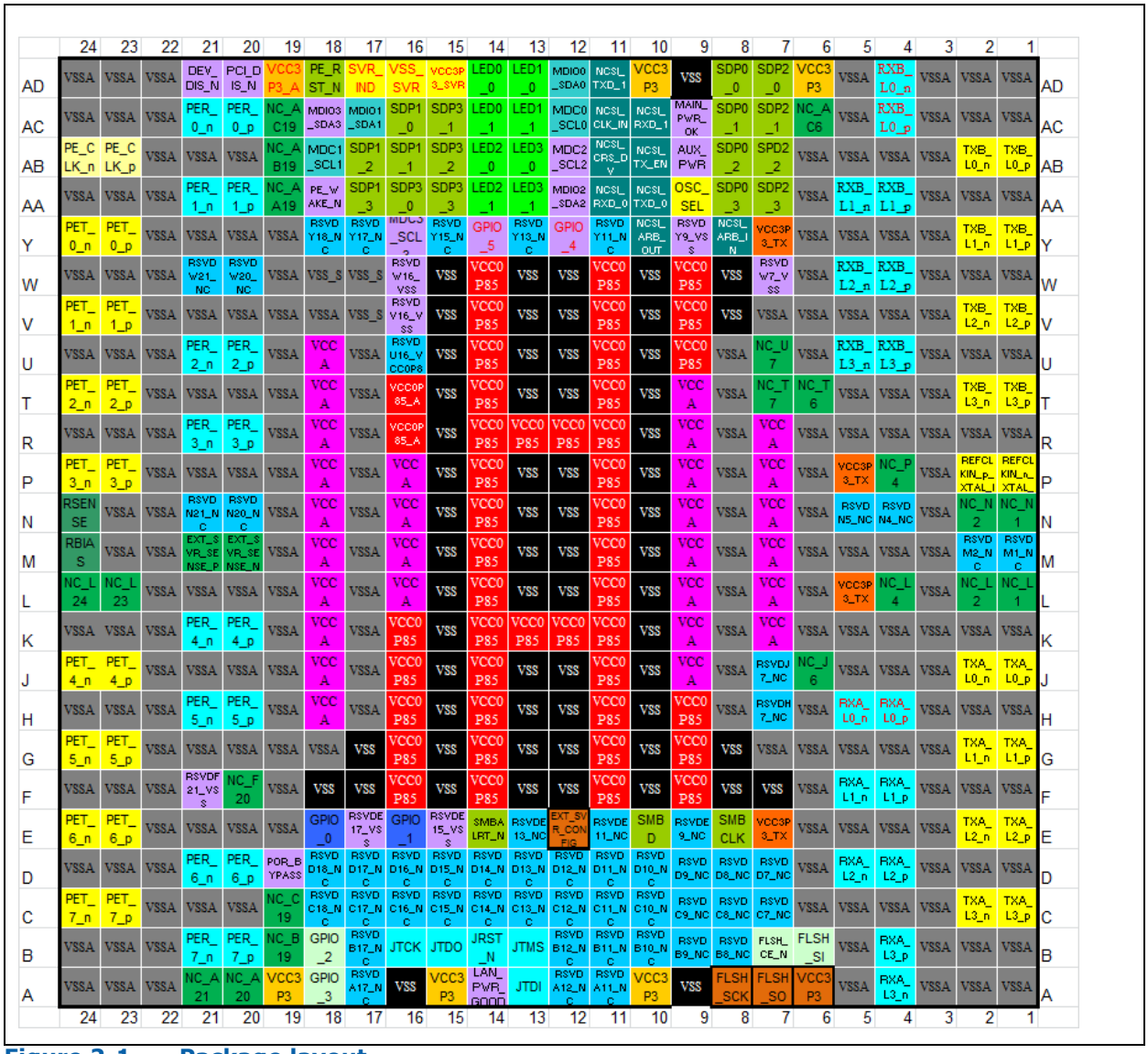
	Pin Name	Internal	External	Comments
	PE_CLK_p PE_CLK_n			
	PET_x_p PET_x_n			x=0,...,7.
	PE_WAKE_N		Pup	The pull-up resistor exists in the main platform as part of the system wake signal.
	PE_RST_N	Pup		
	REFCLKIN_p_XTAL_IN REFCLKIN_n_XTAL_OUT			
	RXx_Ly_p RXx_Ly_n			x=0,1. y=0,...,3.
	TXx_Ly_p TXx_Ly_n			x=0,1. y=0,...,3.
	MDIO0_SDAx		Pup 3.3 KΩ	x=0,...,3.
	MDC0_SCLx		Pup 3.3 KΩ	x=0,...,3.
	NCSI_CLK_IN	Pup	Pdn 100 KΩ	
	NCSI_CRS_DV	Pup	Pdn 100 KΩ	
	NCSI_RXD_0 NCSI_RXD_1		Pup 100 KΩ Pup 100 KΩ	
	NCSI_TX_EN	Pup	Pdn 100 KΩ	
	NCSI_TXD_0 NCSI_TXD_1	Pup Pup	Pup 100 KΩ Pup 100 KΩ	
	NCSI_ARB_IN	Pup	Pdn 10 KΩ	
	NCSI_ARB_OUT	Pup		
	SMBCLK		Pup 10 KΩ	
	SMBD		Pup 10 KΩ	
	SMBALRT_N		Pup 10 KΩ	
	FLSH_SI			
	FLSH_SO	Pup		
	FLSH_SCK			
	FLSH_CE_N		Pup 10 KΩ	
	LEDx_y			
	SDPx_y	Pup		
	LAN_PWR_GOOD	Pup	Pup 10 KΩ	
	POR_BYPASS		Pdn 100 Ω	
	OSC_SEL	Pup	See comment	Connect to VSS / VCC if using a crystal or oscillator, respectively.
	AUX_PWR	Pup	See comment	Connect the AUX_PWR signal to a 10 KΩ pull-up resistor if AUX power is available. Connect a pull-down resistor (0 Ω) if AUX power is not available.

	Pin Name	Internal	External	Comments
	MAIN_PWR_OK	Pup	See comment	Connect MAIN_PWR_OK signal to main power through a pull-up resistor. A pull-up value should be considered as 10 KΩ.
	PCI_DIS_N	Pup	Pup 10 KΩ	
	DEV_DIS_N	Pup	Pup 10 KΩ	
	RBIAS RSENSE			
	JTCK		Pdn 470 Ω	The external PD is used for 82599 compatibility.
	JTDI			The external PD is used for 82599 compatibility.
	JTDO		Pup 8.2 KΩ	The external PD is used for 82599 compatibility.
	JTMS			The external PD is used for 82599 compatibility.
	JRST_N		Pdn 10 KΩ	
	RSVDx_NC			x=0,...,31.
	RSVDE11_VSS			
	RSVDE13_VSS RSVDE9_VSS			
	RSVDM1_NC RSVDM2_NC RSVDN20_NC RSVDN21_NC RSVDN4_NC			
	RSVDN5_NC RSVDW20_NC RSVDW21_NC			
	RSVDY11_VSS RSVDY13_VSS RSVDY15_VSS RSVDY18_NC RSVDY17_NC RSVDH7_NC RSVDJ7_NC	Pup Pup Pup		
	NC			
	RSVDE15_VSS RSVDY9_VSS RSVDF21_VSS	Pup Pup Pup	Pdn 100 Ω Pdn 100 Ω Pdn 100 Ω	
	RSVDV16_VSS RSVDW16_VSS RSVDE17_VSS	Pup		
	RSVDW7_VSS			
	SVR_IND			
	VCC3P3_SVR			
	VSS_SVR			
	VSS_S			

	Pin Name	Internal	External	Comments
	EXT_SVR_CONFIG			
	EXT_SVR_SENSE_P			
	EXT_SVR_SENSE_N			
	RSVDU16_VCCA			

2.3 Package Layout

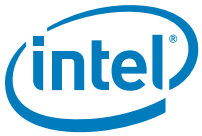
Figure 2-1 depicts a top view ball map of the XL710, in a 25 mm x 25 mm 576-pin Flip-Chip Ball Grid Array (FCBGA) package. See Section 14.7 for package mechanical specifications.



	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
AD	VSSA	VSSA	VSSA	DEV_DIS_N	PCI_DIS_N	VCC3P3_A	PE_RST_N	SVR_IND	VSS_SVR	VCC3P3_SVR	LED0_0	LED1_0	MDIO0_SDA0	NCSL_TXD_1	VCC3P3	VSS	SDP0_0	SDP2_0	VCC3P3	VSSA	RXB_L0_n	VSSA	VSSA	VSSA	AD
AC	VSSA	VSSA	VSSA	PER_0_n	PER_0_p	NC_AC19	MDIO3_SDA3	MDIO1_SDA1	SDP1_0	SDP3_1	LED0_1	LED1_1	MDC0_SCL0	NCSL_CLK_IN	NCSL_RXD_1	MAIN_PWR_OK	SDP0_1	SDP2_1	NC_AC6	VSSA	RXB_L0_p	VSSA	VSSA	VSSA	AC
AB	PE_CLK_n	PE_CLK_p	VSSA	VSSA	VSSA	NC_AB19	MDC1_SCL1	SDP1_2	SDP1_1	SDP3_2	LED2_0	LED3_0	MDC2_SCL2	NCSL_CRS_DV	NCSL_TX_EN	AUX_PWR	SDP0_2	SPD2_2	VSSA	VSSA	VSSA	VSSA	TXB_L0_n	TXB_L0_p	AB
AA	VSSA	VSSA	VSSA	PER_1_n	PER_1_p	NC_AA19	PE_WAKE_N	SDP1_3	SDP3_0	SDP3_3	LED2_1	LED3_1	MDIO2_SDA2	NCSL_RXD_0	NCSL_TXD_0	OSC_SEL	SDP0_3	SDP2_3	VSSA	RXB_L1_n	RXB_L1_p	VSSA	VSSA	VSSA	AA
Y	PET_0_n	PET_0_p	VSSA	VSSA	VSSA	VSSA	RSVDY18_NC	RSVDY17_NC	MDC3_SCL3	RSVDY15_NC	GPIO_5	RSVDY13_NC	GPIO_4	RSVDY11_NC	NCSL_ARB_OUT	RSVDY9_VSS	NCSL_ARB_IN	VCC3P3_TX	VSSA	VSSA	VSSA	VSSA	TXB_L1_n	TXB_L1_p	Y
W	VSSA	VSSA	VSSA	RSVDW21_NC	RSVDW20_NC	VSSA	VSS_S	VSS_S	RSVDW16_VSS	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCC0P85	VSS	RSVDW7_VSS	VSSA	RXB_L2_n	RXB_L2_p	VSSA	VSSA	VSSA	W
V	PET_1_n	PET_1_p	VSSA	VSSA	VSSA	VSSA	VSSA	VSS_S	RSVDV16_VSS	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCC0P85	VSS	VSSA	VSSA	VSSA	VSSA	VSSA	TXB_L2_n	TXB_L2_p	V
U	VSSA	VSSA	VSSA	PER_2_n	PER_2_p	VSSA	VCCA	VSSA	RSVDU16_VCC0P8	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCC0P85	VSSA	NC_U7	VSSA	RXB_L3_n	RXB_L3_p	VSSA	VSSA	VSSA	U
T	PET_2_n	PET_2_p	VSSA	VSSA	VSSA	VSSA	VCCA	VSSA	VCC0P85_A	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCCA	VSSA	NC_T7	NC_T6	VSSA	VSSA	VSSA	TXB_L3_n	TXB_L3_p	T
R	VSSA	VSSA	VSSA	PER_3_n	PER_3_p	VSSA	VCCA	VSSA	VCC0P85_A	VSS	VCC0P85	VCC0P85	VCC0P85	VCC0P85	VSS	VCCA	VSSA	VCCA	VSSA	VSSA	VSSA	VSSA	VSSA	VSSA	R
P	PET_3_n	PET_3_p	VSSA	VSSA	VSSA	VSSA	VCCA	VSSA	VCCA	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCCA	VSSA	VCCA	VSSA	VCC3P3_TX	NC_P4	VSSA	REFCLKIN_p_XTAL_I	REFCLKIN_n_XTAL_	P
N	RSENSE	VSSA	VSSA	RSVDN21_NC	RSVDN20_NC	VSSA	VCCA	VSSA	VCCA	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCCA	VSSA	VCCA	VSSA	RSVDN5_NC	RSVDN4_NC	VSSA	NC_N2	NC_N1	N
M	RBIAS	VSSA	VSSA	EXT_SVR_SENSE_P	EXT_SVR_SENSE_N	VSSA	VCCA	VSSA	VCCA	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCCA	VSSA	VCCA	VSSA	VSSA	VSSA	VSSA	RSVDM2_NC	RSVDM1_NC	M
L	NC_L24	NC_L23	VSSA	VSSA	VSSA	VSSA	VCCA	VSSA	VCCA	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCCA	VSSA	VCCA	VSSA	VCC3P3_TX	NC_L4	VSSA	NC_L2	NC_L1	L
K	VSSA	VSSA	VSSA	PER_4_n	PER_4_p	VSSA	VCCA	VSSA	VCC0P85	VSS	VCC0P85	VCC0P85	VCC0P85	VCC0P85	VSS	VCCA	VSSA	VCCA	VSSA	VSSA	VSSA	VSSA	VSSA	VSSA	K
J	PET_4_n	PET_4_p	VSSA	VSSA	VSSA	VSSA	VCCA	VSSA	VCC0P85	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCCA	VSSA	RSVDJ7_NC	NC_J6	VSSA	VSSA	VSSA	TXA_L0_n	TXA_L0_p	J
H	VSSA	VSSA	VSSA	PER_5_n	PER_5_p	VSSA	VCCA	VSSA	VCC0P85	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCC0P85	VSSA	RSVDH7_NC	VSSA	RXA_L0_n	RXA_L0_p	VSSA	VSSA	VSSA	H
G	PET_5_n	PET_5_p	VSSA	VSSA	VSSA	VSSA	VSSA	VSS	VCC0P85	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCC0P85	VSS	VSSA	VSSA	VSSA	VSSA	VSSA	TXA_L1_n	TXA_L1_p	G
F	VSSA	VSSA	VSSA	RSVDF21_VSS	NC_F20	VSSA	VSS	VSS	VCC0P85	VSS	VCC0P85	VSS	VSS	VCC0P85	VSS	VCC0P85	VSS	VSS	VSSA	RXA_L1_n	RXA_L1_p	VSSA	VSSA	VSSA	F
E	PET_6_n	PET_6_p	VSSA	VSSA	VSSA	VSSA	GPIO_0	RSVDE17_VSS	GPIO_1	RSVDE15_VSS	SMBALRT_N	RSVDE13_NC	EXT_SVR_CONFIG	RSVDE11_NC	SMBD	RSVDE9_NC	SMBCLK	VCC3P3_TX	VSSA	VSSA	VSSA	VSSA	TXA_L2_n	TXA_L2_p	E
D	VSSA	VSSA	VSSA	PER_6_n	PER_6_p	POR_BYPASS	RSVDD18_NC	RSVDD17_NC	RSVDD16_NC	RSVDD15_NC	RSVDD14_NC	RSVDD13_NC	RSVDD12_NC	RSVDD11_NC	RSVDD10_NC	RSVDD9_NC	RSVDD8_NC	RSVDD7_NC	VSSA	RXA_L2_n	RXA_L2_p	VSSA	VSSA	VSSA	D
C	PET_7_n	PET_7_p	VSSA	VSSA	VSSA	NC_C19	RSVDC18_NC	RSVDC17_NC	RSVDC16_NC	RSVDC15_NC	RSVDC14_NC	RSVDC13_NC	RSVDC12_NC	RSVDC11_NC	RSVDC10_NC	RSVDC9_NC	RSVDC8_NC	RSVDC7_NC	VSSA	VSSA	VSSA	VSSA	TXA_L3_n	TXA_L3_p	C
B	VSSA	VSSA	VSSA	PER_7_n	PER_7_p	NC_B19	GPIO_2	RSVDB17_NC	JTCK	JTDO	JRST_N	JTMS	RSVDB12_NC	RSVDB11_NC	RSVDB10_NC	RSVDB9_NC	RSVDB8_NC	FLSH_CE_N	FLSH_SI	VSSA	RXA_L3_p	VSSA	VSSA	VSSA	B
A	VSSA	VSSA	VSSA	NC_A21	NC_A20	VCC3P3	GPIO_3	RSVDA17_NC	VSS	VCC3P3	LAN_PWR_GOOD	JTDI	RSVDA12_NC	RSVDA11_NC	VCC3P3	VSS	FLSH_SCK	FLSH_SO	VCC3P3	VSSA	RXA_L3_n	VSSA	VSSA	VSSA	A
	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	

Figure 2-1. Package layout

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



3.0 Interconnects

3.1 PCI-Express* (PCIe*)

3.1.1 General

Note: The XL710 supports Rev. 3.0 of the PCIe base specification.

In addition to the capabilities required by the PCIe specifications, the XL710 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, and 8GT/s
  - Interface width of 1, 4, or 8 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 packet size: 2 KB, Maximum read request size: 4 KB
  - Extended tags
  - Function-Level Reset (FLR)
  - TLP Processing Hints (TPH)
- Reliability
  - Advanced Error Reporting (AER)
  - End-to-End CRC (ECRC) generation and checking
  - Recovery from data poisoning
  - Completion timeout
- Power management:
  - Active state power management (L0s and L1 states)
  - Wake capability

- DFT and DFM support for high-volume manufacturing
- The XL710 supports the following extended capabilities:
  - AER
  - Device Serial Number (DSN)
  - Alternative RID Interpretation (ARI)
  - Single Root I/O Virtualization (SR-IOV)
  - TPH Requester
  - Access Control Services (ACS)

3.1.2 Transaction layer

3.1.2.1 Transactions accepted by the XL710

Table 3-1 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 3-1. Transaction types accepted by the transaction layer

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

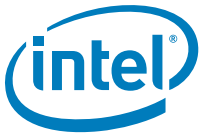
1. MCTP messages contain payload.

Flow Control Types Legend:

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

3.1.2.2 Size of target accesses

3.1.2.2.1 Memory accesses



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

- Write accesses:
  - 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 per PCIe rules
- Read accesses:
  - CSR reads are 32 bit or 64 bit only
    - 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 completer abort - the completion includes a CA status and an error is generated per PCIe rules

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

- Write accesses:
  - Writes to Flash are 8-bit wide only
  - Any larger write accesses are handled as completer abort - data is dropped and an error is generated per PCIe rules
- Read accesses:
  - Reads to Flash are 32-bit wide
  - Partial reads with at least one byte disabled are handled internally as a full read. That is, any side effect of the full read (such as clear by read) is also applicable to partial reads. The completion on PCIe adheres to the specification rules regarding the number of bytes reported in the completion
  - Zero-length reads generate a completion, but the Flash is not accessed and undefined data is returned
  - Larger CSR read requests are handled as completer abort - the completion includes a CA status and an error is generated per PCIe rules

3.1.2.3 I/O accesses

Rules for accesses to the I/O BAR:

- Write accesses:
  - Write accesses are 32-bit wide
  - Zero-length writes have no internal impact (nothing written, no effect such as clear-by-write). The transaction is treated as a successful operation (no error event)
  - Other accesses (partial writes, larger writes) are handled as completer abort - data is dropped and an error is generated per PCIe rules
- Read accesses:

- — Reads to the I/O BAR are 32-bit wide
- — Partial reads with at least one byte disabled are handled internally as a full read. That is, any side effect of the full read (such as clear by read) is also applicable to partial reads. The completion on PCIe 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 per PCIe rules
- — See Section 11.1.1.5 for more details.

3.1.2.3.1 Messages

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

3.1.2.3.2 Support for dynamic changes

The XL710 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.

3.1.2.4 Transactions initiated by the XL710

Table 3-2 lists the transactions initiated by the device and their attributes.

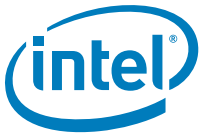
Table 3-2. Transaction types initiated by the transaction layer

Transaction type	Payload Size	FC Type
Configuration Read Request Completion	Dword	CPLH + CPLD
Configuration Write Request Completion	-	CPLH
IO Read Request Completion	Dword	CPLH + CPLD
IO Write Request Completion	-	CPLH
Read Request Completion	Dword/Qword	CPLH + CPLD
Memory Read Request	-	NPH
Memory Write Request	≤ MAX_PAYLOAD_SIZE	PH + PD
Message	≤ 64 bytes[1]	PH + 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 2 KB.
  - — 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 XL710 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.

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. The XL710 therefore 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 XL710 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, then it is not broken at the cache line boundary but rather sent as a single packet. However, if any of the general rules require that the request is broken into two or more packets, then 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), then the request is broken into multiple TLPs staring 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 240bytes of payload (since 240bytes + 0x10 = 256bytes 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 3-3 lists the response to messages sent to the device. Unlisted messages are not supported by the device and are treated as an unsupported request.

Table 3-3. Messages in the XL710 (as a receiver)

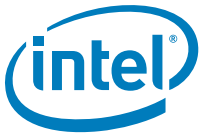
Message Code [7:0]	Routing r2r1r0	Message	XL710 Response
0x00	011b	Unlock	Silently drop.
0x14	100b	PM_Active_State_NAK	Accepted.
0x19	011b	PME_Turn_Off	Accepted.
0x40 0x41 0x43 0x44 0x45 0x47 0x48	100b	Ignored messages (used to be hot-plug messages)	Silently drop.
0x50	100b	Slot power limit support (has one Dword data)	Silently drop.
0x7E	000b 010b 011b 100b	Vendor_defined type 0	Drop and handle as an unsupported request.
0x7F	100b	Vendor_defined type 1	Silently drop.
0x7F	000b 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 3-4 lists the messages sent by the device.

Table 3-4. Messages in the XL710 (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 10.7.3.1 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 XL710 only supports TC = 0b and VC = 0b (default).

3.1.2.6.2 TLP Processing Hints (TPH)

The XL710 supports the TPH capability defined in the PCI Express specification. It does not support extended TPH requests.

Existence of a TLP Process Hint (TPH) is indicated on the PCIe link by setting the TH bit in the TLP header. Using the PCIe TLP Steering Tag (ST) and Processing Hints (PH) fields, the XL710 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 XL710 exposes a PCIe TPH capability structure (see Section 12.4.5) with no steering table.

Requiredsteps 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 (PH) fields.
4. Steering information should be programed in the CPUID fields in the receive or transmit queue context.

The Processing Hints (PH) and Steering Tags (ST) are set according to the characteristics of the traffic as listed in Table 8-4.

Note: In order 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 8.2.4.
- The policy for FCoE traffic is described in Section 9.3.3.
- Accesses to the host memory cache do not use TPH hints.
- Accesses to the admin command queues do not use TPH hints.

3.1.2.7 Device ordering rules

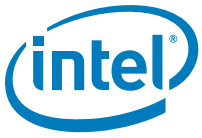
The XL710 meets the PCIe ordering rules as follows:

Deadlock avoidance – The XL710 meets the PCIe ordering rules that prevent deadlocks:

1. 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 in order to allow later posted writes to proceed.
2. Target posted writes overtake stalled target configuration writes.
3. 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 XL710 are allowed to proceed.

Consistency of data:

1. Descriptor/Data Ordering — The XL710 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.
2. Target NP read requests might pass target posted writes addressing different PCI functions.
3. 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. A common mechanism is to issue a read (like the Transactions Pending bit in the Device Status register) to that function(s). Once the completion is received by host software, it insures that any previous posted writes have been 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 doesn't know when the tables are actually updated in the XL710, 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:

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

3.1.2.7.1 Processing of target accesses

The XL710 meets the specification requirements regarding target accesses as described in the previous section.

In addition, the following behaviors apply:

- Target accesses from different functions might be processed in a different order than the order 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 XL710 takes advantage of the relaxed ordering rules in PCIe. By setting the relaxed ordering bit in the packet header, the XL710 enables the system to optimize performance in the following cases:

1. Relaxed ordering for LAN and FCoE descriptor and data reads — When the XL710 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.RXDESCRDROEN* bit (loaded from NVM) enables relaxed ordering for Rx descriptor reads.
   — The *GLLAN_TCTL.TXDESCRDROEN* bit (loaded from NVM) enables relaxed ordering for Tx descriptor reads.
   — The *GLLAN_TCTL.TXDATARDROEN* bit (loaded from NVM) enables relaxed ordering for Tx data reads.
2. Relaxed ordering for LAN and FCoE Rx data writes — When the XL710 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.RXDATAWRROEN* bit (loaded from NVM) enables relaxed ordering for Rx data writes.
3. The XL710 does not relax ordering for the following requests:
   — LAN and FCoE descriptor writes
   — LAN Tx head write back
   — Interrupt messages
   — MCTP messages
   — HMC requests
   — EMP requests
   — Any other requests not previously mentioned

Relaxed ordering is globally enabled in the XL710 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.

3.1.2.7.3 ID-based ordering (IDO)

ID-based ordering was introduced in the PCIe rev. 2.1 specification. When enabled, the XL710 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.

3.1.2.8 Flow control

3.1.2.8.1 Flow control rules

The XL710 only implements the default Virtual Channel (VC0). A single set of credits is maintained for VC0.

Table 3-5. Flow control credits allocation

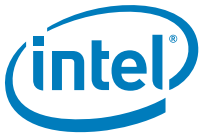
Credit Type	Operations	Number of Credits (per device)
Posted Request Header (PH)	Target write (one unit) Message (one unit)	**96** header credit units.
Posted Request Data (PD)	Target write Message	288 data credits units.
Non-Posted Request Header (NPH)	Target read (one unit) Configuration read (one 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 XL710 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 XL710 implements the optional flow control update timeout mechanism. See the PCIe specification.



3.1.2.9 End-to-End CRC (ECRC)

The XL710 supports ECRC as defined in the PCIe specification. The following functionality is provided:

- Inserting ECRC in transmitted TLPs:
  - The XL710 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 XL710 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 XL710 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 XL710 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 signalled 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

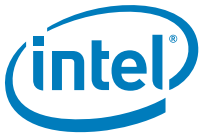
The XL710 supports Gen 1 (2.5GT/s), Gen 2 (5GT/s), and Gen 3 (8GT/s).

The following configuration controls link speed:

- PCIe *Supported Link Speeds* bit — Indicates the link speeds supported by the XL710.
- 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 XL710 does not initiate a hardware autonomous speed change.

The XL710 supports entering compliance mode at the speed indicated in the *Target Link Speed* field in the PCIe Link Control 2 register. Compliance mode functionality is controlled via the PCIe Link Control 2 register.



3.1.4.2 Link width

The XL710 supports a maximum link width of 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. Hardware default is the x8 link.

During link configuration, the platform and the XL710 negotiate on a common link width. The link width must be one of the supported PCIe link widths (x1, x4, x8), such that:

- If maximum link width = x8, then the XL710 negotiates to either x8, x4, or x1
- If maximum link width = x4, then the XL710 negotiates to either x4 or x1
- If maximum link width = x1, then the XL710 only negotiates to x1

The XL710 does not initiate a hardware autonomous link width change.

3.1.4.3 Lane configurations

The XL710 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 XL710 does not initiate lane reversal. The Link Partner (LP) might initiate lane reversal (in order to end up with an optimal lane width) and The XL710 consents with the lane reversal.
- If lane 0 is found invalid, The XL710 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 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, then the respective configuration is not provided on that system.

Figure 3-1, Figure 3-2, and Figure 3-3 depict the initial link width configuration and link degradation options. In Figure 3-1 and Figure 3-2, the upper part of the figures describe link options where the LP and the XL710are aligned. The bottom part of the figures describe link options where the LP and the XL710 are reversed in order.

- Figure 3-1 applies when either the LP or the XL710 is physically set to x8.
- Figure 3-2 applies when either the LP or the XL710 is physically set to x4 and both are not physically set x8.
- Figure 3-3 applies when both the LP or the XL710 is physically set to x1.

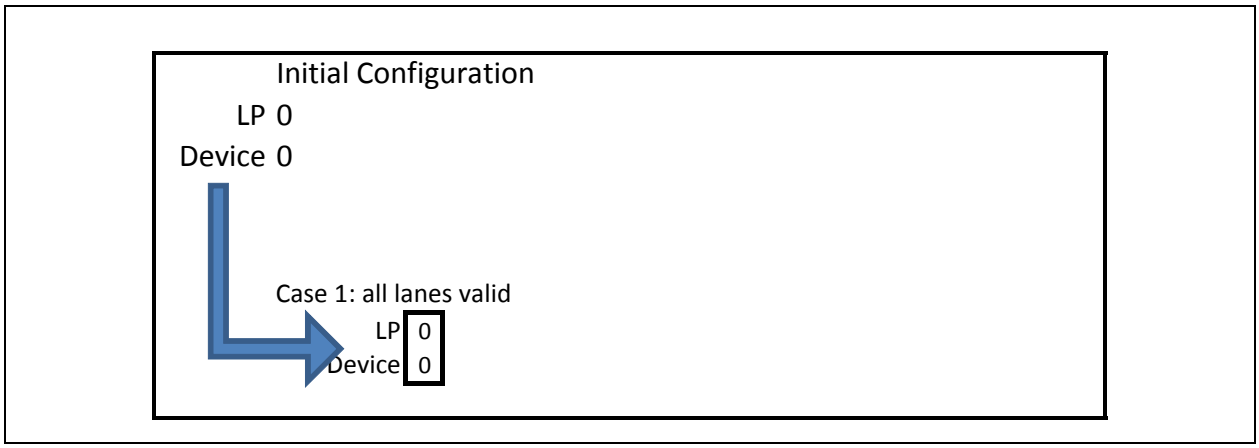


Figure 3-1. Link width configurations for a x1 port

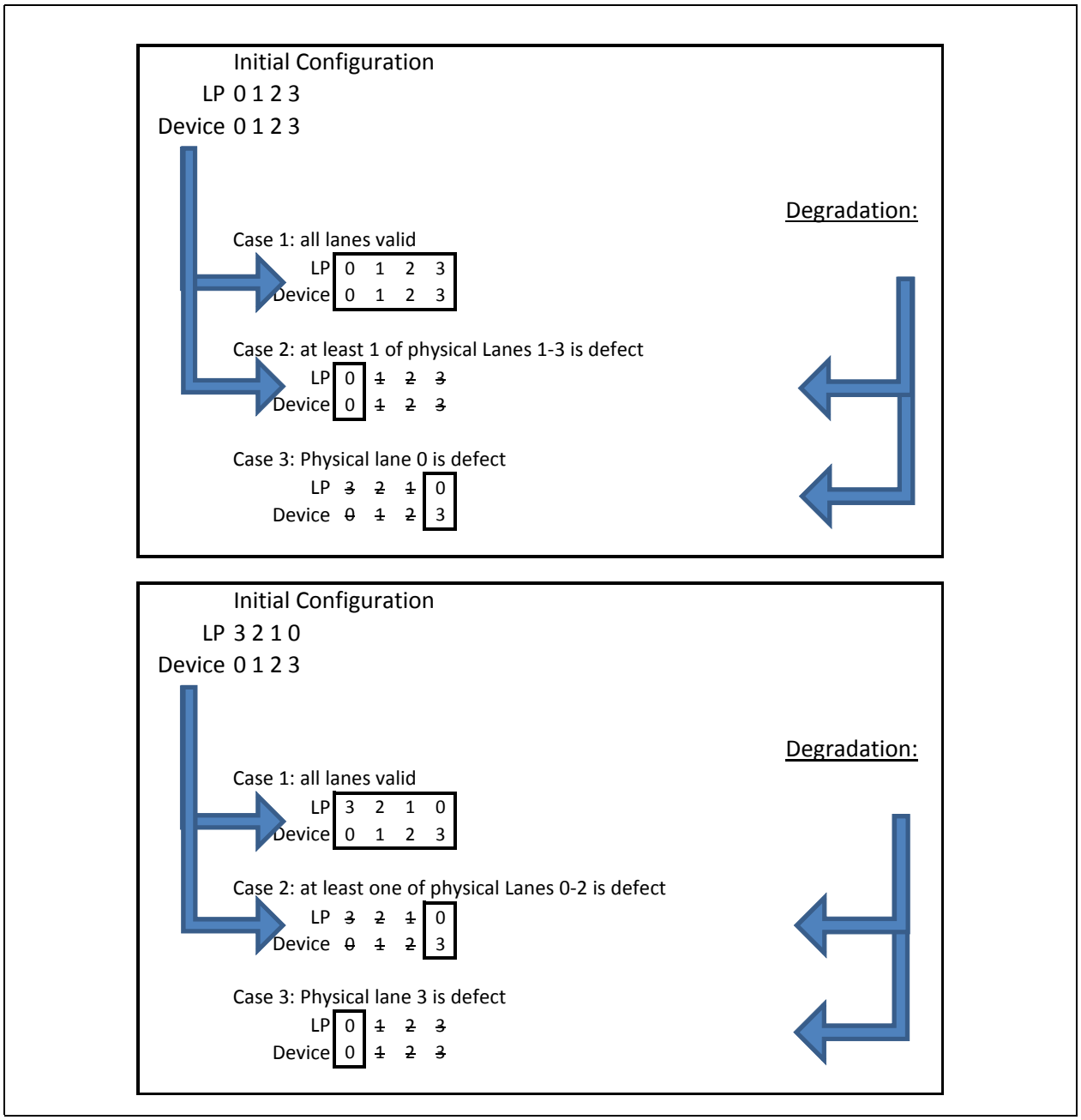


Figure 3-2. Link width configurations for a x4 port

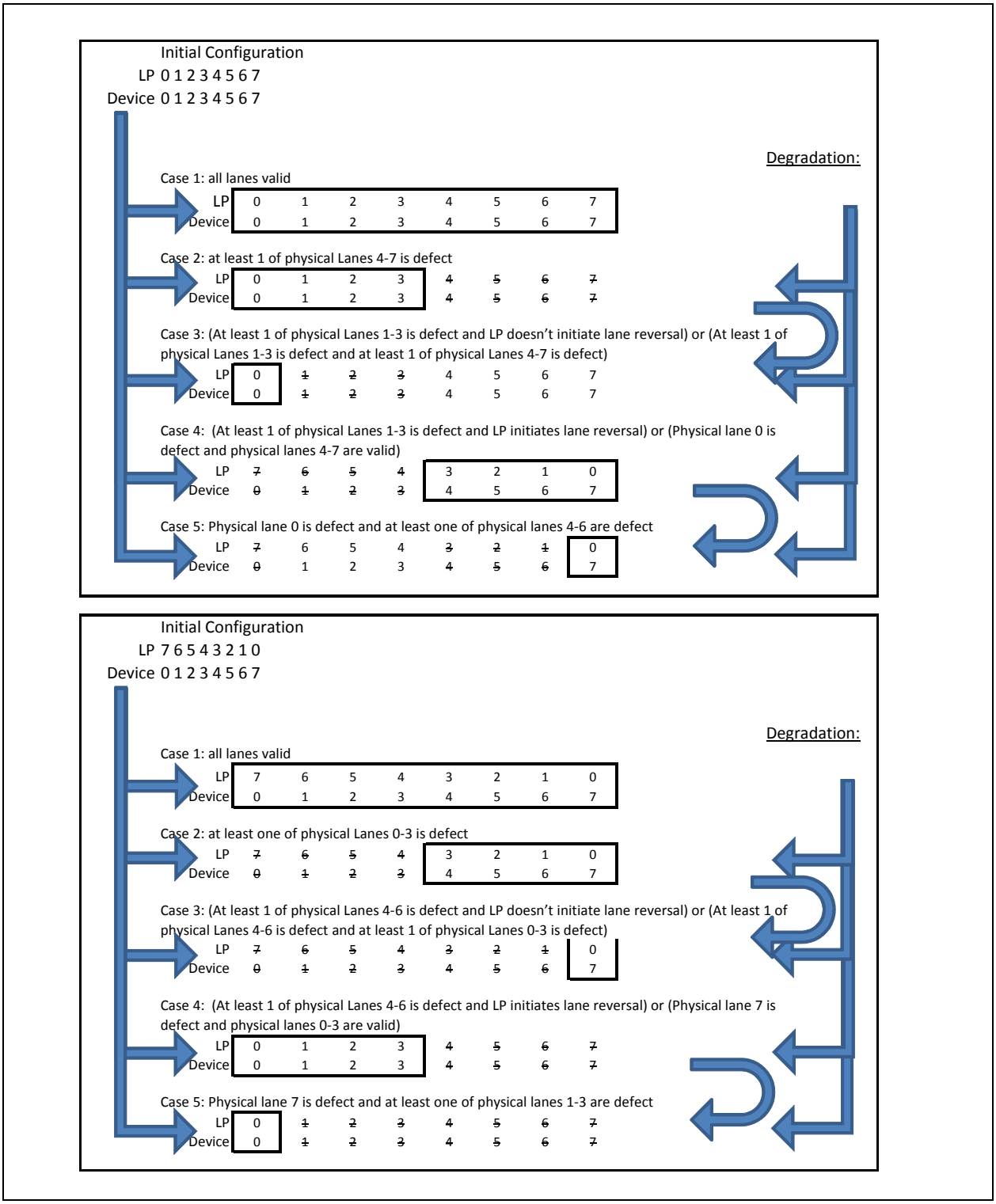
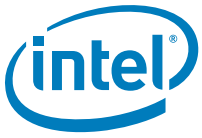


Figure 3-3. Link width configurations for a x8 port



3.1.4.4 Receiver framing requirements

This section applies to Gen 3 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

3.1.5.1 General description

PCIe defines two error reporting paradigms: the baseline capability and the 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 XL710, but the AER capability needs to 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 3-4 shows, in detail, the flow of error reporting in PCIe. See also Figure 6-2 in the PCIe base specification.

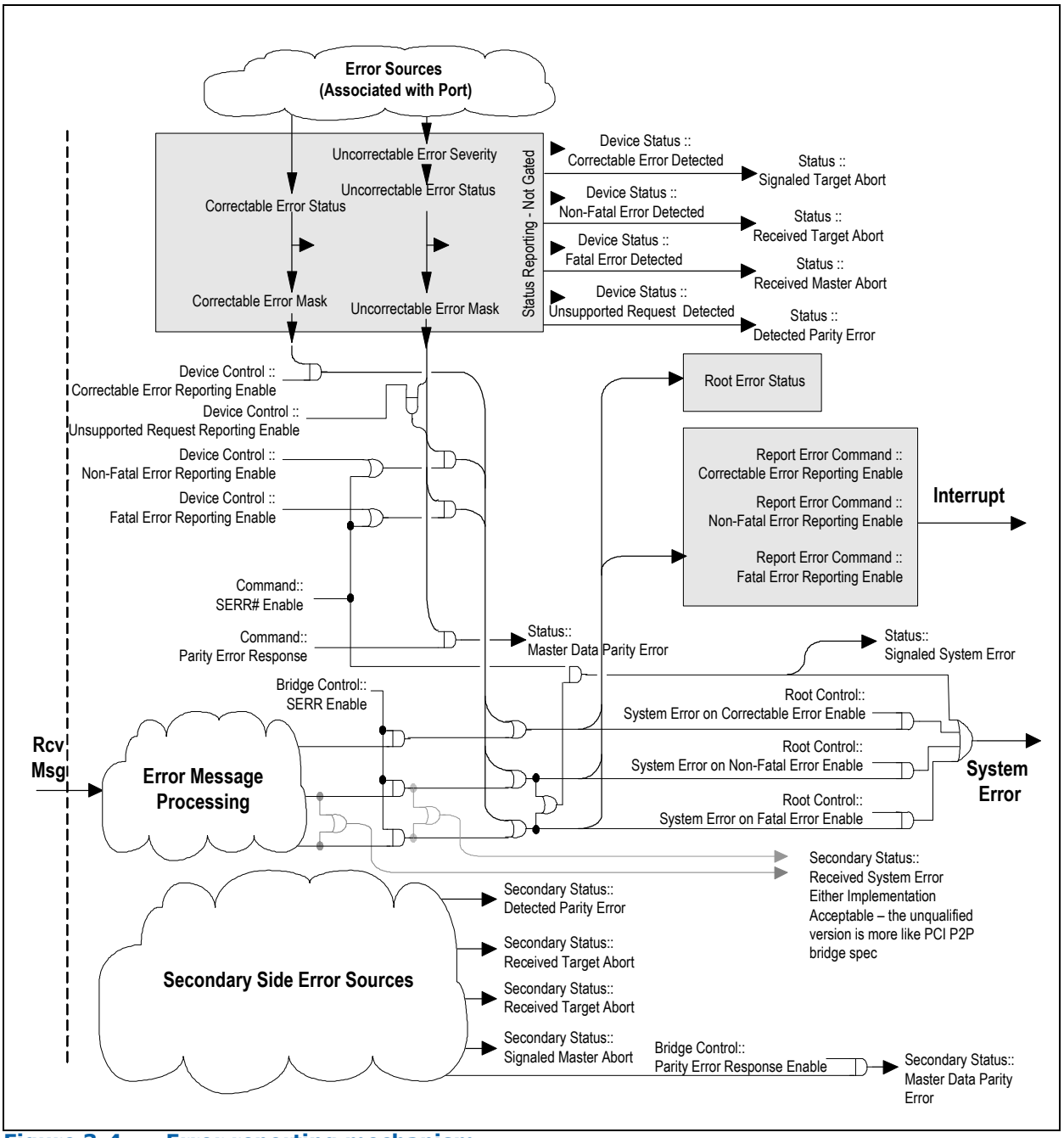


Figure 3-4. Error reporting mechanism

3.1.5.2 Error events

Table 3-6 lists the error events identified by the XL710 and the response in terms of logging, reporting, and actions taken. Refer to the PCIe specification for the effect on the PCI Status register.

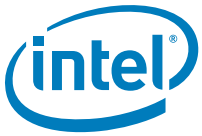


Table 3-6. Response and reporting of PCIe error events

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	
TLP Errors			
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 Report error to device driver per Section 3.1.5.8
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 Report error to device driver per Section 3.1.5.8
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	Uncorrectable ERR_NONFATAL Log header	Send Completion With UR f 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 Report error to device driver per Section 3.1.5.8
Completion Timeout	Completion Timeout Timer Expired	Uncorrectable ERR_NONFATAL	See Section 3.1.5.3 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

Table 3-6. Response and reporting of PCIe error events

Error Name	Error Events	Default Severity	Action
Completer Abort	Received Target Access with illegal data size per Section 3.1.2.2	Uncorrectable. ERR_NONFATAL Log header	Send completion with CA 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 Report error to device driver per Section 3.1.5.8
Unexpected Completion	Received Completion Without a Request For It (Tag, ID, etc.)	Uncorrectable ERR_NONFATAL Log Header	Discard TLP If error is defined as non-fatal (default severity): • Treat as an advisory non-fatal error: Send an ERR_COR Message If error is defined as fatal: • Send ERR_FATAL message
Receiver Overflow	Received TLP Beyond Allocated Credits	Uncorrectable ERR_FATAL	Receiver Behavior is Undefined
Flow Control Protocol Error	Minimum Initial Flow Control Advertisements Flow Control Update for Infinite Credit Advertisement	Uncorrectable. ERR_FATAL	Receiver Behavior is Undefined
Malformed TLP (MP)	Data Payload Exceed 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 Don't Use TC0. Usage of Unsupported VC Target request crosses a 4KB boundary	Uncorrectable ERR_FATAL Log Header	Drop the Packet, Free FC Credits
Completion with Unsuccessful Completion Status		No Action (already done by originator of completion)	Free FC Credits

3.1.5.3 Completion timeout mechanism

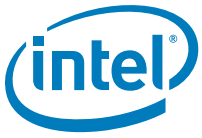
The XL710 supports completion timeout as defined in the PCIe specification.

The XL710 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 is hardwired to 1b to indicate that disabling completion timeout is supported
  - The PCIe *Completion Timeout Disable* bit in Device Control 2 register controls whether completion timeout is enabled
- A programmable range of timeout values
  - The XL710 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:

- The appropriate message is sent on PCIe as listed in Table 3-6.



- The affected queue or client takes action based on the nature of the original request. An interrupt is issued to the respective PF.
- Software might identify the source of the event (whether due to TLP poisoning, to a completion timeout, or an actual malicious event) by reading the error reporting counters or the performance and 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:

- The appropriate message is sent on PCIe as listed in Table 3-6.
- An interrupt is issued as described in Section 3.1.5.8.
- 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, etc.

3.1.5.5 Completion with unsuccessful completion status

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

3.1.5.6 Error pollution

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

3.1.5.7 Blocking on upper address

The *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 (for example, a packet is sent unconditionally).

Processing a blocked transaction:

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

3.1.5.8 Proprietary error reporting

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

Several CSRs are dedicated to this functionality, with a separate bit allocated per error type (see Table 3-7):

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

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

- The device checks if the respective bit is set in the PCIe Errors Reported register. If cleared, done. Else, continue.
- The respective bit is set in the PCIe Interrupt Cause register.
- If the respective bit is set in the PCIe Interrupt Enable register, an interrupt is issued to the PCI function. The *PCI_EXCEPTION* cause is used (see the PF Interrupt Zero Cause - PFINT_ICR0 register).

Table 3-7. PCIe errors reported to device software

Error Event	Index	Description and Comments	Function Association[1]
Exceeded upper address limit (read requests)	00	See Section 3.1.5.7	Sent to PF
Exceeded upper address limit (write requests)	01	See Section 3.1.5.7	Sent to PF
Reserved	02	Reserved entries	N/A
Poisoned TLP received	03	See Section 3.1.5.4	Sent to PF
Reserved	04-05	Reserved entries	N/A
ECRC error detected	06	ECRC check failed on a received TLP. See Section 3.1.2.9	Sent to all PFs

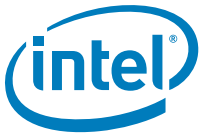


Table 3-7. PCIe errors reported to device software

Error Event	Index	Description and Comments	Function Association[1]
Unsupported Request - Request Type	07	Request causes an Unsupported Request due to receipt of TLP with unsupported Request Type	Sent to all PFs
Unsupported Request - Vendor Message	08	Request causes an Unsupported Request due to receipt of an Unsupported Vendor Defined Type 0 Message	Sent to PF
Unsupported Request - Message Code	09	Request causes an Unsupported Request due to receipt of an invalid Message Code	Sent to PF
Unsupported Request - Function Number	10	Request causes an Unsupported Request due to receipt of a not-supported Function Number	Sent to all PFs
Unsupported Request - Address Range	11	Request causes an Unsupported Request due to receipt of a not-supported Address Range	Sent to all PFs
Completer abort - target size	13	Received Target Access with illegal data size per Section 3.1.2.2 (CA)	Sent to PF
Reserved	14 - 31	Reserved entries	N/A

1. Error detected in a VF is reported to its PF.

3.1.6 Performance and statistics counters

The XL710 incorporates counters to track the behavior and performance of the PCIe interconnect. The XL710 implements several types of counters:

- 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. Registers *GLPCI_GSCN_0_3[2]* and *GLPCI_GSCN_0_3[3]* form the second 64-bit counter.
  - The enable bits for the two 64-bit counters are *GLPCI_GSCL_1.GIO_COUNT_ EN_0* and *GLPCI_GSCL_1.GIO_COUNT_ EN_2*, respectively.
- A single counter that measures latency - the time it takes an event to complete.
- Two counters that measures bandwidth, counting both TLPs and bytes.

General characteristics of the counters:

- Software can reset, stop, or start the counters (all at the same time).
- The counters are shared by all PCI functions (service mode of sharing).

3.1.6.1 Event counters

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 3.1.6.2 for more details.

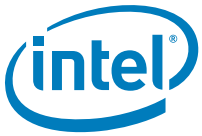
The list of events supported by the XL710 are listed in Table .

Table 3-8. PCIe* statistic events encoding

Events	Event Mapping (Hex)	Description
Cycles	0x00	Increment on each PCIe clock tick.
Transaction Layer events		
Bad Request TLPs	0x10	Number of bad TLPs arriving to the transaction layer.These include: • Request caused UR. • Request caused CA. • Malformed TLP.
Bad Completions	0x11	Number of bad completions received. These include: • Unexpected completion. • UR status. • CA status.
Completion Timeout	0x12	Number of completion timeout events.
Poisoned TLP	0x13	Number of TLPs received with poisoned data.
ECRC Check	0x14	Number of TLPs that foil ECRC check.
Link Layer events		
Retry Buffer Resent	0x30	Increment when a TLP is resent from the retry buffer (for any reason).
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.
NACK DLLP Received	0x33	Increment every time a NACK DLLP is received by the link layer.
No PH Credits	0x34	Increment every time the number of PH credits available from the link partner reaches zero.
No PD Credits	0x35	Increment every time the number of PD credits available from the link partner reaches zero.
No NPH Credits	0x36	Increment every time the number of NPH credits available from the link partner reaches zero.
Physical Layer events		
Receive Error	0x50	Increment when one of the following occurs: 1. Decoder error occurred during training in the PHY. It is reported only when training ends. 2. Decoder error occurred during link-up or till the end of the current packet (in case the link failed). This error is masked when entering/ exiting EI.
LTSSM in L0s in both Rx & Tx	0x52	Increment when LTSSM enters L0s state in either Tx or Rx and the other direction is already in L0s.
LTSSM in L0s in Rx	0x53	Increment when LTSSM enters L0s state in Rx.
LTSSM in L0s in Tx	0x54	Increment when LTSSM enters L0s state in Tx.
LTSSM in L1 ASPM	0x55	Increment when LTSSM enters L1 ASPM (requested from host side).
LTSSM in recovery	0x56	Increment when LTSSM enters recovery state.

3.1.6.2 Leaky bucket mode

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



- One of four 16-bit Leaky Bucket Counters (LBC) is enabled via the *LBC_ENABLE_[3:0]* bits in the PCIe Statistic Control register #1.
- The LBC is controlled by the *GIO_COUNT_START, GIO_COUNT_STOP, GIO_COUNT_RESET* bits in the PCIe Statistic Control register #1.
- The LBC increments every time the respective event occurs.
- The LBC is decremented every T μs as defined in the *LBC_TIMER_N* field in the PCIe Statistic Control registers #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 registers #5...#8 (*GLPCI_GSCL_5_8*), the respective statistics counter increments, and the LBC counter is cleared to zero.

3.1.6.3 Latency counter

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

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

The latency counter is controlled by the following CSR fields:

- *PCI_COUNT_LAT_EN* in the *GLPCI_GSCL_1* register enables the latency counter.
- *PCI_COUNT_LAT_CT* is a single 32-bit counter that counts time in number of internal clocks. The duration of a single clock cycle is a function of link speed and width, and is listed in Table 3-9.
- Counting is started, stopped, reset through the PCIe Statistic Control register #1.
- *PCI_COUNT_LAT_EV* in the *GLPCI_GSCL_1* register selects the event to be measured according to the list in Table 3-10.

Table 3-9. Resolution of the PCI_COUNT_LAT_CT counter

PCIe Operation Speed	Setting of the GLPCI_CLKCTL.PCI_CLK_DYN Bit	PCIe Operational Link Width	Cycle Duration (ns)
Gen 1 (2.5G)	0b	x	8
Gen 2 (5.0G)	0b	x	4
Gen 3 (8.0G)	0b	x	2
Gen 1 (2.5G)	1b	8 lanes	16
Gen 2 (5.0G)	1b	8 lanes	8
Gen 3 (8.0G)	1b	8 lanes	4
Gen 1 (2.5G)	1b	1 or 4 lanes	32
Gen 2 (5.0G)	1b	1 or 4 lanes	16
Gen 3 (8.0G)	1b	1 or 4 lanes	8

Table 3-10. PCIe latency counter events

Events	Event Mapping (Hex)	Description
Max Read Latency	0x00	Captures the maximum read latency experienced since the last counter reset. Latency is measured from time the read request starts until the time the completion starts to arrive.
Min Read Latency	0x01	Captures the minimal read latency experienced since the last counter reset. Latency is measured from time the read request starts until the time the completion starts to arrive.
Avg Read Latency	0x02	Captures the average read latency experienced since the last counter reset. Latency is measured from time the read request starts until the time the completion starts to arrive.

3.1.6.4 Bandwidth counter

The bandwidth counters are controlled by the following CSRs:

- *PCI_COUNT_BW_EN* in the *GLPCI_GSCL_1* register enables the bandwidth counter.
- *GLPCI_PKTCT.PCI_COUNT_BW_PCT* is a 32-bit counter that counts packets.
- *GLPCI_BYTCTH.PCI_COUNT_BW_BCT* is a 64-bit counter that counts bytes.
- Counting is started, stopped, reset through the PCIe Statistic Control register #1.
- *GLPCI_GSCL_1.PCI_COUNT_BW_EV* - selects the event to be measured according to the list in Table 3-11.

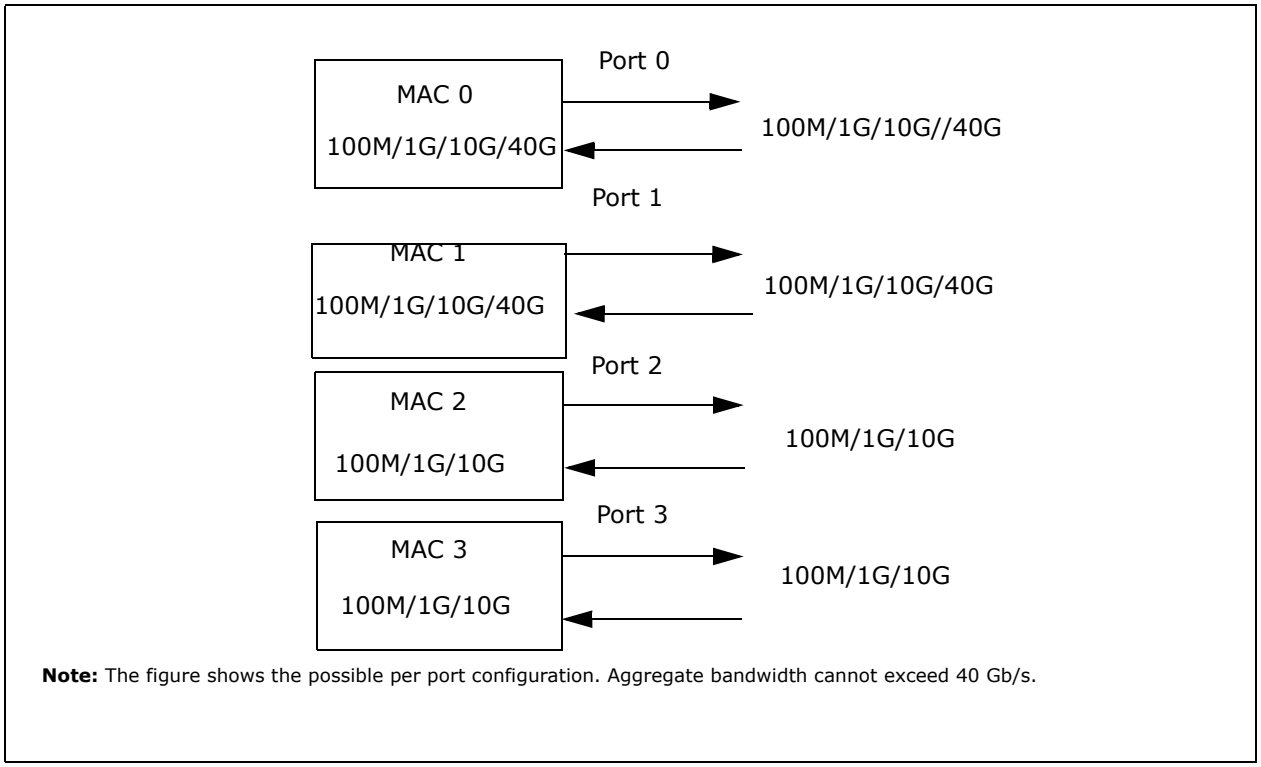
Table 3-11. PCIe bandwidth counter events

Events	Event Mapping (Hex)	Description
Posted Write bandwidth	0x00	Counts the number of TLPs and bytes issued using posted transactions. Bytes count includes payload bytes and does include any PCIe overhead.
Non Posted bandwidth	0x01	Counts the number of TLPs and bytes issued using non-posted transactions. Only TLPs are counted. Bytes count is undefined.
Completion bandwidth	0x02	Counts the number of TLPs and bytes issued using completions. Bytes count includes payload bytes and does include any PCIe overhead.

3.2 Ethernet interconnect

3.2.1 Media Access Control (MAC) layer

The XL710 supports up to 4 full duplex Ethernet MAC ports compliant with IEEE Std802.3-2008 and IEEE Std 802.3ba-2010 standards (Clause 4 and Annex 4A). The MAC ports can be configured to operate at different speeds of operation, not to exceed an aggregate bandwidth of 40 Gb/s, as shown in Figure 3-5.



Note: The figure shows the possible per port configuration. Aggregate bandwidth cannot exceed 40 Gb/s.

Figure 3-5. XL710 MAC configuration

Each MAC port is associated with a corresponding Media Access Unit (MAU) that provides the physical layer interfaces as explained in Section 3.2.1. The MAUs need to be configured to operate with appropriate physical layer protocols based on the MAC operating speed. Physical layers supported by the XL710 for different speeds of operation are explained in Section 3.2.2.

3.2.1.1 MAC speed configuration

Table 3-12 lists the possible speed configurations available on each MAC port.

Table 3-12. MAC port and possible speed configurations

Port		1 Gb/s	10 Gb/s	40 Gb/s
MAC 0		Y	Y	Y
MAC 1		Y	Y	Y
MAC 2		Y	Y	N
MAC 3		Y	Y	N

- When MAC port 0 or 1 or both are configured to operate at up to 40 Gb/s, ports 2 and 3 should be disabled.
- When MAC port 0 is configured to operate at up to 40 Gb/s, ports 1 through 3 should be disabled.
- All 4 ports can be enabled when MAC ports 0, 1, 2 and 3 are configured for operation at up to 10 Gb/s or below.

When LAN ports are disabled in multi-port system configurations, corresponding PCIe functions need to be disabled through the NVM or external disable pins. See Section 4.2.3 for details on PCI function disable and LAN port disable functionality.

The XL710 supports auto-negotiation protocol when configured for certain physical interfaces. The link speed on each port can either be configured through Link Configuration Admin commands (see Section 3.2.4) or auto selected when auto-negotiation is enabled on the respective ports.

3.2.1.2 Ethernet CRC generation and stripping

The XL710 MAC supports CRC generation (Ethernet frame check sequence). The MACs can be independently programmed to generate and append the 32-bit CRC (FCS) in the transmit direction by setting the *CRC Enable* field in the Set MAC config AQ command.

The default value is set for the MAC to append the CRC.

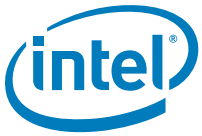
Note: When CRC generation is disabled in the MAC, automatic padding in the MAC does not work correctly, therefore all transmitted packets must be at least 64 bytes long.

3.2.1.3 Transmit padding

The minimum frame size for Ethernet as specified by IEEE Std 802.3 standard is 64 bytes. The XL710 MAC pads, with ZEROs, Ethernet packets that are smaller than 64 bytes during transmit. Refer to the padding rules for received packets and loop-back packets in Section 7.2.2.

3.2.1.4 Jumbo frame support

The XL710 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 Set MAC config <Max Frame Size > field.



3.2.1.5 Ethernet Flow Control (FC)

The XL710 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 XL710 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: A XL710 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 7.7.1.1.5, Section 7.7.1.2.10, and Section 7.7.2.2 for the other programming related to flow control. In 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.

At 10 Gb/s and lower speeds:

- MAC Flow Control (PRTDCB_MFLCN) register — Enables flow control and passing of MAC control packets to the host.
- Flow Control Configuration (PRTDCB_FCCFG) — Determines mode for Tx flow control (no FC, LFC, or versus PFC).
- Flow Control Transmit Timer Value (PRTDCB_FCTTVN[3:0]) — a set of 16-bit timer values to include in transmitted PAUSE frame. A single timer is used in LFC mode and up to eight timers are used in PFC mode.
- Flow Control Refresh Threshold Value (PRTDCB_FCRTV) — 16-bit PAUSE refresh threshold value (in legacy FC PRTDCB_FCRTV must be smaller than PRTDCB_FCTTVN[0]).
- PRTDCB_TC2PFC.TC2PFC bitmap shall be set to 0xFF in LFC mode.

For a 40 Gb/s link:

- PRTMAC_HSEC_CTL_RX_ENABLE_GPP — Controls the processing of incoming LFC frames.
- PRTMAC_HSEC_CTL_RX_ENABLE_PPP — Controls the processing of incoming PFC frames.
- PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE — Bit-map that controls the processing of incoming PFC and LFC frames. Contains an enable bit per priority for PFC and a single bit for LFC.
- PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE — Bit-map that controls the sending of PFC and LFC frames. Contains an enable bit per priority for PFC and a single bit for LFC.
- PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER — Array that controls the retransmission time of pause packets for each of the eight priorities in PFC operation and for the LFC operation.
- PRTMAC_HSEC_CTL_TX_SA_PART1, PRTMAC_HSEC_CTL_TX_SA_PART2 — Configurable source MAC address used in the pause packets sent.
- PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA — Arrays that control the timer value included in transmitted pause frames for each of the eight priorities in PFC operation and for the LFC operation.

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

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.

The 40 Gb/s MAC supports the following configurations to further manage how Rx control frames are identified:

- UC DA Check Enable (PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP).
- When enabled, the MAC might identify control frames based on the configured UC DA. Configuration is defined later in this section.
- SA Check Enable (PRTMAC_HSEC_CTL_RX_CHECK_SA_GCP).

When enabled, the MAC identifies control frames only if their MAC SA matches the configured value described later in this section.Further, the 40 Gb/s MAC also supports configurations for different addresses:

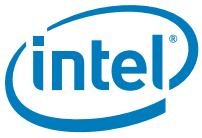
- UC MAC DA (PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1/2) — Single configuration for all control frames).
- MAC SA (PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART1/2) — Single configuration for all control frames.

The 10 Gb/s and lower speeds MAC supports the following configurations:

- UC MAC DA (PRTGL_SAL and PRTGL_SAH).

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):



1. A match on the six-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.
2. 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 frame formats

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. ***Note:***
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.

The 40 Gb/s MAC supports the following configurations to further manage 802.3x FC frames' Rx identification:

- UC MAC DA Check Enable (PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GPP) — When enabled, 802.3x frames can be identified based on a configured UC MAC_DA. Configuration is defined later in this section.
- MAC SA Check Enable (PRTMAC_HSEC_CTL_RX_CHECK_SA_GPP) — When enabled, the MAC identifies 802.3x frames only if their MAC SA matches the configured value described later in this section.

Further, when enabled for checking the 40 Gb/s MAC, it also supports configurations for the different addresses:

- UC MAC DA (PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1/2) — Single configuration for all control frames.
- MAC SA (PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART1/2) — Single configuration for all control frames.

The 10 Gb/s and lower speeds MAC supports the following configuration to further manage 802.3x FC frames' Rx identification:

- UC MAC DA (PRTGL_SAL and PRTGL_SAH).

3.2.1.5.1.3 PFC

DCB 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 DCB environment as opposed to the PAUSE mechanism defined in 802.3x. The PFC frame format is specified in IEEE P802.3bd.

Table 3-15. Packet format for PFC

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.

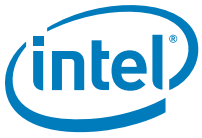
Table 3-16. Format of priority enable vector

	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, etc. The XL710 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 for the binding of received PFC frames to Tx TCs, and to the PRTDCB_RUP2TC register for the binding of transmitted PFC frames to Rx TCs.

When a PFC frame is formatted by the XL710, 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.

The 40 Gb/s MAC supports the following configurations to manage PFC frames' identification on Rx:

- UC DA Check Enable (PRTMAC_HSEC_CTL_RX_CHECK_UCAST_PPP)— When enabled, PFC frames can be identified based on a configured UC MAC_DA. Configuration is defined later in this section.
- SA Check Enable (PRTMAC_HSEC_CTL_RX_CHECK_SA_PPP)- When enabled, the MAC identifies 802.3x frames only if their MAC SA matches the configured value described later in this section.

Further, when enabled for checking the 40 Gb/s MAC, it also supports configurations for the different addresses:

- UC MAC DA (PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1/2) — Single configuration for all control frames).
- MAC SA (PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART1/2) — Single configuration for all control frames.

The 10 Gb/s and lower speeds MAC supports the following configurations to manage PFC frames' identification on Rx:

- UC MAC DA (PRTGL_SAL and PRTGL_SAH).

3.2.1.5.1.4 Operation and rules

The XL710 operates in either LFC mode or in PFC mode. Enabling both modes concurrently is not allowed.

The 10 Gb/s and lower speeds MAC implements the following configurations:

- LFC (PAUSE) is enabled by the *RFCE* bit in the PRTDCB_MFLCN register.
- PFC is enabled by the *RPFCE* bit in the PRTDCB_MFLCN register.

The 40 Gb/s MAC implements the following configurations:

- PRTMAC_HSEC_CTL_RX_ENABLE_GPP — Enables the processing of incoming LFC frames.
- PRTMAC_HSEC_CTL_RX_ENABLE_PPP — Enables the processing of incoming PFC frames.

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 DCB management agent. It is the EMP responsibility to reconfigure the LFC settings after the auto-negotiation process was resolved.

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

- Increments the appropriate statistics register(s).
- 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, then the device sets the timer to the maximum value among all enabled timer fields associated with the TC.
- Disable packet transmission or schedule the disabling of transmission after the current packet completes.
  - — 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 40 Gb/s line speed, the XL710 does not begin to transmit a (new) frame more than 118 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 XL710 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 XL710 must not begin to transmit a (new) frame more than 2 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 and 48 pause quanta at the link speed of 40 Gb/s). This extra budget in addition to the link delay is aimed to compensate the fact that decision to stop new transmissions from a specific TC must be taken earlier in the transmit data path than for the LFC case.

3.2.1.5.2 PAUSE and MAC control frames forwarding

3.2.1.5.2.1 At 10 Gb/s and lower speeds

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

- The *DPF* bit controls transfer of PAUSE packets to the host. The same policy applies to both LFC and PFC packets as listed in Table 3-17. Note that any packet must pass the L2 filters as well.
- The *PMCF* bit controls transfer of non-PAUSE packets to the host. Note that when LFC frames are not enabled (RFCE = 0b) then LFC frames are considered as MAC Control (MC) frames for this matter. Similarly, when PFC frames are not enabled (RPFCE = 0b) then PFC frames are considered as MC frames as well.

Note: When virtualization is enabled, forwarded control packets are queued according to the regular switching procedure.

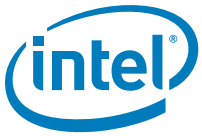


Table 3-17. Transfer of PAUSE packet to host (DPF bit) in 10 Gb/s MAC

RFCE	RPFCM	DPF	LFC Handling	PFC Handling
0b	0b	X	Treat as MC (according to PMCF setting).	Treat as MC (according to PMCF setting).
1b	0b	0b	Accept.	Treat as MC (according to PMCF setting).
1b	0b	1b	Reject.	Treat as MC (according to PMCF setting).
0b	1b	0b	Treat as MC (according to PMCF setting).	Accept.
0b	1b	1b	Treat as MC (according to PMCF setting).	Reject.
1b	1b	X	Unsupported setting.	Unsupported setting.

3.2.1.5.2.2 For 40 Gb/s links

The following configurations control the forwarding of MAC control frames (including PAUSE frames):

- PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL — Controls the forwarding of incoming MAC control frames to the host and/or to EMP. When enabled, incoming PAUSE frames are not terminated by the MAC block. Affects both PFC and LFC frames.
- PRTMAC_HSEC CTL_RX_ENABLE_GCP— Controls whether incoming control frames are processed by the MAC block.
- PRTMAC_HSEC CTL_RX_ENABLE_PPP/GPP — When control frames are enabled for processing, by setting the PRTMAC_HSEC CTL_RX_ENABLE_GCP, these configurations are used to enable PFC and LFC frame processing.

Note: When enabled for processing frames, is identified according to the configurations described in Section 3.2.1.5.1.1.1.

Table 3-18 lists these settings.

Table 3-18. Transfer of PAUSE packet-to-host in 40 Gb/s MAC

Rx Enable GCP	RX_ENABLE_PPP (PFC)	RX_ENABLE_GPP (LFC)	Rx Forward Control	PFC and LFC Handling
0b	X	X	X	PAUSE frames are forwarded as regular packets.
1b	X	X	0b	PAUSE frames are dropped.
1b	X	X	1b	PAUSE frames are forwarded as regular packets.

Table 3-19. PAUSE frame processing in 40 Gb/s MAC

RX Enable GCP	RX_ENABLE_PPP (PFC)	RX_ENABLE_GPP (LFC)	PFC and LFC Handling
0b	X	X	Both PFC and LFC frames are treated as regular packets.
1b	1b	0b	1. PFC frames are processed by MAC. 2. LFC frames are not processed by the MAC. 3. Both PFC and LFC frames are dropped/forwarded based on RX_FORWARD_CONTROL configuration as listed in Table 3-18.
1b	0b	1b	4. PFC frames are not processed by the MAC. 1. LFC frames are processed by MAC. 2. Both PFC and LFC frames are dropped/forwarded based on RX_FORWARD_CONTROL configuration as listed in Table 3-18.
1b	1b	1b	Unsupported setting.

3.2.1.5.3 Transmitting PAUSE frames

The XL710 generates PAUSE packets to insure there is enough space in its receive packet buffers to avoid packet drop. The XL710 monitors the fullness of its receive FIFOs and compares it with the contents of a programmable threshold. When the threshold is reached, the XL710 sends a PAUSE frame. The XL710 supports both LFC and PFC — but not both concurrently. When DCB is enabled, it sends only PFC, and 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 by setting the PRTDCB_FCCFG.TFCE field or the PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE register on 40 Gb/s links only after it is negotiated between the link partners (possibly by auto-negotiation).

3.2.1.5.3.1 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).

3.2.1.5.3.2 Operation and rules

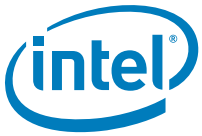
At 10 Gb/s and lower speeds, The *TFCE* field in the Flow Control Configuration (PRTDCB_FCCFG) register enables transmission of PAUSE packets as well as selects between the LFC mode and the PFC mode.

For a 40 Gb/s link, the PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE bit-map controls transmission of PFC and LFC.

Refer to Section 7.7.1.2.4 for the criteria used by the device for sending a XOFF frame to the neighbor. The XL710 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 XL710 starts counting down in an internal shadow counter that is used to mirror the pause time-out counter at the partner's end. When this internal counter reaches the value set in PRTDCB_FCRTV register or in PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER register for 40 Gb/s links 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 XL710 sends an XON message (a PAUSE frame with a timer value of zero). Refer to Section 7.7.1.2.7 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 XL710 supports transmission pacing by extending the IPG (the gap between consecutive packets). The pacing mode enables the average data rate to be slowed in systems that cannot support the full link rate (10 Gb/s, 1 Gb/s or 100 Mb/s). As listed in Table 3-20, the pacing modes work by stretching the IPG in proportion to the data sent. In this case, the data sent is measured from the end of preamble to the last byte of the packet. No allowance is made for the preamble or default IPG when using pacing mode.

Example 1:

Consider an example of a 64-byte frame. To achieve a 1 Gb/s data rate when link rate is 10 Gb/s and packet length is 64 bytes (16 Dwords), programmers need to add an additional IPG of 144 Dwords (nine times the packet size to reach 1 Gb/s). Which when added to the default IPG gives an IPG of 147 Dwords.

Example 2:

Consider an example of a 65-byte frame. To achieve a 1 Gb/s data rate when link rate is 10 Gb/s and packet length is 65 bytes (17 Dwords when rounded up) programmers need to add an additional IPG of 153 Dwords (nine times the packet duration in Dwords). Which when added to the default IPG gives an IPG of 156 Dwords. Note that in these case, where the packet length counted in Dwords is not an integer, programmers need to count any fraction of a Dword as a whole Dword for computing the additional IPG.

Table 3-20 lists the pacing configurations supported by the XL710 at link rates of 10 Gb/s. When operating at lower link speeds the pacing speed is proportional to the link speed. Pacing is configured in the *Pacing Config* field in the Set MAC config admin command.

Note: The 40 Gb/s MAC does not support the previous mechanism and pacing might be achieved through the use of the rate limiting mechanism of the Tx scheduler.

Table 3-20. Pacing speeds at 10 Gb/s link speed

Pacing Speeds (Gb/s)	Delay Inserted into IPG	*Pacing Config* field in the Set MAC config admin command
10 (LAN)	None	0000b
9.294196 (WAN)	1 byte for 13 transmitted	1111b
9.0	1 Dword for 9 transmitted	1001b
8.0	1 Dword for 4 transmitted	1000b
7.0	3 Dwords for 7 transmitted	0111b
6.0	2 Dwords for 3 transmitted	0110b
5.0	1 Dwords for 1 transmitted	0101b
4.0	3 Dwords for 2 transmitted	0100b
3.0	7 Dwords for 3 transmitted	0011b
2.0	4 Dwords for 1 transmitted	0010b
1.0	9 Dwords for 1 transmitted	0001b
10	None	Default

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 XL710 supports an additional mode of operation, where it can auto-negotiate 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 5.3.2.

3.2.1.8 MAC errors

The XL710's MAC supports identification of the following erroneous packets:

- L2 CRC Error

  Packet's FCS check resulted in an error.
- Length Error - undersize or oversize

  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.
- Illegal Byte Error

  Indication that an illegal control byte was received on the XLGMII interface.

  An illegal control byte is any value that is not /I/,/E/,/T/ or /D/.
- Error Byte Error

  Indication that a packet was received with the error control byte (/E/) on the XLGMII interface. Indicates that the PCS has encountered an error during the packet's reception.
- 802.3 Header Length Error

  <Length> field information in the 802.3 header does not match the packet's actual size.

Packet's 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: Packet's 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 3-21 lists the different MAC error counters supported by the device.

Table 3-21. MAC error counters

Counter Name	Description
CRC Error	Counts the number of packets received with CRC errors that are not fragments.
Illegal Byte Error	Counts the number of packets received with an illegal control byte on the XLGMII interface.
Error Byte	Counts the number of packets that were received with an error control byte on the 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.

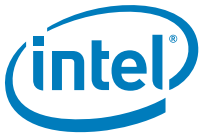


Table 3-21. MAC error counters

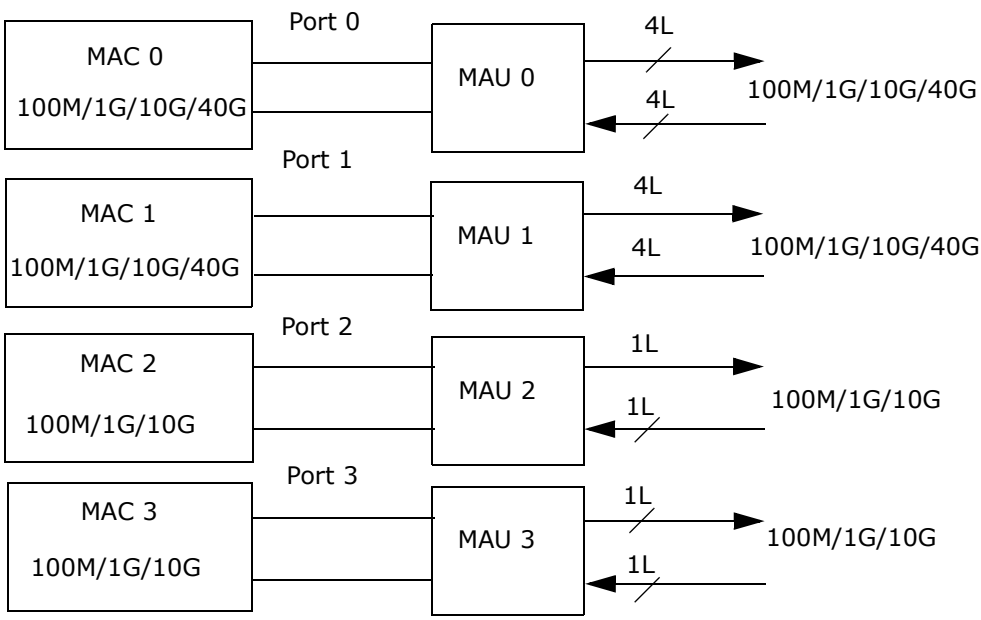
Counter Name	Description
Receive Fragment	Counts the number of packets with bad CRC that are smaller than 64 bytes
Receive Oversize	Counts the number of packet that are larger than the configured max frame size.
MAC Short Packet Discard	Counts the number of packets smaller than 32 bytes that were discarded by the MAC layer.

3.2.2 Physical Layer (PHY)

The XL710 provides up to four Ethernet MAC ports with integrated PHY interfaces to connect either directly to the medium (backplane or direct attached twinaxial copper cable assemblies) or to external PHYs. The XL710 Ethernet physical interfaces are multi-rate Medium Attachment Unit Interfaces (MAUI) that can be configured for operation at 40 Gb/s, 10 Gb/s, 1 Gb/s or 100 Mb/s link speeds. The XL710 supports eight physical high speed SerDes lanes, each capable of operating at up to 10.3125 GBaud. The XL710 supports auto-negotiation when configured for backplane Ethernet to automatically select between 1000BASE-KX, 10GBASE-KX4, 10GBASE-KR and 40GBASE-KR4. The XL710 also supports auto-negotiation for operation with 40GBASE-CR4 twinaxial copper cable assembly. The XL710 supports polarity correction on the MAUI interfaces to ease board routing when using integrated PHYs.

3.2.2.1 MAC and MAUI port configuration

The XL710 supports up to four Ethernet MAC ports and the corresponding MAU / physical interfaces can be configured to different speeds and protocols as shown in Figure 3-6.

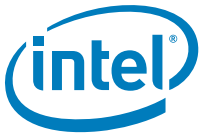


Note: Figure shows maximum configuration, not all configurations are available in all modes.

Figure 3-6. XL710 MAC and MAU interfaces

When in 40 Gb/s mode, the MAUI for port 0 and 1 can be configured for one of the following physical interfaces:

- A four lane XLAUI interface for connectivity to external 40 GbE PHYs.
- A four lane XLPPI interface for connectivity to 40GBASE-SR4 optical modules.
- A four lane 40GBASE-KR4 interface for direct connectivity to the backplane medium.



- A four lane 40GBASE-CR4 interface for direct connectivity for up to 7 m twinaxial copper cable assemblies.

When in 10 Gb/s operating mode, the MAUI for each of the Ethernet ports 0 through 3 can be independently configured for one following physical interfaces:

- A four lane XAUI interface on up to two Ethernet ports for connectivity to external PHYs.
- A four lane 10GBASE-KX4 interface on up to two Ethernet ports for direct connectivity to backplane medium.
- A single lane 10GBASE-KR interface on up to four Ethernet ports for direct connectivity to backplane medium.
- A single lane SFI interface connectivity to 10GBASE-LR/SR optical modules or for direct connectivity for up to 7 m twinaxial copper cable assemblies.

When in 1 Gb/s operating mode, the MAUI interface for each of the Ethernet ports 0 through 3 can be independently configured for one of the following physical interfaces:

- A single lane 1000BASE-KX interface for direct connectivity to backplane medium.
- A single lane SGMII (1 Gb/s) over a KX compliant electrical interface.

When in 100 Mb/s operating mode, the MAUI interface for each of the Ethernet ports 0 through 3 can be independently configured for one of the following physical interfaces:

- A single lane SGMII (100 Mb/s) over a KX compliant electrical interface.

Table 3-22. Supported Electrical Modes

Speed	Mode	PMD Spec
40G	XLAUI	IEEE 802.3 Clause 83 IEEE 802.3 Annex 83A IEEE 802.3 Annex 83B
	40GBASE-KR4	IEEE 802.3 Clause 84
	40GBASE-CR4	IEEE 802.3 Clause 85
10G	XAUI	IEEE 802.3 Clause 47
	10GBASE-KX4	IEEE 802.3 Clause 48
	SFI	SFF 8431
1G	1000BASE-KX	IEEE 802.3 Clause 36
	SGMII	IEEE 802.3 Clause 36
100M	SGMII	IEEE 802.3 Clause 36

When any of the Ethernet port is directly connected to the medium (backplane or twinaxial copper cable assembly) auto-negotiation protocol runs on MAUI lane 0 only, as per the IEEE Clause 73 requirements.

In dual port (four lane per port) configuration (40G/10G/1G: KR/KX4/KX) AN needs to run on lane 0 of the 4 lanes per port.

In quad port (one lane per port) configuration (10G/1G: KR/KX) AN needs to run on lane 0 of each of the one lane per port.

Table 3-23 lists the possible speed, interface type and lane configurations supported by the XL710 on Ethernet MAC ports 0 through 3. When Ethernet port 0 or 1 or both are configured to operate at 40 Gb/s, ports 2, 3 are disabled.

Table 3-23. Port, speed, interface type, and lane configuration

Port Num	Speed Config	Num of Lanes in Each Direction	Port Type Config	Rate Per Lane (GBaud)	MAU Lanes Used
Port 0	40 Gb/s	4	XLAUI, XLPPI, KR4, CR4	10.3125	Lane0, Lane1, Lane2, Lane3
	10 Gb/s	4	XAUI, KX4	3.125	Lane0, Lane1, Lane2, Lane3
	10 Gb/s 1 Gb/s 100 Mb/s	1	KR, SFI, KX, SGMII	10.3125, 1.25	Lane0
Port 1	40 Gb/s	4	XLAUI, XLPPI, KR4, CR4	10.3125	Lane0, Lane1, Lane2, Lane3
	10 Gb/s	4	XAUI, KX4	3.125	Lane0, Lane1, Lane2, Lane3
	10 Gb/s 1 Gb/s 100 Mb/s	1	KR, SFI, KX, SGMII	10.3125, 1.25	Lane0
Port 2	10 Gb/s 1 Gb/s 100 Mb/s	1	KR, SFI, KX, SGMII	10.3125, 1.25	Lane0
Port 3	10 Gb/s 1 Gb/s 100 Mb/s	1	KR, SFI, KX, SGMII	10.3125, 1.25	Lane0

3.2.2.1.1 MAU logical lanes to physical pin configuration

The XL710 supports eight physical SerDes lanes, each capable of operating at up 10.3125 GBaud. The lanes are organized in two groups A and B. The differential output lanes are named TXA_L0_p/n to TXA_L3_p/n for group A and TXB_L0_p/n to TXB_L3_p/n for group B. The differential input lanes are named RXA_L0_p/n to RXA_L3_p/n for group A, and RXB_L0_p/n to RXB_L3_p/n for group B. See Figure 3-7 for a functional block diagram of MAU logical lane to physical lane mapping. Depending on the XL710 port configuration, the MAC ports and corresponding MAUI lanes can be mapped to the physical lanes in group A or group B to adapt to different system configurations for blade server applications.

The XL710 provides different combinations of logical to physical lane configurations to enable flexible board routing for blade systems and LOMs, and to enable backward compatibility to 82599 pin out. See Section 3.2.2.2 for logical lane to physical pin mapping for the supported port configurations.

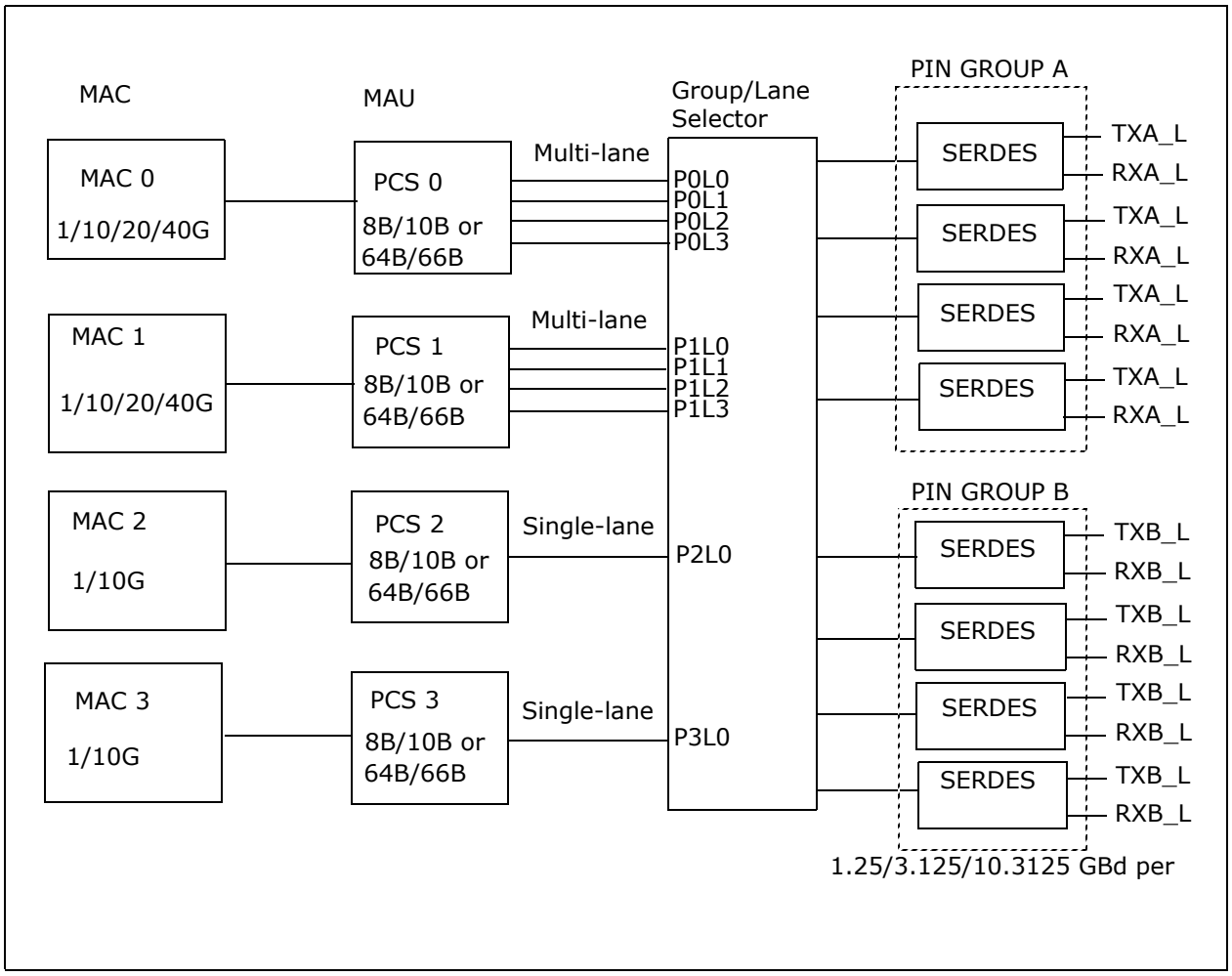


Figure 3-7. MAU logical lane to physical lane mapping

3.2.2.2 Possible port-to-physical lane configurations for various PHY interfaces

See Table 3-51 for possible port to physical lane configurations allowed by the XL710 for various PHY interfaces for single, dual and quad port configurations.

3.2.2.3 40 Gb/s interfaces

The XL710 provides complete functionality to support two 40 Gb/s ports. The device performs all functions required for transmission and reception as specified in IEEE Std.

The XL710 includes a PHY interface (XLAUI) to attach to external physical layer components, and XLPPI interface for direct connectivity to SR4 optical modules.

The XL710 also integrates complete PHY layer functionality for direct connectivity to backplane medium (KR4) or to copper cable assemblies (CR4).

3.2.2.3.1 XLAUI operating mode

The 40Gb/s Attachment Unit Interface (XLAUI) supports data rates of 40 Gb/s over four differential pairs in each direction for a total of eight pairs, with each pair operating at 10.3125 Gbaud. XLAUI is used to connect the XL710 to an external 40 GbE PHY device.

3.2.2.3.1.1 XLAUI overview

XLAUI is a full-duplex interface that uses four self-clocked serial differential links in each direction to achieve 40 Gb/s data throughput. Each serial link operates at 10.3125 Gbaud and carries 64B/66B encoded data. The 40GBASE-R multilane PCS distributes the encoded data on to four lanes, inserts alignment markers on transmit, and performs lane alignment on receive. Functional and electrical specifications of XLAUI can be found in IEEE Clause 83, and Annexes 83A, 83B. The architectural positioning of XLAUI is shown in Figure 3-8. XLAUI is inserted between the PMA sublayers to enable chip-to-chip communication.

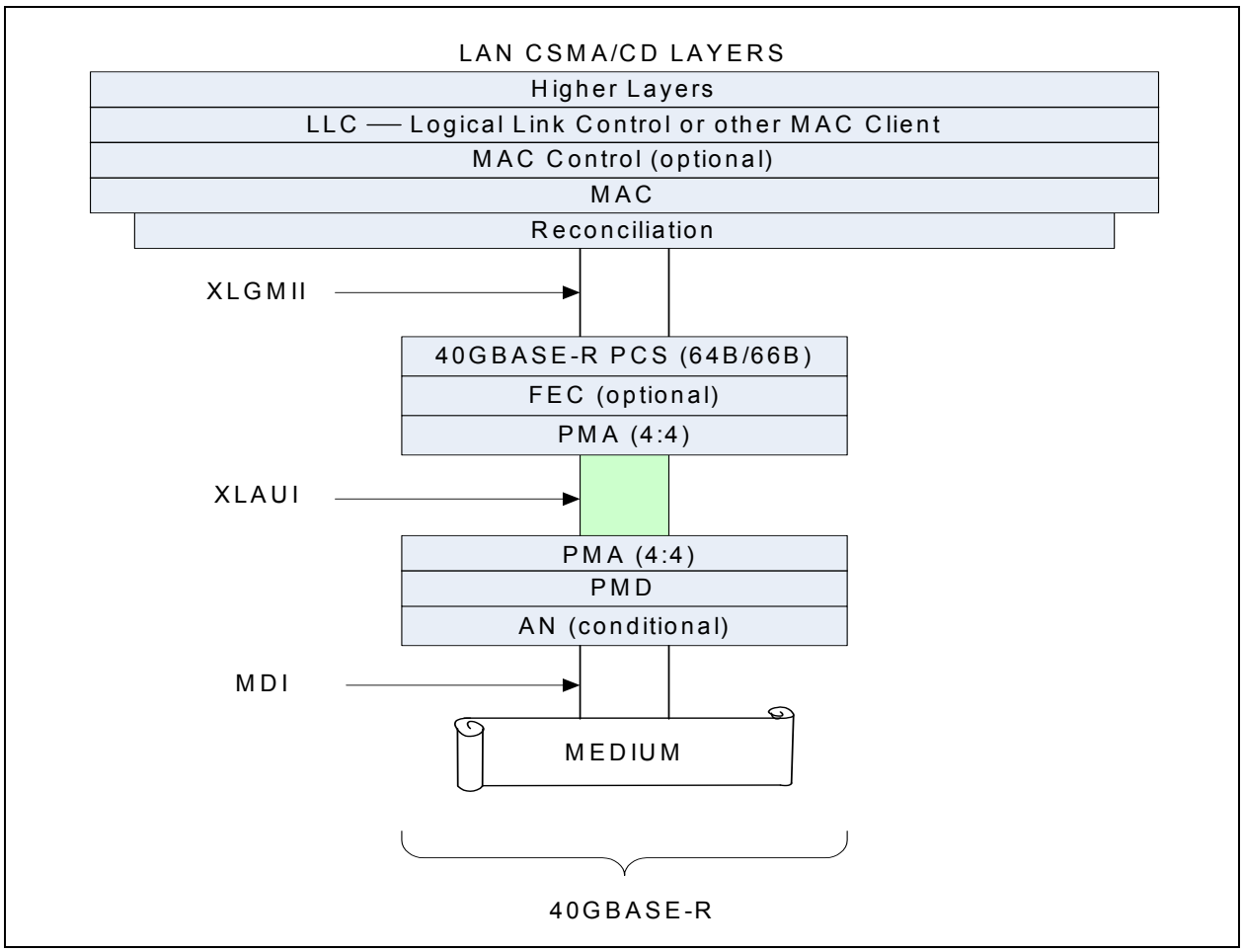
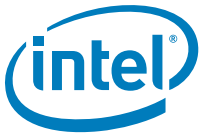


Figure 3-8. Architectural positioning of XLAUI



The XLAUI interface has the following characteristics:

a. Independent transmit and receive data paths, four lanes in each direction.
b. Differential AC coupled signaling with low voltage swing.
c. Self-timed interface.
d. Uses 64B/66B coding.
e. Each lane operates at a signaling rate of 10G.3125 GBaud.

3.2.2.3.1.2 XLAUI electrical characteristics

The XLAUI lane is a low swing AC coupled differential interface using NRZ signaling. AC coupling allows for inter-operability between components operating at different supply voltages. Differential signal swings specifications depend on several factors, such as transmitter de-emphasis and transmission line losses.

The XLAUI signal paths are point-to-point connections. Each path corresponds to a XLAUI lane and is comprised of two complementary signals making a balanced differential pair. There are four differential pairs in each direction for a total of eight pairs, or 16 connections. The electrical characteristics of XLAUI interface can be found in IEEE Std Annexes 83A, 83B.

3.2.2.3.2 40GBASE-KR4 operating mode

The KR4 operating mode supports data rates of 40 Gb/s over differential controlled impedance copper traces over improved FR4 PCBs. Data is transferred over four differential pairs in each direction for a total of eight pairs, with each pair operating at 10.3125 GBaud. The interface is used to connect the XL710 to a KR4 switch port over a backplane (or midplane) medium.

KR4 supports Clause 73 auto-negotiation, used for auto configuration of the backplane link to one of the following modes: KR4, KR, KX4, or KX. Auto-negotiation also supports parallel detect functionality to auto-detect legacy KX or KX4 PHYs that do not support auto-negotiation (or has disabled auto-negotiation).

The MAU interface is configured for KR4 operation when auto-negotiation detects a KR4 link partner. KR4 operation can also be configured through the Set PHY config admin command.

3.2.2.3.2.1 KR4 overview

The 40GBASE-KR4 PMD is defined in IEEE Clause 84. KR4 specifies 40 Gb/s operation over four differential pairs in each direction for a total of eight pairs, or 16 connections. This system uses the 40GBASE-R PCS and PMA as defined in IEEE Std Clause 82 and Clause 83 with amendments for Clause 73 auto-negotiation. KR4 is a full-duplex interface that uses four self-clocked serial differential links in each direction to achieve 40 Gb/s data throughput. Each serial link operates at 10.3125 Gbaud to accommodate both data and overhead. The 40GBASE-R multi-lane PCS provides lane alignment and lane de-skew capabilities. Figure 3-9 shows the architectural positioning of 40GBASE-KR4. The XL710

supports Forward Error Correction (FEC) when operating in KR4 mode. FEC can be enabled through auto-negotiation. FEC provides better link performance that can be used to lower the BER of the backplane link.

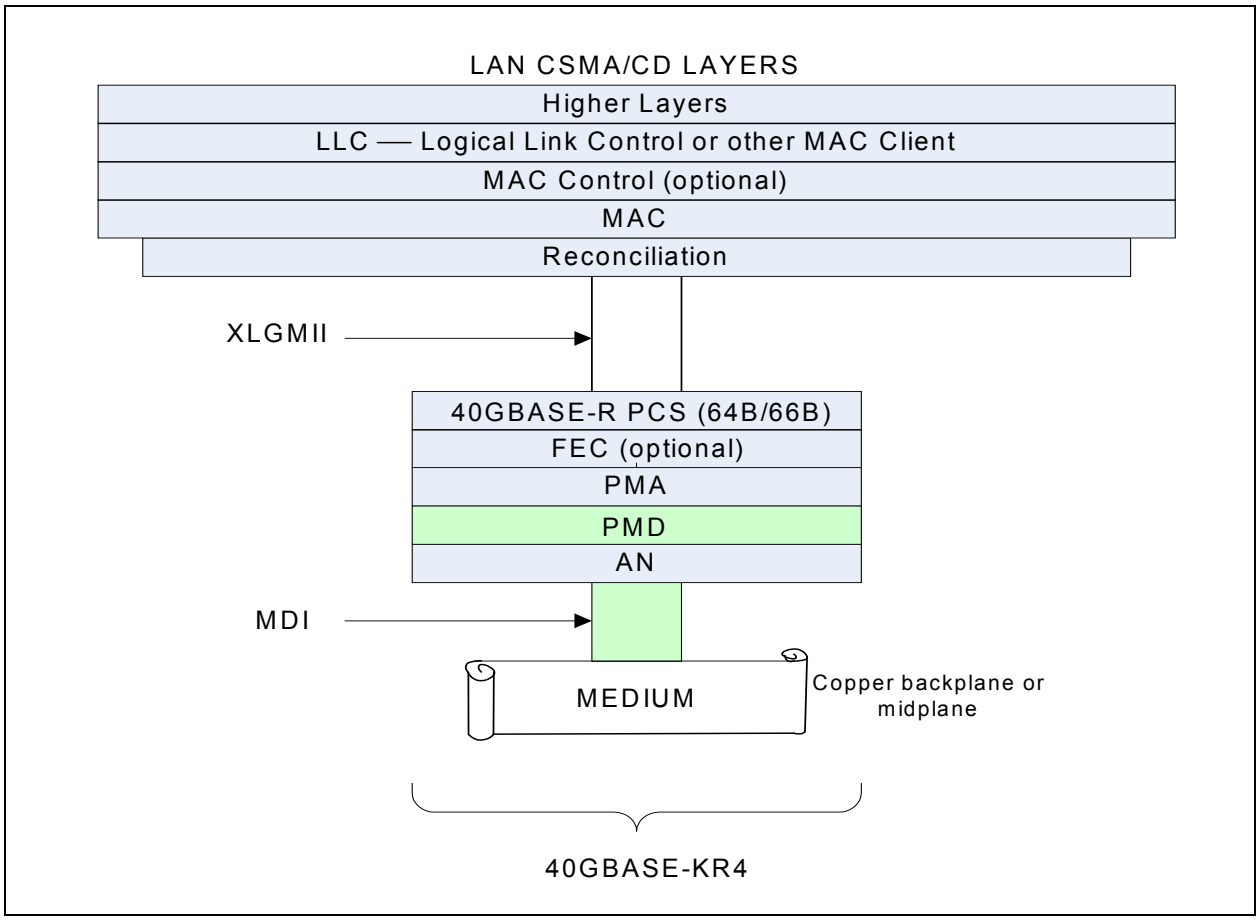


Figure 3-9. Architectural positioning of 40GBASE-KR4

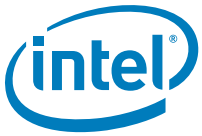
3.2.2.3.2.2 KR4 electrical characteristics

The KR4 lane is a low swing AC coupled differential interface using NRZ signaling. AC coupling allows for inter-operability between components operating from at different supply voltages. Low swing differential signaling provides noise immunity and reduced EMI. Differential signal swings specifications depend on several factors, such as transmitter pre-equalization and transmission line losses.

The KR signal paths are point-to-point connections. Each path corresponds to a KR lane and is comprised of two complementary signals making a balanced differential pair. There are four differential pairs in each direction for a total of eight pairs, or 16 connections.

The 40GBASE-KR4 link requires a nominal 100 Ω differential source and load terminations with AC coupling on the receive side. The signal paths are intended to operate at up to one meter, including two connectors, over controlled impedance traces on improved FR4 PCBs.

3.2.2.3.3 40GBASE-CR4 operating mode



The CR4 operating mode supports data rates of 40 Gb/s over up to 7 m of shielded balanced copper cable assemblies. Data is transferred over four differential pairs in each direction for a total of eight pairs, with each pair operating at 10.3125 Gbaud. The interface is used to directly connect the XL710 to an edge access switch or an end station through copper cable assembly. CR4 supports Clause 73 auto-negotiation, used for auto configuration of the copper link and to automatically enable FEC if requested by the LP. The XL710 supports the FEC when operating in CR4 mode. FEC can be enabled through auto-negotiation. FEC provides better link performance to lower the BER of the copper cable medium.

The MAU interface is configured for CR4 operation when auto-negotiation detects a CR4 link partner. CR4 operation can also be configured through Set PHY config admin command. Clause 73 auto-negotiation is used for both KR and CR links, hence the firmware should ensure appropriate technology ability bits are configured in the auto-negotiation base page register based on the system configuration (backplane based boards or copper cable assembly boards).

3.2.2.3.3.1 CR4 overview

The 40GBASE-CR4 PMD is defined in IEEE Std 802.3ba-2010 Clause 85. CR4 specifies 40 Gb/s operation over up 7 m of copper cable assemblies with four differential pairs in each direction for a total of eight pairs, or 16 connections. This system uses the 40GBASE-R PCS and PMA as defined in IEEE P802.3ba Clause 82 and Clause 83 with amendments for Clause 73 auto-negotiation. CR4 is a full-duplex interface that uses four self-clocked serial differential links in each direction to achieve 40 Gb/s data throughput. Each serial link operates at 10.3125 Gbaud to accommodate both data and overhead. The 40GBASE-R multi-lane PCS provides lane alignment and lane de-skew capabilities. Figure 3-10 shows the architectural positioning of 40GBASE-CR4.

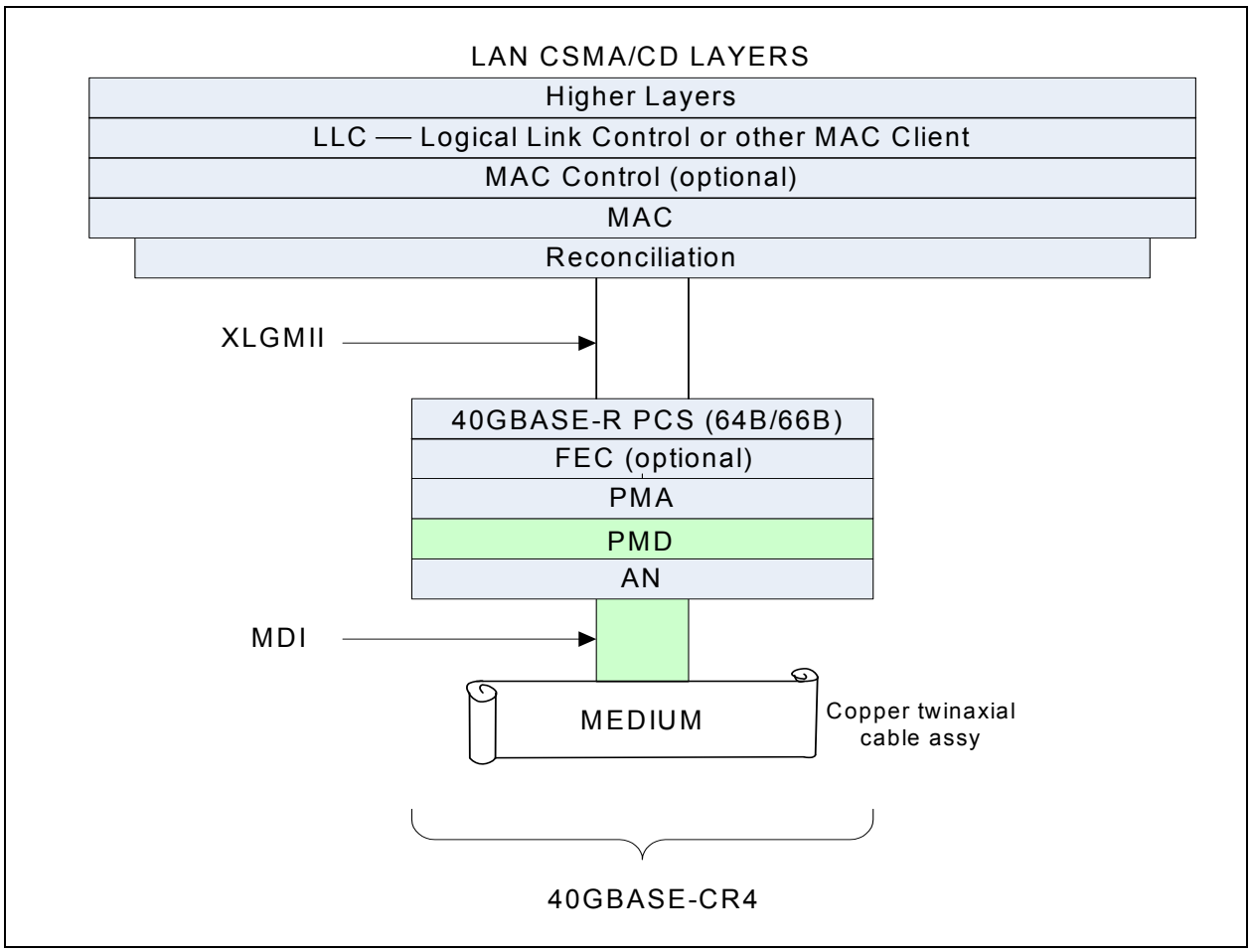


Figure 3-10. Architectural positioning of 40GBASE-CR4

3.2.2.3.3.2 CR4 electrical characteristics

The CR4 lane is a low swing AC coupled differential interface using NRZ signaling. AC coupling allows for inter-operability between components operating from at different supply voltages. Low swing differential signaling provides noise immunity and reduced EMI. Differential signal swings specifications depend on several factors, such as transmitter pre-equalization and transmission line losses.

The CR4 signal paths are point-to-point connections. Each path corresponds to a lane and is comprised of two complementary signals making a balanced differential pair. There are four differential pairs in each direction for a total of eight pairs, or 16 connections.

The 40GBASE-CR4 link requires AC coupling on the receive side. The CR4 signal paths are intended for operation over twinaxial cable assemblies ranging from 0.5 m to 7 m in length. The differential insertion loss for PCB traces on the board from transmitter to connector is provided in an informative specification in IEEE Std 802.3ba-2010 Annex 85A.

3.2.2.3.4 XLPPI operating mode

The 40Gb/s Parallel Physical Interface (XLPPI) supports data rates of 40 Gb/s over four differential pairs in each direction for a total of eight pairs, with each pair operating at 10.3125 Gbaud. XLPPI is used to directly connect the XL710 to an external 40GBASE-SR4 optical modules. These type of optical modules do not have clock and data recovery circuits inside the modules. Figure 3-11 shows the architectural positioning of XLPPI and SR4/LR4 PMD.

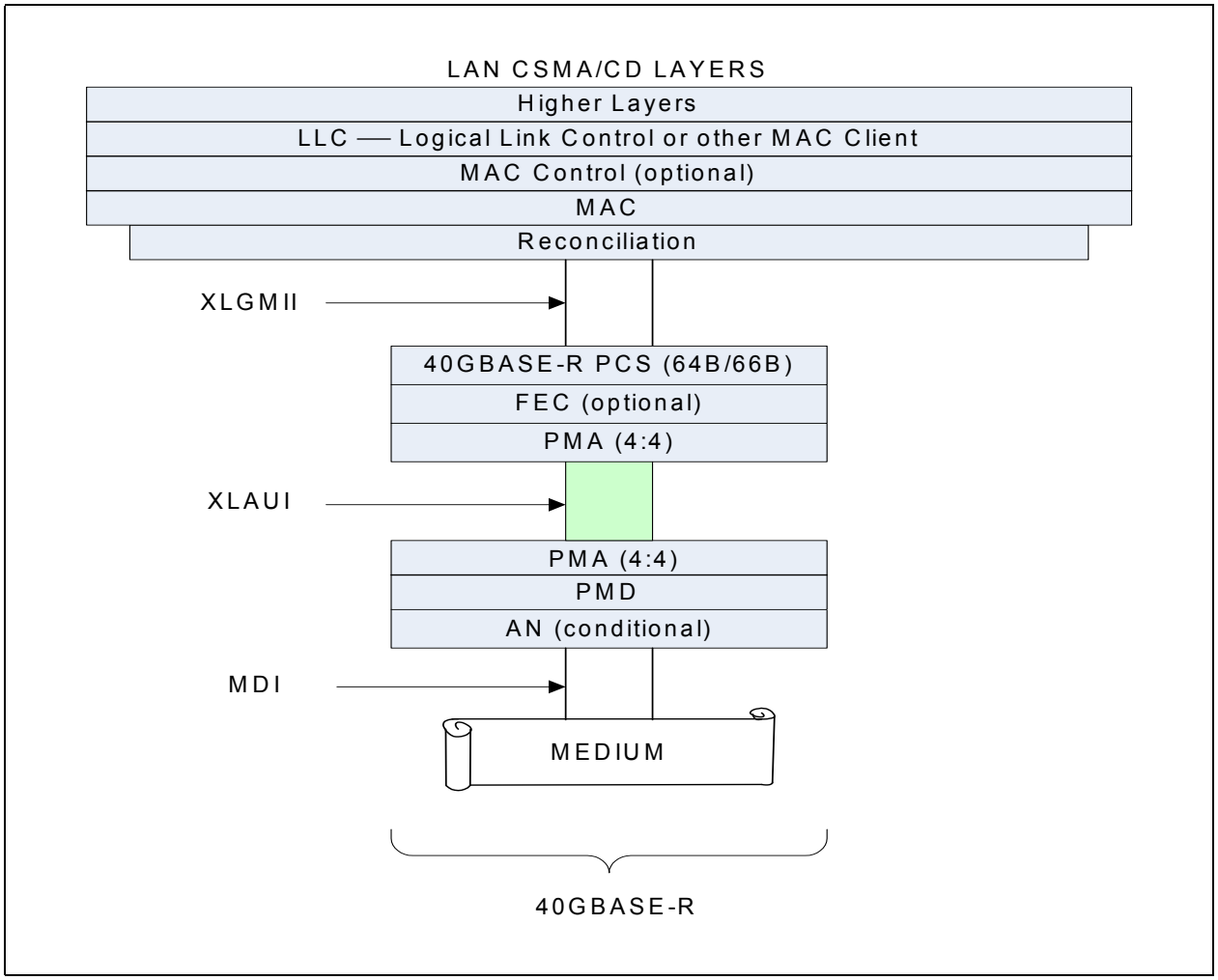


Figure 3-11. Architectural positioning of XLPPI and SR4/LR4 PMD

3.2.2.3.4.1 XLPPI overview

XLPPI is a 40 Gb/s full-duplex non re-timed serial differential link with four lanes in each direction. Each serial link operates at 10.3125 Gbaud and carries 64B/66B encoded data. The 40GBASE-R multilane PCS distributes the encoded data on to four lanes, inserts alignment markers on transmit, and takes care of lane alignment on receive. Functional and electrical specifications of XLPPI can be found in IEEE IEEE Std 802.3ba-2010 Annex 86A. XLPPI is a PMD service interface to enable chip to module communication for 40GBASE-SR4 optical PMDs.

3.2.2.4 10 Gb/s interfaces

The XL710 provides complete functionality to support up to four 10 Gb/s ports. The device performs all functions required for transmission and reception defined in the various standards.

A PHY interface is included to attach either to an external PMA or Physical Medium Dependent (PMD) components.

The XL710 enables 10 Gb/s operation compliant to the XAUI, KX4, KR, SFI specifications.

3.2.2.4.1 XAUI operating mode

The 10 Gb/s Attachment Unit Interface (XAUI) supports data rates of 10 Gb/s over four differential paths in each direction for a total of eight pairs, with each path operating at 3.125 Gb/s. The interface is used to connect the XL710 to an external 10 Gb/s PHY device with a XAUI interface. XAUI operating mode can be forced by firmware (see Section 3.2.4).

When configured to operate in XAUI mode the XL710 supports IEEE 802.3az EEE operation (see Section 5.3.1.7.3 for further information).

3.2.2.4.1.1 XAUI overview

XAUI is a full-duplex interface that uses four self-clocked serial differential links in each direction to achieve 10 Gb/s data throughput. Each serial link operates at 3.125 GBaud to accommodate both data and the overhead associated with 8B/10B coding. The self-clocked nature eliminates skew concerns between clock and data, and enables a functional reach of up to 50 cm. Conversion between the XGMII and XAUI interfaces occurs at the XGXS (XAUI Extender Sublayer). Functional and electrical specifications of XAUI interface can be found in IEEE802.3 clause 47.

The XAUI interface has the following characteristics:

a. Simple signal mapping to the XGMII.
b. Independent transmit and receive data paths.
c. Four lanes conveying the XGMII 32-bit data and control.
d. Differential signaling with low voltage swing.
e. Self-timed interface enables jitter control to the PCS.
f. Using 8B/10B coding.

Figure 3-12 Shows the architectural positioning of XAUI.

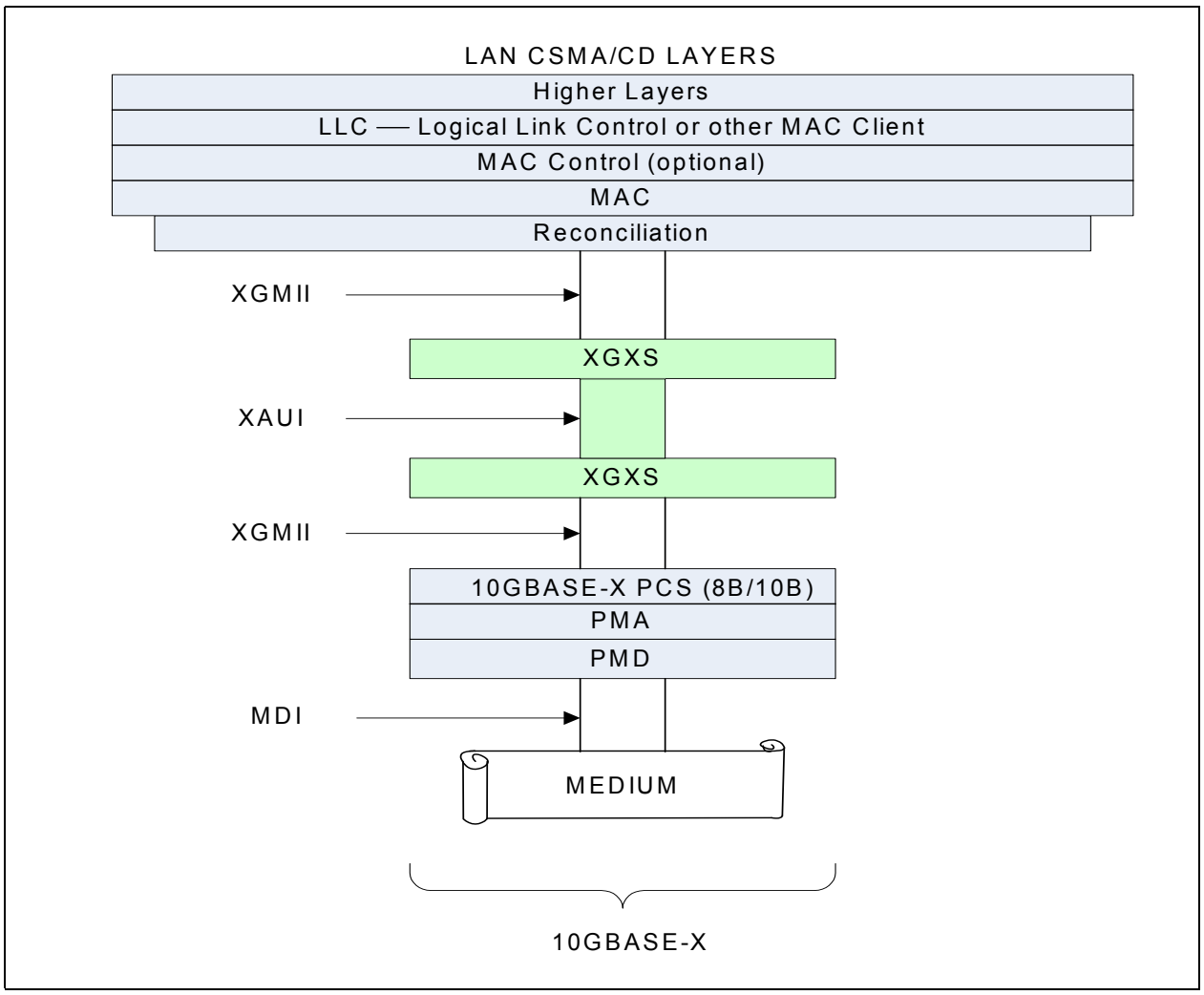


Figure 3-12. Architectural positioning of XAUI

3.2.2.4.1.2 XAUI operation

XAUI supports the 10 Gb/s data rate of the XGMII. The 10 Gb/s MAC data stream is converted into four lanes at the XGMII interface. The byte stream of each lane is 8B/10B encoded by the XGXS for transmission across the XAUI at a nominal rate of 3.125 GBaud. The XGXS and XAUI at both sides of the connection (MAC or PHY) can operate on independent clocks.

The following is a list of the major concepts of XGXS and XAUI:

1. The XGMII is organized into four lanes with each lane conveying a data octet or control character on each edge of the associated clock. The source XGXS converts bytes on an XGMII lane into a self clocked, serial, 8B/10B encoded data stream. Each of the four XGMII lanes is transmitted across one of the four XAUI lanes.
2. The source XGXS converts XGMII Idle control characters (inter-frame) into an 8B/10B code sequence.

3. The destination XGXS recovers clock and data from each XAUI lane and de-skews the four XAUI lanes into the single-clock XGMII.
4. The destination XGXS adds to or deletes from the inter-frame gap as needed for clock rate disparity compensation prior to converting the inter-frame code sequence back into XGMII idle control characters.
5. The XGXS uses the same code and coding rules as the 10GBASE-X PCS and PMA specified in IEEE 802.3 Clause 48.

3.2.2.4.1.3 XAUI electrical characteristics

The XAUI lane is a low swing AC coupled differential interface using NRZ signaling. AC coupling allows for inter-operability between components operating at different supply voltages. Low swing differential signaling provides noise immunity and reduced Electromagnetic Interference (EMI). Differential signal swings specifications depend on several factors, such as transmitter pre-equalization and transmission line losses.

The XAUI signal paths are point-to-point connections. Each path corresponds to a XAUI lane and is comprised of two complementary signals making a balanced differential pair. There are four differential paths in each direction for a total of eight pairs, or 16 connections. The signal paths are intended to operate up to approximately 50 cm over controlled impedance traces on standard FR4 Printed Circuit Boards (PCBs).

3.2.2.4.2 10GBASE-KX4 operating mode

The KX4 interface supports data rates of 10 Gb/s over copper traces in improved FR4 PCBs. Data is transferred over four differential paths in each direction for a total of eight pairs, with each path operating at 3.125 Gbaud to support overhead of 8B/10B coding. The interface is used to connect the XL710 to a KX4 switch port over the backplane or to an external 10 GbE PHY device with a KX4 interface.

The MAUI interface is configured as a KX4 interface while auto-negotiation to a KX4 link partner is detected. KX4 operation can also be forced by firmware (see Section 3.2.4).

When configured to operate in 10GBASE-KX4 mode the XL710 supports IEEE802.3az EEE operation (see Section 5.3.1 for further information).

3.2.2.4.2.1 KX4 overview

10GBASE-KX4 definition is based on XAUI and specifies 10 Gb/s operation over four differential paths in each direction for a total of eight pairs, or 16 connections. This system uses the 10GBASE-X PCS and PMA as defined in IEEE802.3 Clause 48 with amendments for auto-negotiation as specified in IEEE802.3ap. The 10GBASE-KX4 PMD is defined in IEEE802.3ap Clause 71.

KX4 is a full-duplex interface that uses four self-clocked serial differential links in each direction to achieve 10 Gb/s data throughput. Each serial link operates at 3.125 Gbaud to accommodate both data and the overhead associated with 8B/10B coding. The self-clocked nature eliminates skew concerns between clock and data, and enables a functional reach of up to one meter.

Figure 3-13 shows the architectural positioning of 10GBASE-KX4.

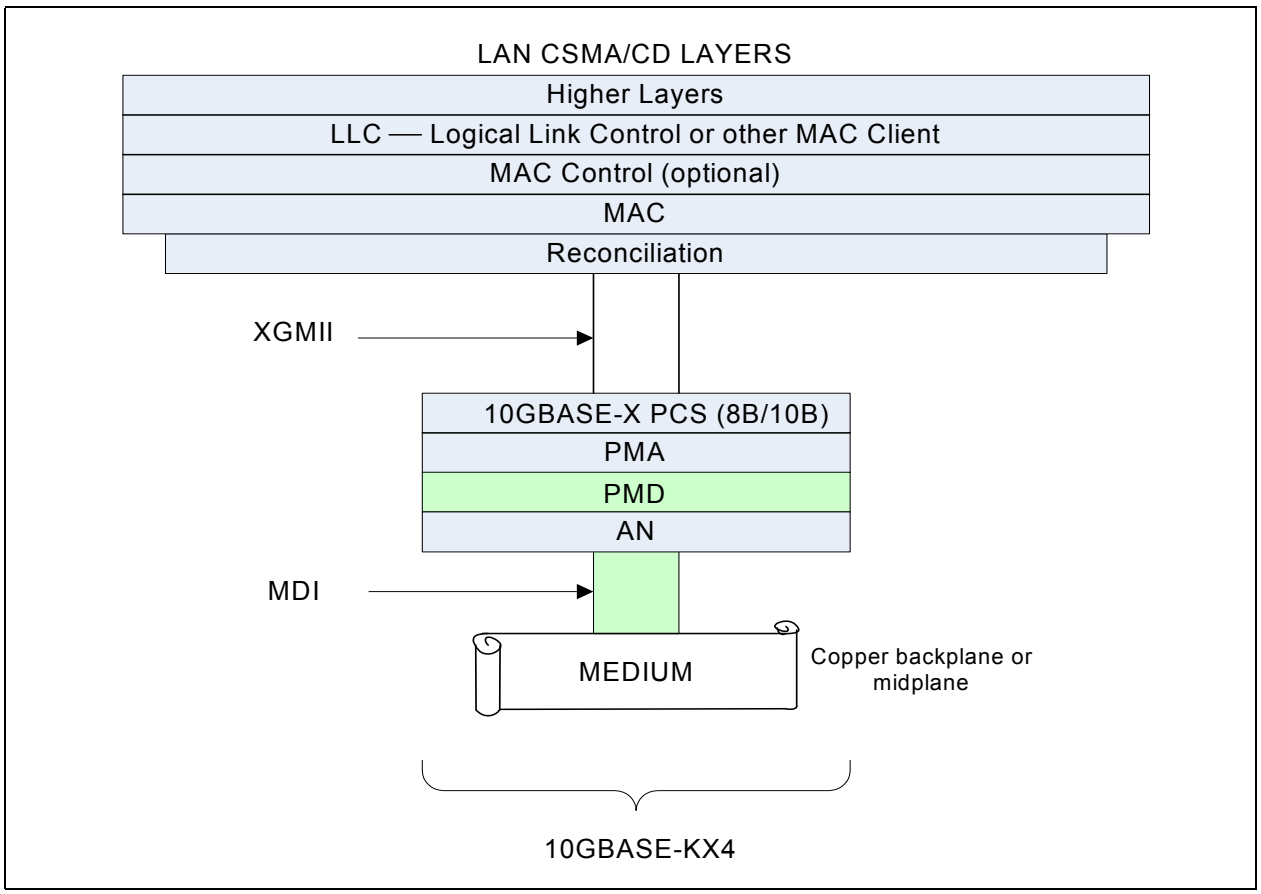


Figure 3-13. Architectural positioning of 10GBASE-KX4

3.2.2.4.2.2 KX4 electrical characteristics

The KX4 lane is a low swing AC coupled differential interface using NRZ signaling. AC coupling allows for inter-operability between components operating at different supply voltages. Low swing differential signaling provides noise immunity and reduced EMI. Differential signal swings specifications depend on several factors, such as transmitter pre-equalization and transmission line losses.

The KX4 signal paths are point-to-point connections. Each path corresponds to a KX4 lane and is comprised of two complementary signals making a balanced differential pair. There are four differential paths in each direction for a total of eight pairs, or 16 connections. The signal paths are intended to operate up to approximately one meter over controlled impedance traces on improved FR4 PCBs.

3.2.2.4.3 10GBASE-KR operating mode

The KR interface supports data rates of 10 Gb/s over copper traces in improved FR4 PCBs. Data is transferred over a single differential path in each direction for a total of two pairs, with each path operating at 10.3125 Gbaud ± 100 ppm to support overhead of 64B/66B coding. The interface is used to connect the XL710 to a KR switch port over the backplane.

The MAUI interface is configured as a KR interface while auto-negotiation to a KR link partner is detected. KR operation can also be forced by firmware (see Section 3.2.4). When in 10GBASE-KR operating mode, MAUI lane 0 is used for receive and transmit activity while lanes 1 to 3 of the MAUI interface are powered down.

When configured to operate in 10GBASE-KR mode the XL710 supports IEEE802.3az EEE operation (see Section 5.3.1 for further information).

3.2.2.4.3.1 KR overview

10GBASE-KR definition enables 10 Gb/s operation over a single differential path in each direction for a total of two pairs, or four connections. This system uses the 10GBASE-KR PCS as defined in IEEE802.3 Clause 49 with amendments for auto-negotiation specified in IEEE Std 802.3ap and 10 Gigabit PMA as defined in IEEE Std 802.3 Clause 51. The 10GBASE-KR PMD is defined in IEEE802.3ap Clause 72. The 10GBASE-KR PHY includes 10GBASE-KR FEC, as defined in IEEE Std 802.3ap Clause 74. FEC support is optional and is negotiated between Link partners during auto-negotiation as defined in IEEE Std 802.3ap Clause 73. Activating FEC improves link quality (2dB coding gain) by enabling correction of up to 11 bit-burst errors.

KR is a full-duplex interface that uses a single self-clocked serial differential link in each direction to achieve 10 Gb/s data throughput. The serial link transfers scrambled data at 10.3125 Gbaud to accommodate both data and the overhead associated with 64B/66B coding. The self-clocked nature eliminates skew concerns between clock and data, and enables a functional reach of up to one meter.

Following initialization and auto-negotiation 10GBASE-KR defines a start-up protocol, where link partners exchange continuous fixed length training frames using differential Manchester Encoding (DME) at a signaling rate equal to one quarter of the 10GBASE-KR signaling rate. This protocol facilitates timing recovery and receive equalization while also providing a mechanism through which the receiver can tune the transmit equalizer to optimize performance over the backplane interconnect. Successful completion of the start-up protocol enables transmission of data between the link partners.

Figure 3-14 shows the architectural positioning of 10GBASE-KR

.

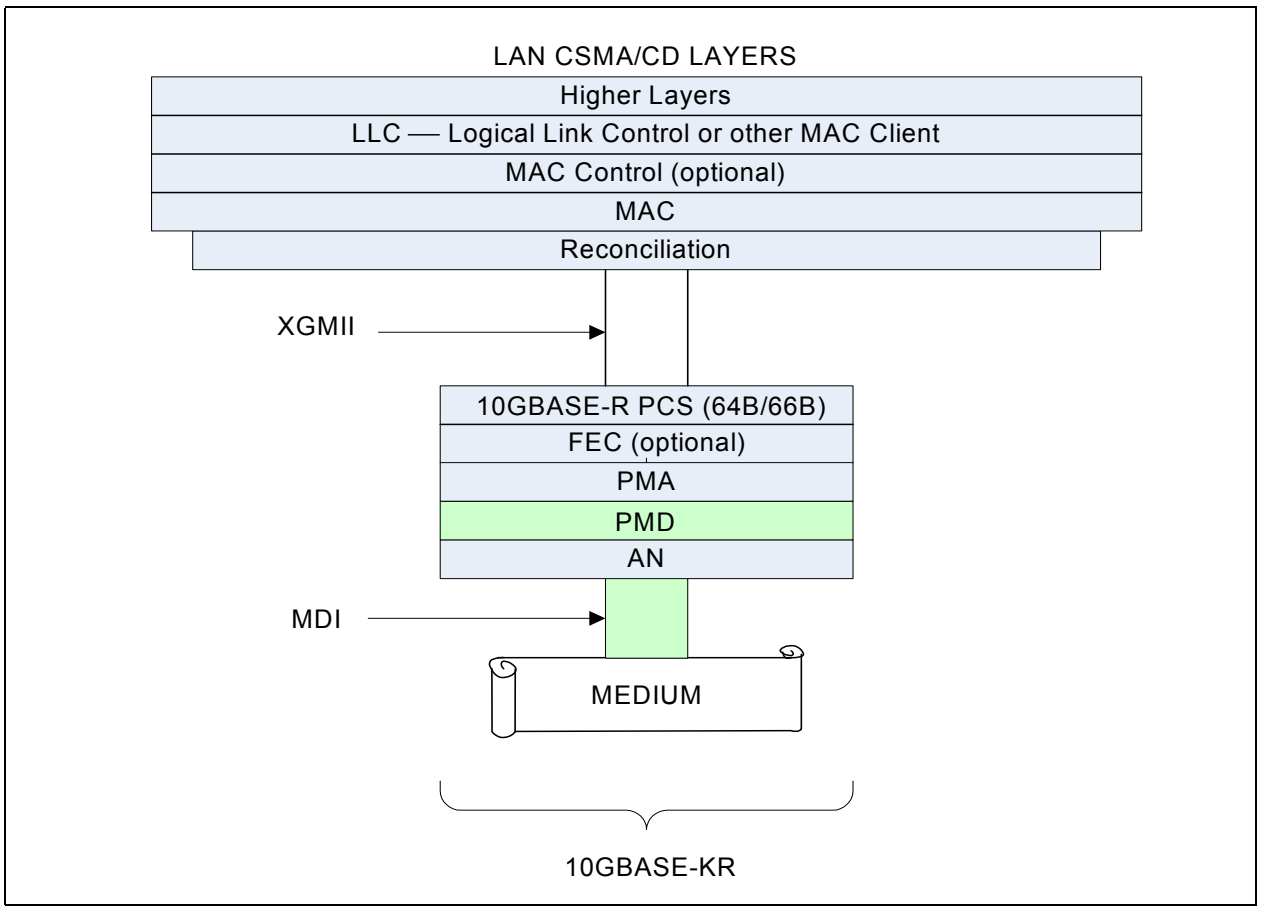


Figure 3-14. Architectural positioning of 10GBASE-KR

3.2.2.4.3.2 KR electrical characteristics

The KR lane is a low swing AC coupled differential interface using NRZ signaling. AC coupling allows for inter-operability between components operating from at different supply voltages. Low swing differential signaling provides noise immunity and improved reduced EMI. Differential signal swings defined specifications depend on several factors, such as transmitter pre-equalization and transmission line losses.

The KR signal paths are point-to-point connections. Each path corresponds to a KR lane and is comprised of two complementary signals making a balanced differential pair. There is a single differential path in each direction for a total of two pairs, or four connections.

The 10GBASE-KR link requires a nominal 100 Ω differential source and load terminations with AC coupling on the receive side. The signal paths are intended to operate up to approximately one meter, including two connectors, over controlled impedance traces on improved FR4 PCBs.

3.2.2.4.3.3 KR Reverse Polarity

The XL710 supports reverse polarity of the KR transmit and receive lanes. It is enabled by NVM settings in the Core 0/1 Analog Configuration Modules.

3.2.2.4.4 SFI operating mode

The MAUI interface is configured as SFI by firmware (see Section 3.2.4). When in SFI operating mode, only the operation of the XL710 Analog Front End (AFE) is modified, while the rest of the XL710 logic and circuitry operates similar to 10GBASE-KR.

When configured to operate in SFI mode, the XL710 supports IEEE 802.3az EEE operation (see Section 5.3.1.7.4 for further information).

3.2.2.4.4.1 SFI overview

SFI definition enables 10 Gb/s operation over a single differential path in each direction for a total of two pairs, or four connections. When in SFI operating mode the XL710 uses the 10GBASE-R PCS and 10 Gigabit PMA as defined in IEEE802.3 Clause 49 and 51, respectively.

SFI is a full-duplex interface that uses a single self-clocked serial differential link in each direction to achieve 10 Gb/s data throughput. The serial link transfers scrambled data at 10.3125 GBaud to accommodate both data and the overhead associated with 64B/66B coding. The self-clocked nature eliminates skew concerns between clock and data.

3.2.2.4.4.2 SFI electrical characteristics

The SFI lane is a low swing AC coupled differential interface using NRZ signaling. AC coupling allows for inter-operability between components operating from at different supply voltages. Low swing differential signaling provides noise immunity and improved reduced EMI. Differential signal swings defined specifications depend on several factors, such as transmitter pre-equalization and transmission line losses.

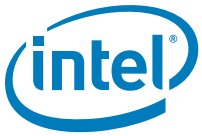
The SFI signal paths are point-to-point connections. Each path corresponds to a SFI lane and is comprised of two complementary signals making a balanced differential pair. There is a single differential path in each direction for a total of two pairs, or four connections. The signal paths are intended to operate on FR4 PCBs.

SFI interface typically operates over 200 mm of improved FR4 material or up to about 150 mm of standard FR4 with one connector. The electrical interface is based on high speed low voltage AC coupled logic with a nominal differential impedance of 100 Ω. The SFI link requires nominal 100 Ω differential source and load terminations on both the host board and the module. The SFI terminations provide both differential and common mode termination to effectively absorb differential and common mode noise and reflections. All SFI transmitters and receivers are AC coupled. SFP+ modules incorporate blocking capacitors on all SFI lines.

3.2.2.5 1 Gb/s interface

The XL710 provides complete support for up to two 1 Gb/s port implementations. The device performs all functions required for transmission and reception defined by the different standards.

A lower-layer PHY interface is included to attach either to external PMA or PMD components.



When operating in 1 Gb/s operation mode, the XL710 uses Lane 0 of the XAUI interface for 1 Gb/s operation while the other three XAUI lanes are powered down.

The XL710 enables 1 Gb/s operation compliant with the KX, BX or SGMII specifications.

3.2.2.5.1 1000BASE-KX operating mode

The MAUI interface, when operating as a KX interface, supports data rates of 1 Gb/s over copper traces on improved FR4 PCBs. Data is transferred over a single differential path in each direction for a total of two pairs (Lane 0 of MAUI interface and Lanes 1 to 3 powered down), with each path operating at 1.25 Gbaud to support overhead of 8B/10B coding. The interface is used to connect the XL710 to a KX compliant switch port over the backplane or to KX compliant 1 Gb/s PHY device. In the event of auto-negotiation defined in IEEE802.3ap clause 73 ending with 1 Gb/s as the HCD, the MAUI interface is configured as a KX interface. KX operating mode can also be forced by firmware (see Section 3.2.4).

3.2.2.5.1.1 KX overview

1000BASE-KX extends the family of 1000BASE-X PHY signaling systems. KX specifies operation at 1 Gb/s over two differential, controlled impedance pairs of traces (one pair for transmit, one pair for receive). This system uses the 1000BASE-X PCS and PMA as defined in IEEE802.3 Clause 36 together with the amendments placed in IEEE802.3ap. The 1000BASE-KX PMD is defined in IEEE802.3ap Clause 70.

KX is a full-duplex interface that uses a single serial differential link in each direction to achieve 1 Gb/s data throughput. Each serial link operates at 1.25 GBaud to accommodate both data and the overhead associated with 8B/10B coding. The self-clocked nature eliminates skew concerns between clock and data, and enables a functional reach of up to one meter. The XL710 also supports IEEE 802.3az EEE operation when configured to operate in 1000BASE-KX mode (see Section 5.3.1 for further information).

Figure 3-15 shows the architecture positioning of 1000BASE-KX.

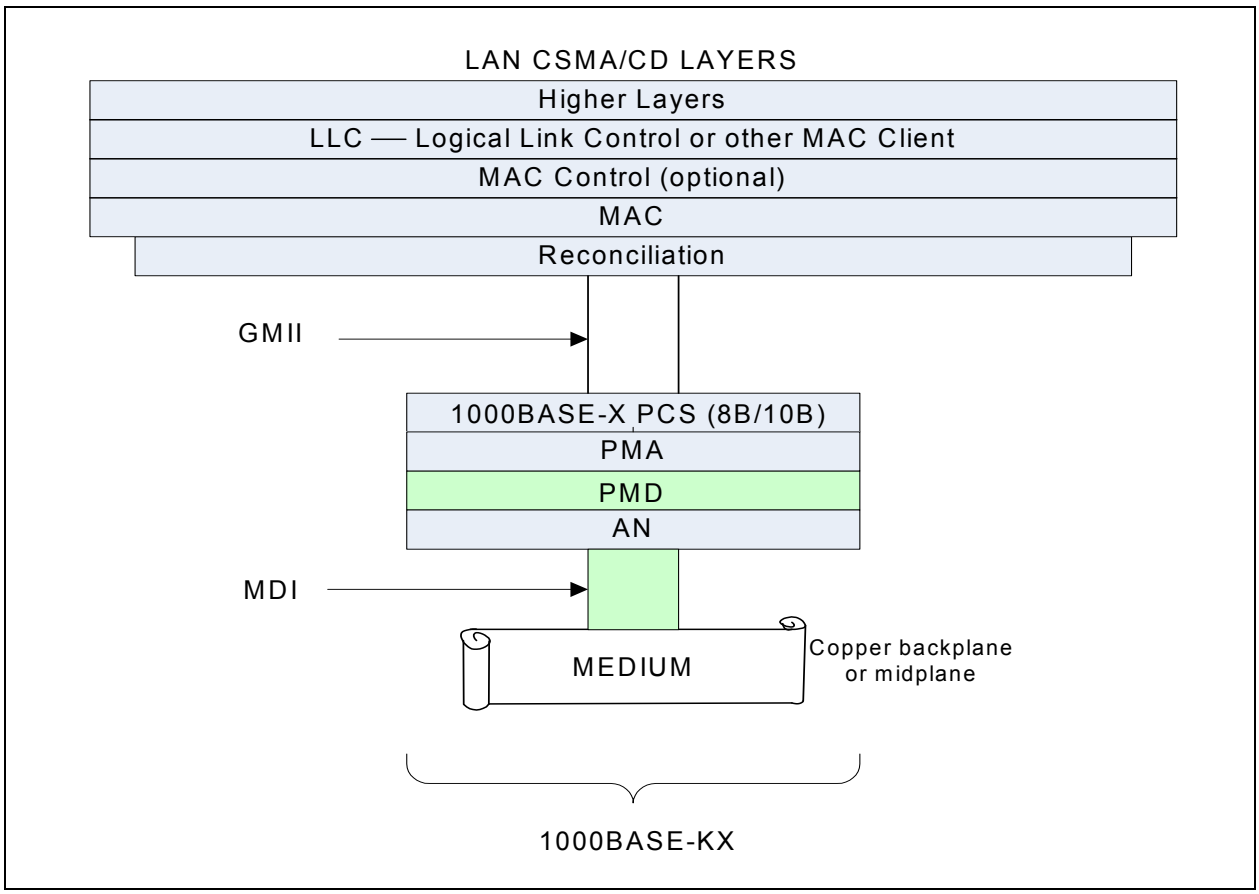


Figure 3-15. Architectural positioning of 1000BASE-KX

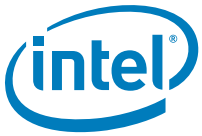
3.2.2.5.1.2 KX electrical characteristics

The KX lane is a low swing AC coupled differential interface using NRZ signaling. AC coupling allows for inter-operability between components operating from at different supply voltages. Low swing differential signaling provides noise immunity and improved reduced EMI. Differential signal swings defined specifications depend on several factors, such as transmitter pre-equalization and transmission line losses.

The KX signal paths are point-to-point connections. Each path corresponds to a KX lane and is comprised of two complementary signals making a balanced differential pair. There is one differential path in each direction for a total of two pairs, or four connections. The signal paths are intended to operate up to approximately one meter over controlled impedance traces on improved FR4 PCBs.

3.2.2.5.2 SGMII support

The XL710 supports 1 Gb/s and 100 Mb/s operation using the SGMII protocol over the KX electrical interface (AC coupling, no source synchronous Tx clock, etc.).



3.2.2.5.2.1 SGMII overview

SGMII interface supported by the XL710 enables operation at 1 Gb/s over two differential, controlled impedance pairs of traces (one pair for transmit, one pair for receive). When operating in SGMII, the MAUI interface uses the 1000BASE-X PCS and PMA as defined in IEEE802.3 Clause 36 and the 1000BASE-KX PMD as defined in IEEE802.3ap Clause 70 or the 1000BASE-BX as defined in the PCIMG 3.1 standard. In SGMII operating mode, the MAUI interface can support data rates of 1 Gb/s and 100 Mb/s. The XL710 also supports IEEE 802.3az EEE operation when configured to operate in SGMII mode (see Section 5.3.1.7.2 for further information).

SGMII, supported by the XL710, is a full-duplex interface that uses a single serial differential link in each direction to achieve 1 Gb/s data throughput. Each serial link operates at 1.25 GBaud to accommodate both data and the overhead associated with 8B/10B coding. The self-clocked nature eliminates skew concerns between clock and data.

SGMII control information, as listed in Table 3-24 is transferred from the PHY to the MAC to signal change of link speed (100 Mb/s or 1 Gb/s). This is achieved by using the auto-negotiation functionality defined in Clause 37 of the IEEE Specification 802.3z. Instead of the ability advertisement, the PHY sends the control information via its tx_config_reg[15:0] as listed in Table 3-24 each time the link speed information changes. Upon receiving control information, the MAC acknowledges the update of the control information by asserting bit 14 of its tx_config_reg[15:0] as listed in Table 3-24. Compared to the definition in IEEE802.3 clause 37, the link_timer inside the auto-negotiation has been changed from 10 ms to 1.6 ms to ensure a prompt update of the link status.

Table 3-24. SGMII link control information

Bit Number	TX_CONFIG_REG[15:0] Sent From PHY to MAC	TX_CONFIG_REG[15:0] Sent From MAC to PHY
15	Link: 1b = link up. 0b = link down.	0b = Reserved for future use.
14	Reserved for auto-negotiation acknowledge as specified in 802.3z	1b.
13	0b: Reserved for future use	0b = Reserved for future use.
12	Duplex mode: 1b = full duplex. 0b = half duplex.	0b = Reserved for future use.
11:10	Speed: Bit 11, 10: 11b = Reserved. 10b = 1000 Mb/s: 1000BASE-TX. 01b = Reserved. 00b = Reserved.	00b = Reserved for future use.
9	EEE: 1b = EEE is supported for 100BASE-TX and 1GBASE-T operation 0b = EEE is not supported.	0b = Reserved for future use.
8:1	0x0 = Reserved for future use.	0x0= Reserved for future use.
0	1b.	1b.

When operating in 100 Mb/s, the SGMII interface elongates the frame by replicating each frame byte 10 times for 100 Mb/s. This frame elongation takes place above the 802.3z PCS layer, thus the start frame delimiter only appears once per frame. Note that the 802.3z PCS layer might remove the first

byte of the elongated frame. An example of a 100 Mb/s elongated frame can be seen in Figure 3-16.

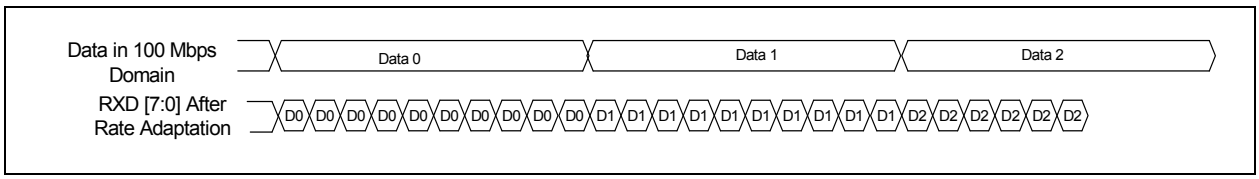


Figure 3-16. Data sampling in 100 Mb/s mode

3.2.2.6 Transceiver module support

The XL710 MAUI interface with additional usage of low speed interface pins (SDP, I^2C and MDIO I/Os) supports connection to transceiver modules compliant with the following Multi Source Agreements (MSAs):

- XENPAK — A cooperation agreement for 10 Gigabit Ethernet Transceiver package Rev 3.0
- X2 — A cooperation agreement for a small Versatile 10 Gigabit Ethernet Transceiver package Rev 2.0b
- XPAK — A cooperation agreement for a small form factor pluggable 10 Gigabit Ethernet Transceiver package Rev 2.2
- SFP+ — SFF-8431 Specifications for Enhanced 8.5 and 10 Gigabit Small Form Factor Pluggable Module SFP+ rev 1.0
- QSFP+ — SFF 8436 Rev 3.4 specifications for quad small form factor pluggable module (used for 40 GbE and 4 x 10 GbE)

Table 3-25 lists the recommended interface (per port) for supporting the various modules. The XL710 supports the high speed interface using the MAUI port and the low speed interface using the SDP pins.

Table 3-25. Optical module interface support

Module Type	High Speed MAUI Protocol	Low Speed Interface (per port)
XENPAK	XAUI	• MDC[1] (1.2V OUT), MDIO[1] (1.2V I/O), • TX ON/OFF[2] (1.2V OUT) • RESET[2] (1.2V OUT) • LASI (1.2V IN — Interrupt)
X2	XAUI	• MDC[1] (1.2V OUT), MDIO[1] (1.2V I/O), • TX ON/OFF[2] (1.2V OUT) • RESET[2] (1.2V OUT) • LASI (1.2V IN — Interrupt)

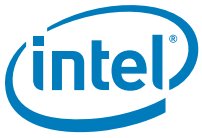


Table 3-25. Optical module interface support

Module Type	High Speed MAUI Protocol	Low Speed Interface (per port)
XPAK	XAUI	• MDC[1] (1.2V OUT), MDIO[1] (1.2V I/O), • TX ON/OFF[2] (1.2V OUT) • RESET[2] (1.2V OUT) • LASI (1.2V IN — Interrupt)
SFP+	SFI	• SCL (I^2C — OD), SDL (I^2C — OD)[3] • MOD_ABS[3] (IN) • TX Disable (LVTTL — OUT) • RS0/1 (LVTTL — OUT) • TX Fault (LVTTL IN) • RX_LOS (LVTTL — IN)
QSFP+	XLAUI / XLPPI / CR4	• SCL[1] (I^2C — OD), SDA[1] (I^2C — OD)[3] • ModSel (LVTTL — IN) • ModPrsL (LVTTL - OUT)[3] • ResetL (LVTTL — IN) • LPMode (LVTTL IN) • IntL (LVTTL — OUT)

1. Single management interface can be used for two ports
2. Output low during reset and power down.
3. These signals are required while the rest are only optional.

The XL710 enables interfacing optical modules using the MAUI pins, MDIO pins, I^2C pins and SDP pins. When interfacing with XENPAK, XPAK and X2 modules, level translators from LVTTL to 1.2V need to be added on the MDIO pins and the relevant SDP pins.

3.2.2.7 Management Data Input/Output (MDIO) interface

The XL710 supports MDIO or I^2C interfaces for control plane connection between the MAC (master side) and external PHY devices. The MDIO interface enables both MAC and firmware access to the PHY for monitor and control of PHY functionality. The XL710 is compliant with the IEEE Std 802.3 Clause 45 in 40 Gb/s, 10 Gb/s and 1 Gb/s operation. The XL710 also supports IEEE Std 802.3 Clause 22 frame formats and register address space for accessing legacy PHY registers. The MDIO interface in the XL710 supports Clause 22 3.3V LVTTL electrical interface. To access PHYs that use Clause 45 specifications (1.2V electrical), level translators need to be added on board.

The XL710 supports up to four MDIO interfaces (one per port) to control external PHYs devices. Depending on the PHY type to manage, each port can be configured for either MDIO interface or 2-wire management interface (I^2C). (see Section 3.2.2.8 for details on I^2C interface).

In order to manage multi-port PHYs, MDIO interface of port 0 can be configured to control a quad port PHY or MDIO interfaces of port 0 and port 1 can be configured for controlling dual port PHYs. The PHY configuration registers for each port mapped into the respective MDIO address space and can be accessed by the MDIO controller firmware.

The firmware performs direct access to MDIO bus for single and multi-port PHY devices. The device driver uses the Admin commands through respective physical functions to manage the PHYs. The firmware provides arbitration of accesses in MFP mode and in SFP mode for shared access to multi-port PHYs.

Figure 3-17 shows the basic connectivity between the MAC and the external PHY.

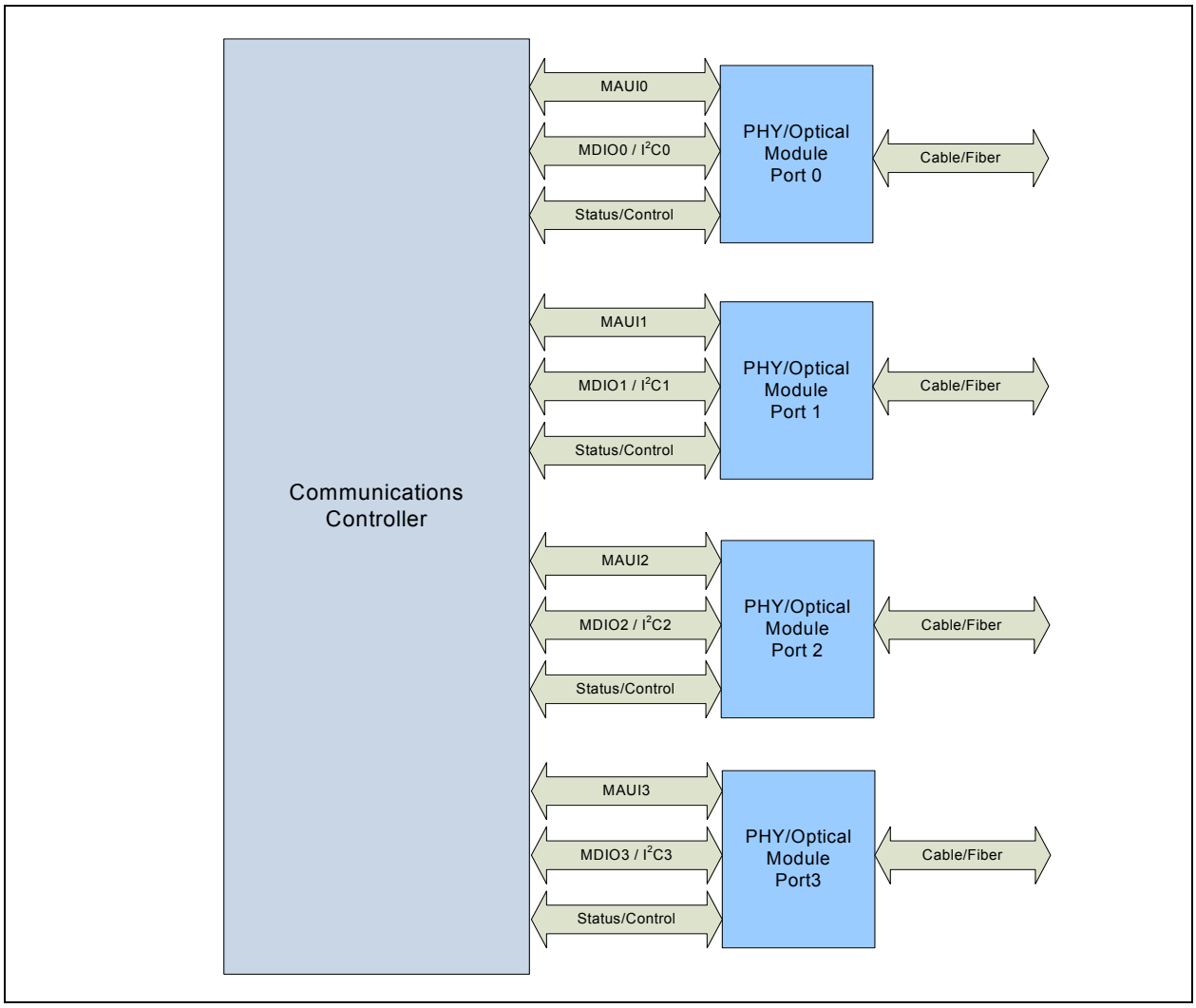


Figure 3-17. Basic PHY MAC connectivity

The MDIO interface is a simple 2-wire serial interface between MAC and PHY and is used to access Control and Status registers inside the PHY. The interface is implemented using two 3.3V I/Os:

1. MDC — MDIO-interface clock signal driven by a MAC (STA) device.
2. MDIO — Read/write data between MAC and PHY.

3.2.2.7.1 MDIO timing relationship to MDC

The MDC clock toggles during a read/write operation at a fixed clock frequency of 2.441 MHz.

MDIO is a bidirectional signal that can be sourced by the Station Management Entity (STA) or the PHY. When the STA sources the MDIO signal, the STA must provide a minimum of 10 ns of setup time and a minimum of 10 ns of hold time referenced to the rising edge of MDC, as shown in Figure 3-18 (measured at the MII connector).

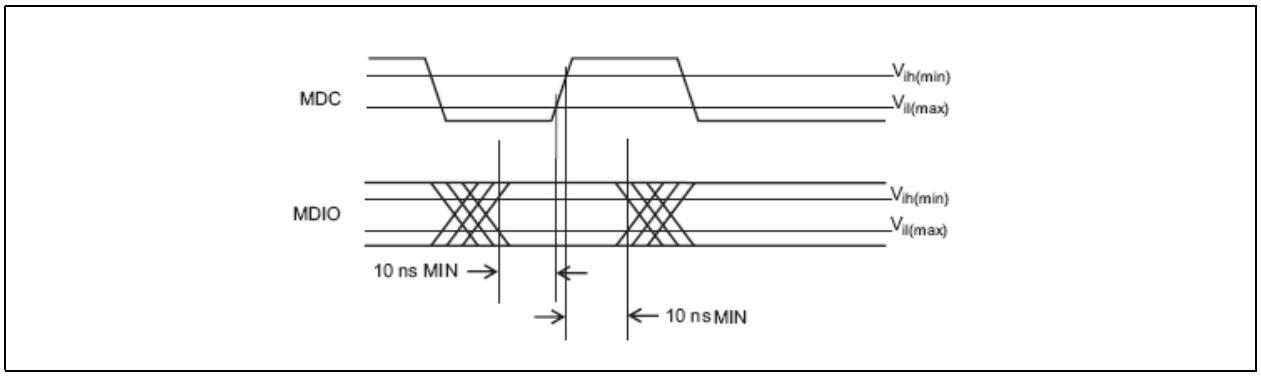


Figure 3-18. MDIO timing sourced by the MAC

When the MDIO signal is sourced by the PHY, it is sampled by the MAC (STA) synchronously with respect to the rising edge of MDC. The clock to output delay from the PHY, as measured at the MII connector, must be a minimum of 0 ns, and a maximum of 300 ns, as shown in Figure 3-19.

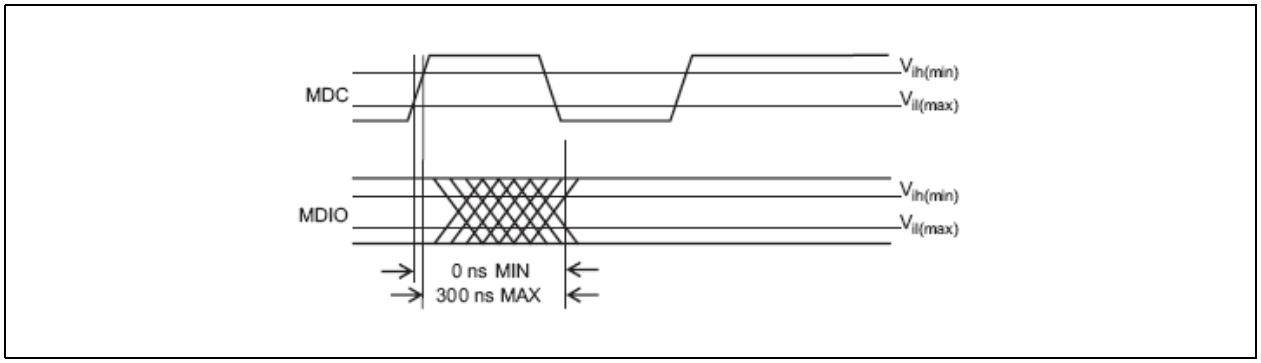


Figure 3-19. MDIO timing sourced by the PHY

3.2.2.7.2 IEEE Std 802.3 Clause 22 and Clause 45 differences

IEEE Std 802.3 Clause 45 provides the ability to access additional device registers while still retaining logical compatibility with interface defined in Clause 22. Clause 22 specifies the MDIO frame format and uses an ST code of 01 to access registers. In Clause 45, additional registers are added to the address space by defining MDIO frames that use a ST code of 00.

Clause 45 (MDIO interface) major concepts:

a. Preserve management frame structure defined in IEEE 802.3 Clause 22.
b. Define mechanism to address more registers than specified in IEEE802.3 Clause 22.

c. Define ST and OP codes to identify and control the extended access functions.

3.2.2.7.3 MDIO management frame structure

The MDIO interface frame structure defined in IEEE802.3 Clause 22 and Clause 45 are compatible so that the two systems supporting different formats can co-exist on the same MDIO bus. The XL710 supports both frame structures to enable interfacing PHYs that support either protocol.

The basic frame format as defined in IEEE802.3 Clause 22 can optionally be used for accessing legacy PHY registers is listed in Table 3-26.

Table 3-26. Clause 22 basic MDIO frame format

	Management Frame Fields							
Frame	**Pre**	**ST**	**OP**	**PRTAD**	**REGAD**	**TA**	**Data**	**Idle**
Read	1...1	01	10	PPPPP	RRRRR	Z0	DDDDDDDDDDDDDDDD	Z
Write	1...1	01	01	PPPPP	RRRRR	10	DDDDDDDDDDDDDDDD	Z

The MDIO interface defined in Clause 45 uses indirect addressing to create an extended address space enabling access to a large number of registers within each MDIO Managed Device (MMD). The MDIO management frame format is listed in Table 3-27.

Table 3-27. Clause 45 indirect addressing MDIO frame format

	Management Frame Fields							
Frame	**Pre**	**ST**	**OP**	**PRTAD**	**DEVAD**	**TA**	**Address / Data**	**Idle**
Address	1...1	00	00	PPPPP	EEEEE	10	AAAAAAAAAAAAAAAA	Z
Write	1...1	00	01	PPPPP	EEEEE	10	DDDDDDDDDDDDDDDD	Z
Read	1...1	00	11	PPPPP	EEEEE	Z0	DDDDDDDDDDDDDDDD	Z
Post-Read Increment Address	1...1	00	10	PPPPP	EEEEE	Z0	DDDDDDDDDDDDDDDD	Z

To support clause 45 indirect addressing each MMD (PHY — MDIO managed device) implements a 16-bit address register that stores the address of the register to be accessed by data transaction frames. The address register must be overwritten by address frames. At power up or device reset, the contents of the address register are undefined. Write, read, and post-read-increment-address frames must access the register whose address is stored in the address register. Write and read frames must not modify the contents of the address register. Upon receiving a post-read-increment-address frame and having completed the read operation, the MMD increments the Address register by one (up to a value of 0xFFFF). Each MMD supported implements a separate address register, so that the MMD's address registers operate independently of one another.

Idle Condition (IDLE) — The IDLE condition on MDIO is a high-impedance state. All three state drivers must be disabled and the PHY's pull-up resistor pulls the MDIO line to a logic one.

Preamble (PRE) — At the beginning of each transaction, the station management entity must send a sequence of 32 contiguous consecutive one bits on MDIO with 32 corresponding cycles on MDC to provide the PHY with a pattern that it can use to establish synchronization. A PHY must observe a sequence of 32 contiguous consecutive one bits on MDIO with 32 corresponding cycles on MDC before it responds to any transaction.

Start of Frame (ST) — The ST is indicated by:

- <00> pattern for clause 45 compatible frames for indirect access cycles.
- <01> pattern for clause 22 compatible frames for direct access cycles.

These patterns ensure a transition from the default value of one on the MDIO signal, and identifies the start of frame.

Operation Code (OP) — The *OP* field indicates the type of transaction being performed by the frame.

For Clause 45 compatible frames:

- A <00> pattern indicates that the frame payload contains the address of the register to access.
- A <01> pattern indicates that the frame payload contains data to be written to the register whose address was provided in the previous address frame.
- A <11> pattern indicates that the frame is an indirect read operation.
- A <10> pattern indicates that the frame is an indirect post-read-increment-address operation.

For Clause 22 compatible frames:

- A <10> pattern indicates a direct read transaction from a register.
- A <01> pattern indicates a direct write transaction to a register.

Port Address (PRTAD) — The PRTAD is five bits, allowing 32 unique PHY port addresses. The first *PRTAD* bit to be transmitted and received is the MSB of the address. A station management entity must have prior knowledge of the appropriate port address for each port to which it is attached, whether connected to a single port or to multiple ports.

Device Address (DEVAD) — The DEVAD is five bits, allowing 32 unique MMDs per port. The first *DEVAD* bit transmitted and received is the MSB of the address. This field is relevant only in clause 45 compatible frames (ST=<00>).

Register Address (REGAD) — The REGAD is five bits, allowing 32 individual registers to be addressed within each PHY. The first *REGAD* bit transmitted and received is the MSB of the address. This field is relevant only in clause 22 compatible frames (ST=<01>).

Turnaround (TA) — The TA time is a 2-bit time spacing between the *DEVAD* field and the *Data* field of a management frame. This is to avoid contention during a read transaction. For a read or post-read-increment-address transaction, both the STA and the PHY must remain in a high-impedance state for the first bit time of the TA. The PHY must drive a zero bit during the second bit time of the TA of a read or post read-increment-address transaction. During a write or address transaction, the STA must drive a one bit for the first bit time of the TA and a zero bit for the second bit time of the TA. Figure 3-20 shows the behavior of the MDIO signal during the *TA* field of a read transaction.

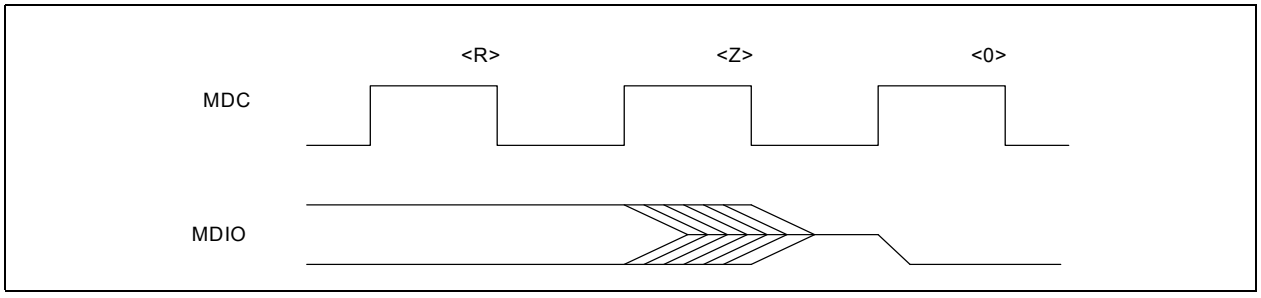


Figure 3-20. Behavior of MDIO during TA field of a read transaction

- Clause 45 compatible frames have 16-bit address/data fields. For an auto-negation address cycle, it contains the address of the register to be accessed on the next cycle. For the data cycle of a write frame, the field contains the data to be written to the register. For a read or post-read-increment-address frame, the field contains the contents of the register. The first bit transmitted and received must be bit 15.
- Clause 22 compatible frames have 16-bit data fields. The first data bit transmitted and received must be bit 15 of the register being addressed.

3.2.2.7.4 MDIO direct access

The software device driver manages the PHYs using the Admin commands (See Section 3.2.4) in both SFP mode and MFP modes of operation. The device driver does not have direct access to the MDIO bus except for diagnostic purposes. Firmware performs direct access to the MDIO interface for reading and writing to the PHYs as described in the following paragraphs.

The MDI is accessed through registers MSCA and MSRWD. A single management frame is sent by setting bit MSCA.MDICMD to 1b after programming the appropriate fields in the MSCA and MSRWD registers. The MSCA.MDICMD bit is auto cleared after the read or write transaction completes. To execute Clause 22 format write operations, the following steps should be done:

1. Data to be written is programmed in field MSRWD.MDIWRDATA.
2. Register MSCA is initialized with the appropriate control information (start, code, etc.) with bit MSCA.MDICMD set to 1b.
3. Wait for bit MSCA.MDICMD to reset to 0b when indicating that the transaction on the MDIO interface is complete.

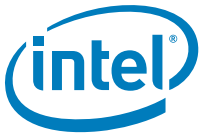
The steps for Clause 22 format read operations are identical to the write operation except that the data in field MSRWD.MDIWRDATA is ignored and the data read from the external device is stored in register field MSRWD.MDIRDDATA bits. Clause 45 format read/write operations must be performed in two steps. The address portion of the pair of frames is sent by setting register field MSCA.MDIADD to the desired address, field MSCA.STCODE to 00b (start code that identifies Clause 45 format), and register field MSCA.OPCODE to 00b (Clause 45 address register write operation). A second data frame must be sent after the address frame completes. This second frame executes the write or read operation to the address specified in the PHY address register.

3.2.2.8 2-wire I^2C management interface

The XL710 supports a 2-wire management interface (I^2C) for connectivity to external PHYs. Typically the SFP+ or QSFP+ optical and direct attached copper PHYs use the SFI specified 2-wire management interface (I^2C) as described in SFF8431 (SFP+), SFF8436 (QSFP+) or SFF8636 (Direct attach Cu).

The XL710 supports up to four I^2C interfaces (one per port) to control external PHY devices. Depending on the PHY type to manage, each port can be configured for either 2-wire management interface (I^2C) or MDIO interface (see Section 3.2.2.7 for details on MDIO interface). The 2-wire interface bus or the MDIO bus are connected via the same pins, and thus are mutually exclusive. The configuration of whether the PHY is controlled via MDIO management interface or I^2C management interface is selected through the control bits in the NVM.

The XL710 provides hardware acceleration of I^2C accesses over the 2-wire management interface. The device driver manages the SFP+/QSFP+ modules using the Admin commands (see Section 3.2.4) in both SFP mode and MFP modes of operation. The device driver does not have direct access to the I^2C bus except for diagnostic purposes. Firmware performs direct access to I^2C interface for reading and writing to the PHY modules as described in the paragraphs that follow.



The I^2C bus is accessed through registers GLGEN_I2CCMD and GLGEN_I2CPARAMS. I^2C accesses through the hardware controller can be performed through the GLGEN_I2CCMD register. I^2C accesses can also be performed using bit banging through GLGEN_I2CPARAMS register if needed. Firmware can execute I^2C write/read operations by writing to the I^2C command register. The status of I^2C cycle completion can be performed by reading the GLGEN_I2CCMD register. Refer to the steps that follow to appropriately set the fields:

1. I^2C register address is placed in GLGEN_I2CCMD.DEVADD field.
2. PHY address is placed in GLGEN_I2CCMD.PHYADD field (typically this is 0xA0 or 0xA2 for modules).
3. Command for read or write operation is placed in the GLGEN_I2CCMD.OP field.
4. For a write operation, data to be written is placed in the GLGEN_I2CCMD.DATA field.
5. Successful completion of a write or read operation is indicated by the XL710 through *Ready* bit (GLGEN_I2CCMD.R). A read error is indicated if GLGEN_I2CCMD.E bit is set along with GLGEN_I2CCMD.R.
6. For a read operation, data can be read from GLGEN_I2CCMD.DATA field when the *Ready* (GLGEN_I2CCMD.R) bit is set.
7. A reset sequence can be sent to the I^2C bus before the actual write/read operation when GLGEN_I2CCMD.Reset bit is set.

3.2.2.9 SDPs for PHY management

The XL710 has a total of 22 GPIO pins that can be configured as SDPs or other dedicated hardware functions for connecting to external PHYs. The XL710 has designated 16 GPIO pins as SDPs for use with ports 0 through 3, as listed in Section 2.2.6.2. However, depending on the board layout, system designers have the flexibility to configure any of the GPIO pins, irrespective of their name, as SDPs. This section describes the use of SDP pins for the management of external PHYs and optical/copper modules. See Section 3.4.2 for details on configuring the pin function, direction, and values of SDP pins.

Firmware controls the SDP pins when used for specific hardware functions for PHY management. The device driver uses Admin commands (see Section 3.2.4) to configure and manage the PHYs; the device driver does not directly access the SDP pins except for diagnostic purposes. In MFP mode, the firmware arbitrates the access to PHY configuration and status including the hardware functions provided by the SDP pins.

Table 3-28 lists the recommended configurations for connecting the XL710 SDPs, MDIO/I^2C pins to external PHYs or optical/copper modules. The configurations include connectivity to QSFP+ modules, SFP+ modules and 10GBASE-T PHYs, in single, dual or quad port configurations. Table 3-29 lists an example of a dual port 10 GbE PHY configuration in 82599 compatible mode. If mapping of these SDP pins to a specific hardware function is not required then the pins can be used as GPIOs (see Section 3.4.3). Figure 3-21 shows an example of the XL710 SDP connections to an SFP+ optical module.

Any of the SDP pins configured as an input for dedicated hardware functions can be configured to cause an interrupt to the firmware (See Section 3.4.2). The firmware in turn reads the appropriate PHY status registers (through MDC or I^2C interfaces) and might post an event and provide necessary status information to the device driver through Admin commands (see Section 3.2.4).

Table 3-28. Example for SDP, MDIO and I²C ports usage for SFP+ and QSFP+ modules

Port Num	Pin Name	GPIOn Index	SFP+	QSFP+	10GBASE-T Copper PHY
0	SDP0_0	GPIO4	Rx_LOS	IntL	PhyInt
	SDP0_1	GPIO5	Tx_Fault	ResetL	PHYRst
	SDP0_2	GPIO6	Mod_ABS	ModPrsL	TxDisable
	SDP0_3	GPIO7	RS0/RS1[1]	LPMode	
	SDP2_0[2]	GPIO12		ModSelL[2][3]	
	MDIO0_SDA0 MDC0_SCL0		SDA SCL	SDA SCL	MDIO[4] MDC
1	SDP1_0	GPIO8	Rx_LOS	IntL	PhyInt[4]
	SDP1_1	GPIO9	Tx_Fault	ResetL	PHYRst[5]
	SDP1_2	GPIO10	Mod_ABS	ModPrsL	TxDisable[4]
	SDP1_3	GPIO11	RS0/RS1	LPMode	
	SDP_2_1[2]	GPIO13		ModSelL[2] [3]	
	MDIO1_SDA1 MDC1_SCL1		SDA SCL	SDA SCL	MDIO MDC
2	SDP2_0	GPIO12	Rx_LOS		PhyInt
	SDP2_1	GPIO13	Tx_Fault		PHYRst
	SDP2_2	GPIO14	Mod_ABS		TxDisable
	SDP2_3	GPIO15	RS0/RS1		
	MDIO2_SDA2 MDC2_SCL2		SDA SCL		MDIO MDC
3	SDP3_0	GPIO16	Rx_LOS		PhyInt
	SDP3_1	GPIO17	Tx_Fault		PHYRst
	SDP3_2	GPIO18	Mod_ABS		TxDisable
	SDP3_3	GPIO19	RS0/RS1		
	MDIO3_SDA3 MDC3_SCL3		SDA SCL		MDIO MDC

1. Typical SFP+ modules provide software or hardware control of TX_Disable and Rate Select (RS0/RS1) inputs. If software control is used, the TX_Disable and RS0/RS1 bits can be accessed through the I²C interface. If hardware control is used for rate selection, inputs RS0 and RS1 are tied together and connected to the same SDP output.
2. ModSelL is module select pin, acts like chip select for a shared I²C interface. In 40 Gb/s mode with QSFP+ module option, SDP2_0 is configured to be associated with Port 0 and SDP2_1 is configured to be associated with port 1
3. Although ModSelL connectivity is supported by the XL710 it is not recommended to share the management interface between two QSFP+ modules.
4. In multi-port PHYs, one MDIO interface can be shared to manage multiple ports
5. In multi-port PHYs, for example, there might not be separate pins per port for PHY reset, PHY interrupt, or Tx disable; in such cases, software/firmware might use the MDIO interface to access the PHY control/status registers of multiple portsThe XL710 needs to be configured as per the actual system configuration.

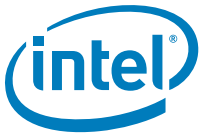


Table 3-29. Example for SDP, MDIO and I²C ports usage for 82599 compatible mode

Port Num	Pin Name	GPIO Index	SFP+	Reserved	Copper PHY	Reserved
0	SDP0_0	GPIO4	RX_LOS		INTR_L	
	SDP0_1	GPIO5	RX_LOS_N			
	SDP0_2	GPIO6	MOD_ABS_N			
	SDP0_3	GPIO7	TX_DISABLE			
	SDP2_1	GPIO13	RS0/RS1 drive		RESET	
	SDP2_2	GPIO14	RS0/RS1 sense			
	SDP2_3	GPIO15	TX_FAULT			
	MDIO0_SDA0 MDC0_SCL0		SDA SCL		MDIO MDC	
1	SDP1_0	GPIO8	RX_LOS		INTR_L	
	SDP1_1	GPIO9	RX_LOS_N			
	SDP1_2	GPIO10	MOD_ABS_N			
	SDP1_3	GPIO11	TX_DISABLE			
	SDP3_1	GPIO17	RS0/RS1 drive		RESET	
	SDP3_2	GPIO18	RS0/RS1 sense			
	SDP3_3	GPIO19	TX_FAULT			
	MDIO1_SDA1 MDC1_SCL1		SDA SCL		MDIO MDC	

Table 3-30 lists the signals defined in Table 3-28 and Table 3-29, their GPIO pin configuration, and behavior during reset and power down state (D3) without management.

Table 3-30. SDP assigned signals/pin configuration

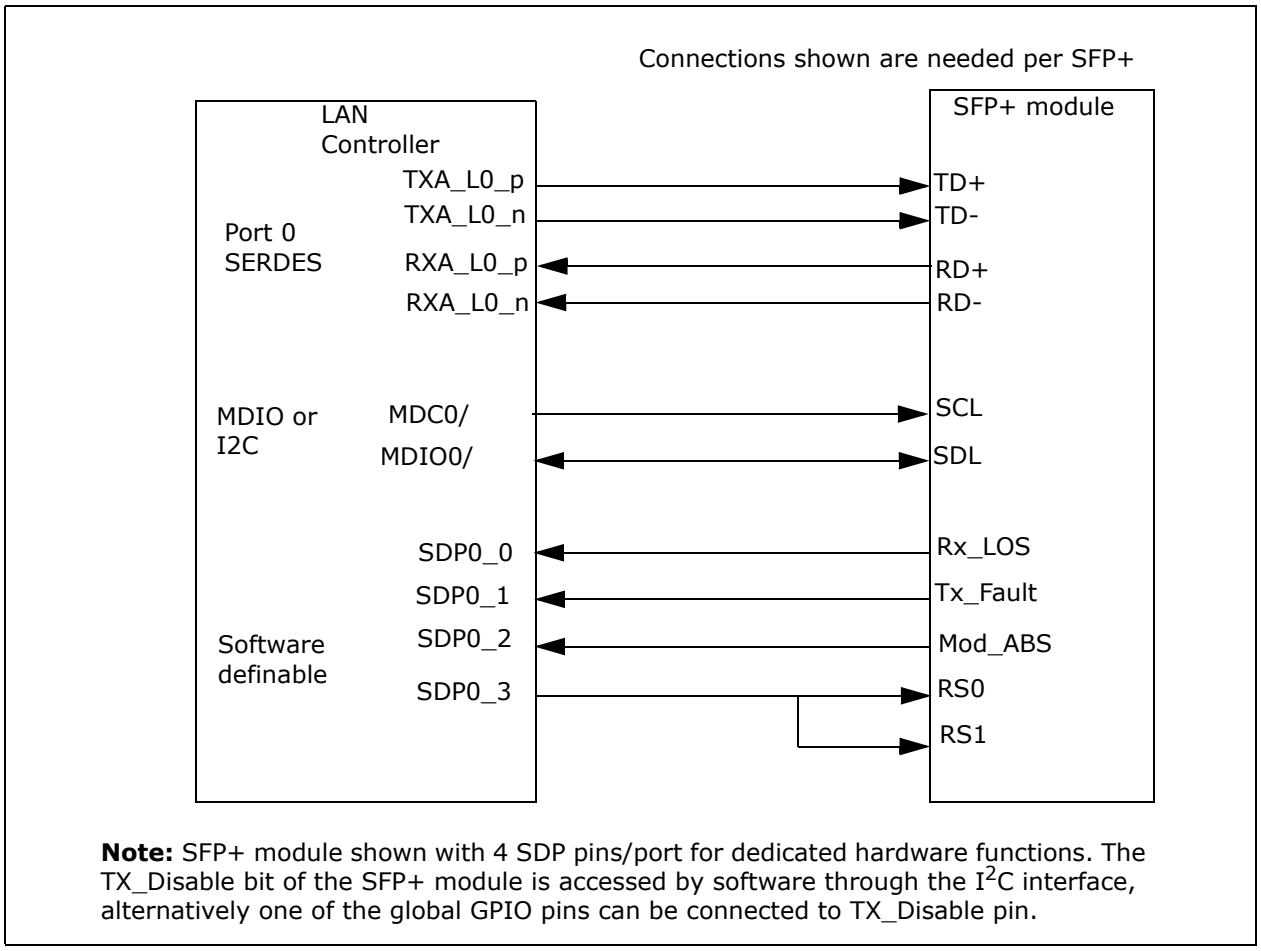
Signal	Description	GPIO Pin Configuration	Default Values at (Reset, D3 no WoL and no MNG)[1]
RX_LOS, RX_LOS_N	RX_LOS high and RX_LOS_N low indicate insufficient optical power for reliable signal reception.	SDP: Input (Interrupt)	Input, no change.
LASI, INTR_L	INTR_L or Link Alarm Status Interrupt (LASI) — when low, indicates possible module operational fault or a status critical to the host system.	SDP: Input (Interrupt)	Input, no change.
RS0/RS1 drive	Short-circuit protected.	SDP: Output	Output, autonomous high or tri-state with pull-up.
RS0/RS1 sense	Directly connected input.	SDP: Input	Input, no change.
MOD_ABS	When low, indicates SFP+ module is present; when high, indicates that the module is absent	SDP: Input (Interrupt)	Input, no change.
TX_DISABLE	When TX_DISABLE is asserted high, optical module transmitter is turned off.	SDP: Output	Output, no change. In order to minimize PHY power software should drive the SDP to high or set to input while populating a pull-up.
TX_FAULT	When high, indicates that the module transmitter has detected a fault condition related to laser operation or safety.	SDP: Input (Interrupt)	Input, no change.

Table 3-30. SDP assigned signals/pin configuration

Signal	Description	GPIO Pin Configuration	Default Values at (Reset, D3 no WoL and no MNG)[1]
RESET_N	When low, XENPAK, X2 or XPAK optical module is reset.	SDP: Output	Output, no change. In order to minimize PHY power software should drive the SDP to low or set to input while populating a pull-down.
RESET	When high, the copper PHY is reset.	SDP: Output	Output, autonomous high or tri-state with pull-up.
TX_DIS	When TX_DIS is asserted high, optical module transmitter is turned off.	SDP: Output	Output, autonomous high or tri-state with pull-up.
TX ON/OFF	1b = Transmitter on. 0b = Transmitter off.	SDP: Output	Output, autonomous low or tri-state with pull-down.
MOD_DET_N	Inverted mode detect.	SDP: Input (Interrupt)	Input, no change.
TS_SDPX	Time sync support pins, can be used as event in or event out.	TBD: According to programmed functionality	Tri-state during reset. No change in D3. External pull-up / pull-down as required by the system designer.
MOD_NR	When high, indicates that the module has detected a condition that renders transmitter and or receiver data invalid.	SDP: Input (Interrupt)	Input, no change.
IntL	When Low, QSFP+ module interrupt. Read status of interrupt over I^2C	SDP: Input (Interrupt)	Input, no change
ModPrsL	When Low, indicates QSFP+ module is present; when high, indicates that the module is absent	SDP: Input (Interrupt)	Input, no change
ResetL	When Low, resets the QSFP+ module to default	SDP: Output	Output, line is pulled high in module.
LPMode	Controls the QSFP+ module power modes	SDP: Output	Output, line is pulled high in module
ModSelL	When Low, I^2C access to module is enabled, else I^2C access is disabled.	SDP: Output	Output, PHY software to drive the line low for access through I^2C interface.
FAN_Status	Optional health indication of the fan.	SDP: Input (Interrupt)	Input, no change.

1. Use GLGEN_GPIO_CTL.OUT_CTL to control the default output (tri-state or driven high) during reset or D3 power state.

3.2.2.9.1 XL710's SFP+ connectivity scheme



Note: SFP+ module shown with 4 SDP pins/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.

Figure 3-21. XL710 SDP connections to SFP+ module

3.2.3 Link configuration

The XL710 network interface meets industry specifications for:

- 40 Gb/s:
  - XLAUI (IEEE Std 802.3ba-2010, Annex 83A, 83B)
  - XLPPI (IEEE Std 802.3ba-2010 Annex 86A)
  - 40GBASE-KR4 (IEEE Std 802.3ba-2010, 40 Gb/s backplane Ethernet specification)
  - 40GBASE-CR4 (IEEE Std 802.3ba-2010, 40 Gb/s twinaxial copper cable assembly specification)
- 10 Gb/s:
  - XAUI (IEEE 802.3ae and IEEE 802.3az)
  - SFI (SFF-8431 Specifications for Enhanced 8.5 and 10 Gigabit Small Form Factor Pluggable Module SFP+)
- 10 Gb/s backplane:
  - Ethernet 10GBASE-KX4 (IEEE 802.3ap and IEEE 802.3az)

  - — Ethernet 10GBASE-KR (IEEE 802.3ap and IEEE 802.3az)
- 1 Gb/s backplane:
  - — Ethernet 1000BASE-KX (IEEE 802.3ap and IEEE 802.3az)
  - — SFI (SFF-8431 Specifications for Enhanced 8.5 and 10 Gigabit Small Form Factor Pluggable Module SFP+)

The MAU interface and the respective SerDes/AFE is configured at start up to support the appropriate protocol as a function of the negotiation process and pre-defined control bits that are either loaded from the NVM or configured by firmware.

3.2.3.1 Firmware link management

In the XL710 all link configurations are performed by firmware. At POR firmware initializes the link and then provides a set of Admin commands, described in Section 3.2.4, to allow for software device drivers' modifications of the link parameters and configurations. Since the XL710 supports both Single Function Mode (SFP) and Multi-Function Mode (MFP) all access to the link configurations is performed using the Admin commands and device drivers do not directly access the MAC, PHY or SDP settings. This is required to prevent conflicting settings in MFP mode and is also used in SFP mode in order to provide the software device driver with an API like interface to the MAC, PHY and SDP registers.

Link initialization by firmware includes reading the initial configuration from the NVM, reading the PHY type connected to the link and programming appropriate values to the MAC and PHY registers to bring up the link. See the detailed flow in Section 3.2.3.1.3 for more information.

The firmware also provides services to the device driver by providing status of the link, modifying the link configuration, and re-initializing the link if requested by the software device driver.

3.2.3.1.1 Link management in SFP mode

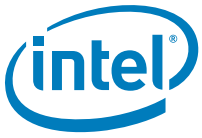
At power up, firmware initializes the link based on the initial configurations loaded from the NVM. Following the initial setup, the software device driver might override any of the link settings; however, even though the software device driver is the owner of the link in SFP mode, it does not directly access the MAC or PHY registers and SDP pins controlling the PHYs instead all access to the link settings is performed through firmware using the Link Configuration Admin Commands.

3.2.3.1.2 MAC link setup and auto-negotiation

Link speed and link characteristics can be determined through static configuration, parallel detection, auto-negotiation or forced operation for diagnostic purposes. The auto-negotiation processes defined in IEEE802.3ap Clause 73 enables selection between CR4 (40 Gb/s Twinax copper) KR4 (40 Gb/s), KR (10 Gb/s), KX4 (10 Gb/s), KX (1 Gb/s) compliant link partners and defining link characteristics and link speed. Note that the auto-negotiation process defined in IEEE802.3 Clause 37 enables detection of the legacy 1 Gb/s link characteristics but not the link speed.

3.2.3.1.2.1 Auto-negotiation for backplane Ethernet and direct attached copper cable assembly (IEEE 802.3 Clause 73)

Auto-negotiation provides the link partners the capability to advertise and detect the abilities (modes of operation) supported by the XL710 at the other end of the link, determine common abilities, and configure for joint operation.



Auto-negotiation for the backplane Ethernet and 40 Gb/s direct attach copper cable assembly is specified by IEEE 802.3, Clause 73. Auto-negotiation for backplane Ethernet uses 48-bit base page and next page format and modifies the timers to allow rapid convergence. Clause 73 auto-negotiation uses Differential Manchester Encoding (DME) signaling, which is suitable for electrical backplanes and twinaxial copper cable assemblies, since DME provides a DC balanced signal.

Auto-negotiation defined in IEEE Clause 73 enables automatic link detection and setup for backplane PHYs of: 40GBASE-KR4, 10GBASE-KR, KX4 and KR; it also enables twinaxial copper cable PHYs detection of 40GBASE-CR4. Auto-negotiation includes support for parallel detection of 1000BASE-KX and 10GBASE-KX4 in addition to transmission and reception of 48-bit extended base page and next page auto-negotiation frames. The XL710 supports transmission and reception of extended base page and next page auto-negotiation frames. The XL710 also supports IEEE 802.3 EEE backplane auto-negotiation (see Section 5.3.1 for further information).

3.2.3.1.2.2 Hardware detection of legacy link partner (parallel detection)

The XL710 supports the IEEE 802.3 Clause 73 parallel detection process to enable a connection to legacy link partners that do not support (or disabled) auto-negotiation. Parallel detection enables detecting the link partner operating mode by activating KX4 and KX alternately and attempting to achieve link synchronization by the related PCS block.

3.2.3.1.3 MAUI link setup flow

The XL710 MAUI interface is configured at start up by firmware (before the device driver is loaded) in the following manner:

1. Firmware reads default settings from the NVM per port.
2. Firmware reads GPIO, MDC/I2C settings pre-loaded by hardware from the NVM.
3. If link is setup for backplane connectivity KX/KX4/KR/KR4 with auto-negotiation:

Note: Internal PHY/PCS settings are pre-loaded from the NVM.

   - Firmware waits for link to be established.
   - Based on resulting speed and parameters (EEE, FC) firmware configures other blocks in the XL710.
   - Firmware generates a Link Status Event (LSE) if enabled.

4. If link is setup for SFP+ or QSFP+ module connectivity (SFI or XLAUI or XLPPI or 40GBASE-CR4):
   - Firmware verifies that the connected module is a qualified module (when enabled)
   - Based on module type, optical or direct attach, firmware configures the PCS registers.
   - Firmware restarts the link setup.
   - When internal link is established, firmware generates a LSE if enabled.
5. If link is setup for BASE-T PHY connectivity (external PHY):
   - Firmware verifies that the connected PHY is valid.
   - Firmware programs external PHY's registers
   - IF auto-negotiation is enabled, firmware restarts the auto-negotiation process on the BASE-T interface.
   - When link is established, firmware programs the internal MAC-PHY interface according to the negotiated speed and parameters (EEE, FC)
   - Firmware generates a LSE if enabled.

Note: If link synchronization is not successful, firmware does not report link-up and continuously polls link indications while waiting for link to be established.

If module or PHY qualification is enabled and connected device is not found in the qualified list, firmware stops the link setup process and reports event to software.

The supported opcode are:

- Set register bits ON
- Set register bits OFF
- Modify register bits
- Update register
- Verify register value
- Test register bits ON
- Test register bits OFF
- Wait for register value
- Wait for register bits ON
- Wait for register bits OFF
- Set PHY control pin state
- Wait for PHY control pin state
- General wait
- Code end

For more details see the firmware documentation.

3.2.3.1.4 Next Page (NP) support

NP support in the XL710 is compliant with IEEE Clause 73.

The XL710 acts as receiver of NP each time the link partner needs to transmit NP data through the KX/KX4/KR4/CR4 auto-negotiation process.

The XL710 supports transmission of configurable NP. It transmits a NP each time the auto-negotiation arbitration state machine is required to go through the NP handshake.

3.2.3.1.5 Forcing link up

Forcing link up can be accomplished by setting the *AN Mode* field in the Set PHY config Admin command. This forces the MAC to the appropriate link speed and interface mode as defined in *Link Speed* and *PHY Type* fields in the Set PHY config Admin command.The Force Link Up mode enables loopback operation by setting the *Loopback Mode* field in the Set loopback modes Admin command.

3.2.4 Link Configuration Admin Commands

The XL710 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 actual status of the link.

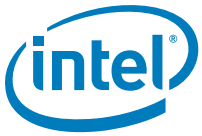


Table 3-31. Link configuration admin commands (0x06xx)

Command	Opcode	Description	Detailed Description
Set PHY Config	0x0601	Set various PHY configuration parameters on port.	Section 3.2.4.1.1
Set MAC Config	0x603	Set various MAC configuration parameters on the port.	Section 3.2.4.1.2
Setup Link and Restart[1] AN	0x0605	Sets up the link and restarts link auto-negotiation. This operation could bring down the link. This command needs to be executed for other set link parameters to take effect on the link.	Section 3.2.4.1.3
Get PHY abilities	0x0600	Get various PHY abilities supported on the port.	Section 3.2.4.1.4
Get Link Status	0x0607	Get link status of the port.	Section 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).	Section 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.	Section 3.2.4.1.7
The following Admin commands can be used by software for diagnostic and maintenance purposes. Typically these commands are not needed for normal runtime operations.			
Set Loopback modes	0x0618	Sets various MAC/PHY loopback modes of the link.	Section 3.2.4.1.8

1. In SFP mode, the Setup link and restart auto-negotiation command needs to be executed by the device driver in order for any other change in link parameters to take effect on the link. This operation could disrupt the link since the link state may 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 its structure.

3.2.4.1.1 Set PHY config

This command is used by the device driver to set the various PHY configuration parameters supported on the port. One or more of the set PHY config parameters can be ignored in MFP mode because the PF might not have the privilege to set some of the PHY config parameters. This status is indicated by the command response.

This is a Direct command. The set PHY command parameters data structure is placed in the descriptor.

Note: This command must be followed by the Setup link and restart auto-negotiation command in order for any changes to the link parameters to actually take place.

Table 3-32. Set PHY config command (opcode: 0x0601)

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	Command opcode.
Datalen	4-5	0x0	Must be 0b, value is ignored.
Return value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.

Table 3-32. Set PHY config command (opcode: 0x0601)

Name	Bytes.Bits	Value	Remarks
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Set PHY Config	16-31	See Table 3-33	16-byte data structure that holds the set PHY config command parameters listed in Table 3-33.

Table 3-33 lists the data structure of the set PHY config command parameters such as PHY type, PHY ID, speed ability, pause ability, etc.

Table 3-33. Set PHY config command data structure

Name	Bytes.Bits	Value	MFP Mode	Remarks
PHY Type	0-3	PHY type	BLOCKED. Cannot be modified in MFP mode.	PHY type supported on port. One bit per PHY type. The XL710 might be capable of supporting multiple PHY types. The following parameter indicates the bit number. 00 = SGMII 01 = 1000BASE-KX. 02 = 10GBASE-KX4. 03 = 10GBASE-KR. 04 = 40GBASE-KR4. 05 = XAUI. 07 = SFI. 08 = XLAUI. 09 = XLPPI. 10 = 40GBASE-CR4 (used with 40 Gb/s QSFP+ direct attach copper). 11 = 10GBASE-CR1 (used with 4x10 Gb/s QSFP+ direct attach copper). 17 = 100BASE-TX. 18 = 1000BASE-T. 19 = 10GBASE-T.
				20 = 10GBASE-SR (10 Gb/s SFP+ SR optical module). 21 = 10GBASE-LR (10 Gb/s SFP+ LR optical module). 22 = 10GBASE-SFP+Cu (direct attach copper). 23 = 10GBASE-CR1 (4x10 Gb/s QSFP+ CR over direct attach copper). 24 = 40GBASE-CR4 (40 Gb/s QSFP+ CR over direct attach copper). 25 = 40GBASE-SR4 (40 Gb/s QSFP+ SR optical module). 26 = 40GBASE-LR4 (40 Gb/s QSFP+ LR optical module). 30Other bits - Reserved, Must be zero.

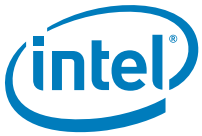


Table 3-33. Set PHY config command data structure (Continued)

Name	Bytes.Bits	Value	MFP Mode	Remarks
				This parameter is used by the device driver to set the various PHY types' configuration parameters to be supported on the port. The port can be configured for a subset of the actual PHY types available on the port. The actual PHY types available are read by the device driver using Set PHY config command. When auto-negotiation is enabled, the XL710 negotiates and selects one of the PHY types enabled. When auto-negotiation is disabled, this field should enable only a single value and then the XL710 is forced to operate in that selected mode.
Link Speed	4	Link speed	BLOCKED. Cannot be modified in MFP mode.	Set link speed on port, only one operational speed is set by the device driver. Bit 4.2 = 1 Gb/s. Bit 4.3 = 10 Gb/s. Bit 4.4 = 40 Gb/s. Other bits - Reserved, Must be zero. This parameter is used by the device driver to set the operational link speed of the port. The XL710 might have the ability to support multiple link speeds on the same port that can be found using the Get PHY ability command (see Section 3.2.4.1.4). One of the link speeds can be enabled due to the result of auto-negotiation. This command can be used by the device driver to manually set the link speed when auto-negotiation is disabled.
Pause ability	5.0:5.1	Pause ability	BLOCKED. Cannot be modified in MFP mode.	5.0 set to 1b to enable IEEE 802.3x Tx link pause ability, or set to 0b to disable pause ability. 5.1 set to 1b to enable IEEE 802.3x Rx link pause ability, or set to 0b to disable 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 XL710's pause ability can be read by using the Get link status or Get PHY abilities 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 in order for the setting to take effect on the link.
Low power mode	5.2	Low power mode	PARTIAL 1. Baseline value is loaded from the NVM. 2. If enabled by default then each PF might disable this. 3. If disabled than PFs cannot enable it. **Note**: No state retention from event to event. For example, when coming out of D3cold, resets the state back to the NVM default.	Set to 1b to enable PHY in low power mode or set to 0b for normal operation. This field is valid only if the PHY supports low power mode. Certain external PHYs connected to the port might have the ability to support low power mode. Setting the PHY in low power mode affects normal operation. This bit should be set to 0b and set restart auto-negotiation to bring the link to normal operation from low power mode.

Table 3-33. Set PHY config command data structure (Continued)

Name	Bytes.Bits	Value	MFP Mode	Remarks
Enable Link	5.3	Enable link	BLOCKED. Cannot be modified in MFP mode. ***Note:*** If the link is disabled by default at power on it might be enabled by any of the PFs but then none of the PFs are able to disable it again.	Set to 1b to enable the link. Set to 0b to disable the link. Device driver should not force link down when port is being used for manageability or WoL.
Enable AN	5.4	Enable AN	BLOCKED. Cannot be modified in MFP mode.	Set to 1b to enable auto-negotiation on the link.
Enable Atomic Link Update	5.5	Enable atomic link update	Flexible. This field on its own is not reflected to hardware and is therefore flexible.	When this field is set to 1b, firmware automatically executes the Setup link and restart auto-negotiation command following this command. When this field is set to 0b the device driver maintains the responsibility for sending the Setup link and restart auto-negotiation command. When automatic link update is enabled, the device driver should be aware that a link change event may occur following the Set PHY config command.
Reserved	5.6:5.7	Reserved		Reserved, Must be zero.
EEE capability enable	6-7	EEE capability	BLOCKED. EEE advertized capability is loaded from the NVM at POR and cannot be modified by PFs.	Sets EEE capability for each PHY type supported on the port. One bit per PHY type. The XL710 might be capable of supporting multiple PHY types. The following parameter indicates the bit number. Ignores values for unsupported PHY types. Bit 6.1 = EEE is enabled for 100BASE-TX. Bit 6.2 = EEE is enabled for 1000BASE-T. Bit 6.3 = EEE is enabled for 10GBASE-T. Bit 6.4 = EEE is enabled for 1000BASE-KX. Bit 6.5 = EEE is enabled for 10GBASE-KX4. Bit 6.6 = Reserved. Other bits = Reserved, Must be zero. This command is used by the device driver to enable the EEE capability of various PHY types supported on the port. The EEE capability of the XL710 can be read by the Get EEE capability command (see Section 3.2.3.1.1). The device driver might set EEE capability for a subset of PHY types supported by the XL710.

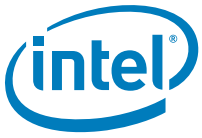


Table 3-33. Set PHY config command data structure (Continued)

Name	Bytes.Bits	Value	MFP Mode	Remarks
EEER	8-11	EEER value	PARTIAL. 1. EEE register settings (TW_SYSTEM, TX_LPI_EN) have a default value loaded from the NVM. 2. TW_SYSTEM: take MIN value of all values assigned by PFs. **Note:** Firmware must insure that the value is not reduced below the MIN value required by the PHY. 3. LPI Disable. **Note:** If EEE is supported and TX_LPI_EN is enabled by default then PFs should be able to disable the XL710 from entering LPI state through an AQ command. Firmware allows even a single PF to change the setting to disable. 4. LPI Enable **Note:** If TX_LPI_EN is disabled (loaded from the NVM) then PFs cannot change this and it is understood that the system does not work with EEE/LPI.	Value to program the EEER register.
Low power Control	12	D3cold LPAN	PARTIAL 1. Baseline value is loaded from NVM. 2. If enabled by default than each PF may disable this. 3. If disabled than PFs cannot enable it. **Note:** No state retention from event to event. For example, when coming out of D3cold, resets the state back to the NVM default.	D3cold LPAN. Bit 12.0 = Set to 0b to disable D3cold low power auto-negotiation. Bit 12.0 = Set to 1b to enable D3cold low power auto-negotiation. Other bits = Reserved, Must be zero.
Reserved	13-15	Reserved		Must be set to 0x0, value is ignored.

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

Table 3-34. Set PHY config command response (opcode: 0x0601)

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	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 (such as MFP mode).
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.

Table 3-34. Set PHY config command response (opcode: 0x0601)

Name	Bytes.Bits	Value	Remarks
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Param0	16-19	Reserved	Zeroed by firmware, value is ignored.
Param1	20-23	Reserved	Must be 0x0, value is ignored.
Data Address high	24-27	Reserved	Value 0x0.
Data Address low	28-31	Reserved	

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 command.

3.2.4.1.2 Set MAC config

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 command parameters data structure is placed in the command descriptor.

Table 3-35. Set MAC config command (opcode: 0x0603)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	Command opcode.
Datalen	4-5	0x0	Must be 0x0b, 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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Set MAC Config	16-31	See Table 3-36	16-byte data structure that holds the Set MAC Config command parameters is listed in Table 3-36.

Table 3-36 lists the data structure of the Set MAC config command parameters such as max frame size, etc.

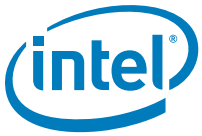


Table 3-36. Set MAC Config command data structure

Name	Bytes.Bits	Value	MFP Mode	Remarks
Max Frame Size	0-1	Max Frame Size	PARTIAL 1. Base line value is loaded from NVM. 2. Always take MAX {value set by all PFs}. **Note:** Firmware should track the baseline value loaded from the NVM and the values assigned by PFs. There is no need to remember which PF set the MIN value. For example, the PF reset does not affect this value.	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.
Reserved	2.0-2.01	Reserved		Must be 0x0.
CRC Enable	2.2	CRC Enable	BLOCKED. Cannot be modified in MFP mode.	Bit 0 = Set to 1b to enable the MAC to append the CRC on transmit. Set to 0b if software appends the CRC. This parameter is used by the device driver to enable the MAC to append the CRC on the link. This is the default configuration. This bit is set to 0b if software needs to disable the MAC to append CRC.
Pacing Config	2.3-2.6	Pacing Config	BLOCKED. Cannot be modified in MFP mode.	Bit 3:0 - This is 4 bit field that allows configuring PACE parameter in the MAC to slow down the effective data rate as defined in Table 3-20.
Reserved	2.7	Reserved		Must be 0b.
Transmit Timer Priority	3	Transmit Timer Priority	N/A	Since the XL710 supports up to eight priorities, this field selects which timer value to set using the value configured in Transmit Timer Value described later in this section. Bit<n> for Priority<n> for n=0 thru 7.
Transmit Timer Value	4-5	Transmit Timer Value	See Max Frame Size previously described.	This is the timer value included in XOFF frames for priority Transmit Timer Priority selected previously. Priority 0 is used for legacy flow control.
FC Refresh Threshold	6-7	FC Refresh Threshold	PARTIAL 1. Base line value is loaded from NVM. 2. Always take MIN {value set by all PFs}. **Note:** Firmware should track the baseline value loaded from NVM and the values assigned by PFs. There is no need to remember which PF set the MIN value. For example, the PF reset does not affect this value.	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.
Reserved	8-15	Reserved		Must be 0x0.

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

Table 3-37. Set MAC Config command response (opcode: 0x0603)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	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 (such as MFP mode).
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Reserved	16-31	Reserved	Value 0x0.

3.2.4.1.3 Setup link and restart auto-negotiation

This command is used by the device driver to setup the link and execute previously sent Set PHY config commands as well as restart the auto-negotiation over the link. This command needs to be executed for any change in link parameters, such as set link speed, etc., to take effect.

This command might have different behaviors in MFP modes:

This is a Direct command.

Table 3-38. Restart AN command (opcode: 0x0605)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	Command opcode.
Datalen	4-5	0x0	Must be 0x0, value is ignored.
Return value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Command Flags	16	Command	Bit 16.1: — Set to 1 to restart the link. Bit 16.2[1]: — Set to 1 to enable link. — Set to 0 to disable link. Other bits = Reserved, must be zero. This command maybe executed automatically by firmware, following a Set PHY config command. In such a case these bits are assigned by firmware as follows: Bit 16.1 is set to 1. Bit 16.2 is copied from the Set PHY config <Enable Link> field.
Reserved	17-31	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.

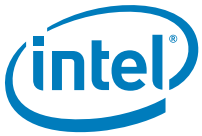


Table 3-39. Restart auto-negotiation command response (opcode: 0x0605)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Reserved	16-31	Reserved	Value 0x0.

3.2.4.1.4 Get PHY abilities

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

This is a Direct command.

Table 3-40. Get PHY abilities command (opcode: 0x0600)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Param0	16-19		First command parameter. 16.0 Report Qualified Modules. List of qualified modules will be part of the response only when this bit is set to 1. 16.1 Report Active/Init. 0b = Report the ACTIVE values. For example, the values assigned to each parameter following the last Set PHY config command. 1b = Report the initial values of the different fields. For example, the values loaded after the last reset event. All other bits are reserved.
Param1	20-23		Second command parameter.
Data Address High	24-27	Buff Addr	High bits of buffer address.
Data Address low	28-31	Buff Addr	Low bits of buffer address.

Table 3-41 lists the get PHY abilities response structure returned by firmware to the Get PHY abilities command.

Direct response, opcode and get PHY abilities response data structure are placed in the descriptor.

Table 3-41. Get PHY abilities command response (opcode: 0x0600)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Param0	16-19		Reserved. Must be set to 0x0.
Param1	20-23		Reserved. Must be set to 0x0.
Data Address High	24-27	Buff Addr	Buffer Address. Buffer content is listed in Table 3-42.
Data Address low	28-31	Buff Addr	

Table 3-42 lists the data structure of the Get PHY abilities command response. The command response returns various PHY parameters such as PHY type, PHY ID, speed ability, pause ability, etc. The following table lists the format of the buffer content for the Get PHY abilities reply.

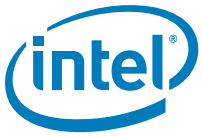


Table 3-42. Get PHY abilities command response data structure

Name	Bytes.Bits	Value	Remarks
PHY Type	0-3	PHY Type	PHY type supported on the port. One bit per PHY type. The XL710 might be capable of supporting multiple PHY types. The following parameter indicates the bit number. 00 = SGMII 01 = 1000BASE-KX. 02 = 10GBASE-KX4. 03 = 10GBASE-KR. 04 = 40GBASE-KR4. 05 = XAUI. 07 = SFI. 08 = XLAUI. 09 = XLPPI. 10 = 40GBASE-CR4 (used with 40 Gb/s QSFP+ direct attach copper). 11 = 10GBASE-CR1 (used with 4x10 Gb/s QSFP+ direct attach copper). 17 = 100BASE-TX. 18 = 1000BASE-T. 19 = 10GBASE-T. 20 = 10GBASE-SR (10 Gb/s SFP+ SR optical module). 21 = 10GBASE-LR (10 Gb/s SFP+ LR optical module). 22 = 10GBASE-SFP+Cu (direct attach copper). 23 = 10GBASE-CR1 (4x10 Gb/s QSFP+ CR over direct attach copper). 24 = 40GBASE-CR4 (40 Gb/s QSFP+ CR over direct attach copper). 25 = 40GBASE-SR4 (40 Gb/s QSFP+ SR optical module). 26 = 40GBASE-LR4 (40 Gb/s QSFP+ LR optical module). 30Other bits - Reserved, must be zero. This parameter is used by the device driver to find out the various PHY types supported on the port. The XL710 might support multiple PHY types on the same port. One of the PHY types might be enabled due to the result of auto-negotiation or manually set by the firmware when auto-negotiation is disabled.
Link Speed Ability	4	Link Speed	Link rate supported on the port. Bit 4.2 = 1 Gb/s. Bit 4.3 = 10 Gb/s. Bit 4.4 = 40 Gb/s. Bit 4.5 = Reserved. Other bits - Reserved, must be zero. This parameter is used by the device driver to find out the various link speeds supported on the port. The XL710 might have the ability to support multiple link speeds on the same port. One of the link speeds might be enabled due to the result of auto-negotiation or manually set by the device driver when auto-negotiation is disabled.
Pause Ability	5.0:5.1	Pause Ability	5.0 returns 1b if the port supports IEEE 802.3x Tx link pause or returns 0b otherwise. 5.1 returns 1b if the port supports IEEE 802.3x Rx link pause or returns 0b otherwise. This parameter is used by the device driver to find out the IEEE 802.3x pause ability of the port.
Low Power Ability	5.2	Low Power Ability	Returns 1b if the PHY has low power ability or returns 0b otherwise. Certain external PHYs connected to the port might have the ability to support low power mode. Setting the PHY to low power mode affects normal operation.
Link Mode	5.3	Link Mode	0b = Link is disabled. 1b = Link is enabled.
AN Mode	5.4	AN Mode	1b = AN is enabled. 0b = AN is disabled.

Table 3-42. Get PHY abilities command response data structure

Name	Bytes.Bits	Value	Remarks
Enable Module Qualification	5.5	Enable Module Qualification	Returns 1b if a external module or PHY qualification check is enabled.
Reserved	5.6:5.7	Reserved	Reserved, must be 0b.
EEE Capability	6-7	EEE Capability	EEE capability for each PHY type supported on the port. One bit per PHY type. The XL710 might be capable of supporting multiple PHY types. The following parameter indicates the bit number. Bit 6.2 = EEE is supported for 1000BASE-T. Bit 6.3 = EEE is supported for 10GBASE-T. Bit 6.4 = EEE is supported for 1000BASE-KX. Bit 6.5 = EEE is supported for 10GBASE-KX4. Other bits = Reserved, must be zero. This parameter is used by the device driver to find out the EEE capability of various PHY types supported on the port. The XL710 might support multiple PHY types on the same port and EEE capability is indicated for each PHY type.
EEER	8-11	EEER Value	Content of EEER register.
Low Power Control	12	D3ColdLPAN	D3cold LPAN. Bit 12.0 0b = D3cold low power auto-negotiation disabled. 1b = D3cold low power auto-negotiation enabled. Other bits = Reserved, must be zero.
Reserved	13-15	Reserved	Reserved, must be zero.
Current PHY ID/ Vendor OUI	16-19	PHY ID/OUI	This parameter is used by the device driver to find out the PHY/module ID connected on the port. If the XL710 is connected to an external BASE-T PHY: This four-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 17:16 = Register3. Bytes 19:18 = Register2. If the XL710 is connected to an external module: This field returns the three-byte vendor OUI of the module (MSB is padded with zeros).
Current Module Type	20-22	Module Type	Returns the three-byte module ID. First byte: Module identifier. Defined by SFP+ (Addr 0xA0, Byte 0) or QSFP+ (Addr 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+ (Addr 0xA0, Byte 6) or QSFP+ (Addr 134, 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 XL710 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 XL710 uses the unused bits in second byte to report SFP+ direct attach cables.

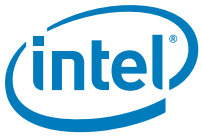


Table 3-42. Get PHY abilities command response data structure

Name	Bytes.Bits	Value	Remarks
Qualified Module Count	23		Number of qualified modules to be listed in the following bytes.
Qualified Module ID-n	24+n*24 - 47+n*24		This is a list of qualified modules that are supported by the XL710 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).

3.2.4.1.5 Get link status

This command is used by the device river 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, etc.

This is a Direct command.

Table 3-43. Get link status command (opcode: 0x0607)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	Command opcode.
Datalen	4-5	0	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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Command Flags	16-17	Reserved	16.1:0. 0 = NOP: LSE notification value is not modified and Get link status response returns the most updated value of enable/disable. 1 = Reserved. 2 = Disable link status event notification to software. 3 = 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 are reserved. Must be 0x0, value is ignored.
Reserved	18-31	Reserved	Must be 0x0, value is ignored.

The following structure describes the response by firmware to the Get link status command.

Table 3-44. Get link status response (opcode: 0x0607)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	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.

Table 3-44. Get link status response (opcode: 0x0607)

Name	Bytes.Bits	Value	Remarks
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Command Flags	16-17	Command Flags	Bit 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 are Reserved. Must be 0, value is ignored
Get Link Status	18-31	See Table 3-45	14-byte data structure that holds the Get link status command response parameters listed in Table 3-45.

Table 3-45 lists the data structure of the Get link status command parameters such as link up/down, negotiated/operating speed, fault conditions, etc.

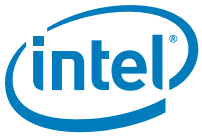


Table 3-45. Get link status response data structure

Name	Bytes.Bits	Value	Remarks
PHY Type	0	PHY Type	Returns operating PHY type or PHY type negotiated if auto-negotiation is enabled. The XL710 might be capable of many different PHY types but only one PHY type is enabled as result of configuration or auto-negotiation. This parameter is an 8-bit integer, each value corresponds to a PHY type as follows. 0x0 = SGMII. 0x1 = 1000BASE-KX. 0x2 = 10GBASE-KX4. 0x3 = 10GBASE-KR. 0x4 = 40GBASE-KR4. 0x5 = XAUI. 0x7 = SFI. 0x8 = XLAUI. 0x9 = XLPPI. 0xA = 40GBASE-CR4 (used with 40Gb/s QSFP+ direct attach copper). 0xB = 10GBASE-CR1 (used with 4x10Gb/s QSFP+ direct attach copper). 0xC = SFP+ Active Direct Attach 0xD = QSFP+ Active Direct Attach 0x11 = 100BASE-TX. 0x12 = 1000BASE-T. 0x13 = 10GBASE-T. 0x14 = 10GBASE-SR (10G SFP+ SR optical module). 0x15 = 10GBASE-LR (10G SFP+ LR optical module). 0x16 = 10GBASE-SFP+Cu (Direct attach copper). 0x17 = 10GBASE-CR1 (4x10G QSFP+ CR over direct attach copper). 0x18 = 40GBASE-CR4 (40G QSFP+ CR over direct attach copper). 0x19 = 40GBASE-SR4 (40G QSFP+ SR optical module). 0x1A = 40GBASE-LR4 (40G QSFP+ LR optical module). 1EOther values are not used. This parameter is used by the device driver to find out the operating PHY type on the port. One of the PHY types might be enabled due to the result of auto-negotiation/parallel detection or manually configured by firmware or software when auto-negotiation is disabled.
Link Speed	1	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 8-bit field, each bit corresponds to a link speed as follows. Only one bit is set at any given time. 1.2 = 1000 Mb/s. 1.3 = 10 Gb/s. 1.4 = 40 Gb/s. 1.4 = Reserved. Other = 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.
Link Status	2.0	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 3-45. Get link status response data structure

Name	Bytes.Bits	Value	Remarks
Link Fault	2.1:2.5	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. Bit 2.5 = Reserved.
Media Available	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 10GBASE-T PHY.
Signal Detect	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	Returns 1b if auto-negotiation completed successfully or returns 0b otherwise. This bit is valid only if the PHY supports auto-negotiation and auto-negotiation is enabled. Typically auto-negotiation is enabled for backplane, BASE-T, and 40 Gb/s CR4 PHYs.
LP AN Ability	3.1	LP AN Ability	Returns 1b if the link partner is able to perform auto-negotiation or returns 0b otherwise. This bit is valid only if the PHY supports auto-negotiation and auto-negotiation is enabled.
Parallel detection Fault	3.2	Parallel Detection fault	Returns 1b if the PHY detects parallel detection fault or returns 0b otherwise. This bit is valid only if the PHY supports auto-negotiation with parallel detection enabled.
FEC Enabled	3.3	FEC Enabled	Returns 1b if FEC is enabled on the link or returns 0b otherwise. This bit is valid only for backplane KR, KR4 and copper CR4 PHYs that support FEC. FEC might be enabled on the link during auto-negotiation.
Low Power state	3.4	Low Power State	Returns 1b if the PHY is in a low power state or returns 0b otherwise.
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 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	When the XL710 is connected to an external SFP+/QSFP+ module, 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	Returns 1b if a temperature alarm condition is reported by the PHY or returns 0b otherwise. Typically an external PHY generates a temperature alarm condition by signaling a PHY interrupt to firmware. The temperature alarm feature should be enabled in the PHY to generate this condition.
Excessive Link Errors	4.1	Excessive Link Errors	Returns 1b if an excessive errors over the link condition is reported by the PHY or returns 0b otherwise

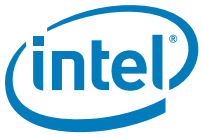


Table 3-45. Get link status response data structure

Name	Bytes.Bits	Value	Remarks
Port TX Suspended	4.2-4.3	Port TX Suspended	0 = Port's Tx active. 1 = Port's Tx suspended and drained. 2 = Reserved. 3 = 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. Other bits are reserved.
Max Frame Size	6-7	Max Frame Size	Maximum frame size set on this port.
Reserved	8.0-8.1		Must be 0b.
CRC Enable	8.2	CRC Enable	1b = CRC append is enabled on this port. 0b = CRC append is disabled on this port.
Pacing Config	8.3-8.6	Pacing Config	This is 4-bit field that enables configuring a pace parameter in the MAC to slow down the effective data rate as listed in Table 3-20.
Reserved	8.7		Must be 0b.
Reserved	9-13	Reserved	Reserved.

3.2.4.1.6 Link status event (opcode: 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 3-46 can be individually masked from generating LSE by using the Set event mask Command (See Section 3-47). The LSE uses the same link status response data structure listed in Table 3-44 and Table 3-45. Firmware posts this data structure to the admin receive queue with the Flags.CMP bit cleared indicating that this is an asynchronous event generated by firmware (such as the message is not in response to an AQ command from software).

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 command (See Table 3-43). 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 3-44).

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 3-46. 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	Signal detect indication has been asserted or de-asserted.
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.

3.2.4.1.7 Set event mask

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.

In MFP mode, multiple PFs can independently set the Event Mask command to mask the event causing sources to that PF. The Set Event Mask command from one of the multiple functions might not affect event masks for other PFs.

This is a Direct command.

Table 3-47. Set event mask command (opcode: 0x0613)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	Command opcode.
Datalen	4-5	0x0	Must be 0,x0 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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.

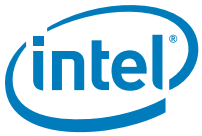


Table 3-47. Set event mask command (opcode: 0x0613)

Name	Bytes.Bits	Value	Remarks
Data Address High	16-19	Reserved	Value 0x0.
Data Address Low	20-23	Reserved	
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. 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 3-48. Set event mask command response (opcode: 0x0613)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Reserved	16-31	Reserved	Must be 0x0, value is ignored.

3.2.4.1.8 Set loopback modes

This command is used by the device driver to set the different loopback modes on the port. This command is used for diagnostic or monitoring purposes only. The command should not be executed during normal operation as this may disrupt link operation.

This is a Direct command.

Table 3-49. Set Loopback modes command (opcode: 0x0618)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	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 is copied by firmware into the completion of this command.

Table 3-49. Set Loopback modes command (opcode: 0x0618)

Name	Bytes.Bits	Value	Remarks
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Loopback Mode	16-17	Loopback mode	Set loopback modes. Bit 16.0 = PHY local loopback. Bit 16.1 = PHY remote loopback. Bit 16.2 = MAC local loopback. Other bits = Reserved, must be zero. ***Note:*** This is a diagnostic feature so all PFs can use it even in MFP mode.
Reserved	18-31	Reserved	Must be 0x0, value is ignored.

The following structure describes the response by firmware to the Set Loopback mode command.

Table 3-50. Set loopback mode command response (opcode: 0x0618)

Name	Bytes.Bits	Value	Remarks
Flags	1-0	0x0	See Section 7.10.5.2.1 for details.
Opcode	2-3	Opcode	Command opcode.
Datalen	4-5	0	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 (such as MFP mode).
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command
Reserved	16-31	Reserved	Must be 0x0, value is ignored.

3.2.5 Possible port to physical lane configurations for various PHY interfaces

Table 3-51 lists possible port-to-physical lane configurations allowed by the XL710 for various PHY interfaces for single, dual and quad port configurations.

Table 3-51. Possible port to physical lane configurations for various PHY interfaces

Cfg ID	Application	Interface Protocol	Physical lane: Group A				Physical lane: Group B				PF Assignment to Port[1]
			L0	L1	L2	L3	L0	L1	L2	L3	
0.3	Dual Port Backplane 1/10G Serial	KR	P0.L0		P1.L0						PF1 to P3
		KX	P0.L0		P1.L0						
		AN	P0.L0		P1.L0						
0.4	Dual Port Backplane 1/10G Serial	KR					P0.L0		P1.L0		PF0 to P1 PF1 to P3
		KX					P0.L0		P1.L0		
		AN					P0.L0		P1.L0		

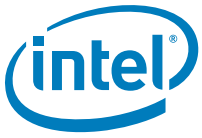


Table 3-51. Possible port to physical lane configurations for various PHY interfaces

Cfg ID	Application	Interface Protocol	Physical lane: Group A				Physical lane: Group B				PF Assignment to Port[1]
			L0	L1	L2	L3	L0	L1	L2	L3	
0.5	Dual Port Backplane 1/10G Serial	KR	P1.L0				P0.L0				PF0 to P1 PF1 to P0
		KX	P1.L0				P0.L0				
		AN	P1.L0				P0.L0				
1.0	Dual Port Backplane 1/10G 4-lanes	KX4	P0.L0	P0.L1	P0.L2	P0.L3	P1.L0	P1.L1	P1.L2	P1.L3	
		KX	P0.L0				P1.L0				
		AN	P0.L0				P1.L0				
1.1	Dual Port Backplane 1/10G 4-lanes	KX4	P0.L3	P0.L2	P0.L1	P0.L0	P1.L3	P1.L2	P1.L1	P1.L0	
		KX				P0.L0				P1.L0	
		AN				P0.L0				P1.L0	
1.2	Dual Port Backplane 1/10G 4-lanes	KX4	P0.L3	P0.L2	P0.L1	P0.L0	P1.L0	P1.L1	P1.L2	P1.L3	
		KX				P0.L0	P1.L0				
		AN				P0.L0	P1.L0				
2.0[2]	Single Port Backplane 40G/10G/1G 4-lanes	KR4	P0.L0	P0.L1	P0.L2	P0.L3					
		KR	P0.L0								
		KX4	P0.L0	P0.L1	P0.L2	P0.L3					
		KX	P0.L0								
		AN	P0.L0								
2.1[2]	Single Port Backplane 40G/10G/1G 4-lanes	KR4	P0.L3	P0.L2	P0.L1	P0.L0					
		KR				P0.L0					
		KX4	P0.L3	P0.L2	P0.L1	P0.L0					
		KX				P0.L0					
		AN				P0.L0					
2.2[2]	Single Port Backplane 40G/10G/1G 4-lanes	KR4					P0.L0	P0.L1	P0.L2	P0.L3	PF0 to P1
		KR					P0.L0				
		KX4					P0.L0	P0.L1	P0.L2	P0.L3	
		KX					P0.L0				
		AN					P0.L0				
2.3[2]	Single Port Backplane 40G/10G/1G 4-lanes	KR4					P0.L3	P0.L2	P0.L1	P0.L0	PF0 to P1
		KR								P0.L0	
		KX4					P0.L3	P0.L2	P0.L1	P0.L0	
		KX								P0.L0	
		AN								P0.L0	
2.4	Dual Port Backplane 40G/10G/1G 4-lanes	KR4	P0.L0	P0.L1	P0.L2	P0.L3	P1.L0	P1.L1	P1.L2	P1.L3	
		KR	P0.L0				P1.L0				
		KX4	P0.L0	P0.L1	P0.L2	P0.L3	P1.L0	P1.L1	P1.L2	P1.L3	
		KX	P0.L0				P1.L0				
		AN	P0.L0				P1.L0				

Table 3-51. Possible port to physical lane configurations for various PHY interfaces

Cfg ID	Application	Interface Protocol	Physical lane: Group A				Physical lane: Group B				PF Assignment to Port[1]
			L0	L1	L2	L3	L0	L1	L2	L3	
2.5	Dual Port Backplane 40G/10G/1G 4-lanes	KR4	P0.L3	P0.L2	P0.L1	P0.L0	P1.L3	P1.L2	P1.L1	P1.L0	
		KR				P0.L0				P1.L0	
		KX4	P0.L3	P0.L2	P0.L1	P0.L0	P1.L3	P1.L2	P1.L1	P1.L0	
		KX				P0.L0				P1.L0	
		AN				P0.L0				P1.L0	
2.8	Single Port Backplane 40G/10G/1G 4-lanes	KR4	P0.L1		P0.L2	P0.L3	P0.L0				
		KR					P0.L0				
		KX4	P0.L1		P0.L2	P0.L3	P0.L0				
		KX					P0.L0				
		AN					P0.L0				
3.0	Quad Port Backplane 1/10G Serial	KR	P0.L0	P1.L0	P2.L0	P3.L0					
		KX	P0.L0	P1.L0	P2.L0	P3.L0					
		AN	P0.L0	P1.L0	P2.L0	P3.L0					
3.1	Quad Port Backplane 1/10G Serial	KR					P0.L0	P1.L0	P2.L0	P3.L0	PF0 to P1 PF1 to P0
		KX					P0.L0	P1.L0	P2.L0	P3.L0	
		AN					P0.L0	P1.L0	P2.L0	P3.L0	
3.2	Quad Port Backplane 1/10G Serial	KR	P0.L0	P1.L0			P2.L0	P3.L0			PF1 to P2 PF2 to P1
		KX	P0.L0	P1.L0			P2.L0	P3.L0			
		AN	P0.L0	P1.L0			P2.L0	P3.L0			
3.5	Quad Port Backplane 1/10G Serial	KR	P2.L0		P3.L0		P0.L0		P1.L0		PF0 to P1 PF1 to P0
		KX	P2.L0		P3.L0		P0.L0		P1.L0		
		AN	P2.L0		P3.L0		P0.L0		P1.L0		
3.8	Quad Port Backplane 1/10G Serial	KR	P1.L0		P2.L0	P3.L0	P0.L0				PF0 to P1 PF1 to P0
		KX	P1.L0		P2.L0	P3.L0	P0.L0				
		AN	P1.L0		P2.L0	P3.L0	P0.L0				
4.0	Single Port QSFP+ 40G 4-lane	CR4	P0.L3	P0.L2	P0.L0	P0.L1					X.01 XLPPI
		XLPPI	P0.L3	P0.L2	P0.L0	P0.L1					
		AN			P0.L0						
4.2	Single Port QSFP+ 40G 4-lane	CR4					P0.L3	P0.L2	P0.L0	P0.L1	PF0 to P1 X.01 XLPPI
		XLPPI					P0.L3	P0.L2	P0.L0	P0.L1	
		AN							P0.L0		
4.4	Single Port QSFP+ 40G 4-lane	CR4	P0.L1		P0.L2	P0.L3	P0.L0				X.01 XLPPI
		XLPPI	P0.L1		P0.L2	P0.L3	P0.L0				
		AN					P0.L0				
4.5	Dual Port QSFP+ 40G 4-lane	CR4	P0.L3	P0.L2	P0.L0	P0.L1	P1.L3	P1.L2	P1.L0	P1.L1	X.01 XLPPI
		XLPPI	P0.L3	P0.L2	P0.L0	P0.L1	P1.L3	P1.L2	P1.L0	P1.L1	
		AN			P0.L0				P1.L0		

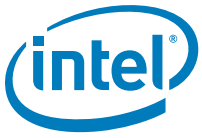


Table 3-51. Possible port to physical lane configurations for various PHY interfaces

Cfg ID	Application	Interface Protocol	Physical lane: Group A				Physical lane: Group B				PF Assignment to Port[1]
			L0	L1	L2	L3	L0	L1	L2	L3	
5.0	Single Port External PHY 40G 4-lane	XLAUI	P0.L3	P0.L2	P0.L0	P0.L1					
5.1	Single Port External PHY 40G 4-lane	XLAUI	P0.L1	P0.L0	P0.L2	P0.L3					
5.2	Single Port External PHY 40G 4-lane	XLAUI					P0.L3	P0.L2	P0.L0	P0.L1	PF0 to P1
5.3	Single Port External PHY 40G 4-lane	XLAUI					P0.L1	P0.L0	P0.L2	P0.L3	PF0 to P1
5.4	Single Port External PHY 40G 4-lane	XLAUI	P0.L1		P0.L2	P0.L3	P0.L0				X.01 XLPPI
6.0	Quad Port QSFP+ 4x10G Serial	SFI	P3.L0	P2.L0	P0.L0	P1.L0					
6.1	Quad Port QSFP+ 4x10G Serial	SFI					P3.L0	P2.L0	P0.L0	P1.L0	PF0 to P1 PF1 to P0
6.2	Quad Port QSFP+ 4x10G Serial	SFI	P1.L0		P2.L0	P3.L0	P0.L0				
7.0	Quad Port SFP+ 10G Serial	SFI	P0.L0	P1.L0	P2.L0	P3.L0					
7.1	Quad Port SFP+ 10G Serial	SFI					P0.L0	P1.L0	P2.L0	P3.L0	
7.2	Quad Port SFP+ 10G Serial	SFI	P1.L0		P2.L0	P3.L0	P0.L0				PF0 to P1 PF1 to P0
8.0, 8.01	Dual Port External PHY 2x10G XAUI & 2x 1G/100M SGMII	XAUI	P0.L0	P0.L1	P0.L2	P0.L3	P1.L0	P1.L1	P1.L2	P1.L3	X.01 SGMII
		SGMII	P0.L0				P1.L0				
8.1, 8.11	Dual Port External PHY 2x10G XAUI & 2x 1G/100M SGMII	XAUI	P0.L3	P0.L2	P0.L1	P0.L0	P1.L3	P1.L2	P1.L1	P1.L0	X.01 SGMII
		SGMII				P0.L0				P1.L0	
8.2, 8.21	Dual Port External PHY 2x10G XAUI & 2x 1G/100M SGMII	XAUI	P0.L3	P0.L2	P0.L1	P0.L0	P1.L0	P1.L1	P1.L2	P1.L3	X.01 SGMII
		SGMII				P0.L0	P1.L0				
8.3, 8.31	Dual Port External PHY 2x10G XAUI & 2x 1G/100M SGMII	XAUI	P0.L0	P0.L1	P0.L2	P0.L3	P1.L3	P1.L2	P1.L1	P1.L0	X.01 SGMII
		SGMII	P0.L0							P1.L0	

Table 3-51. Possible port to physical lane configurations for various PHY interfaces

Cfg ID	Application	Interface Protocol	Physical lane: Group A				Physical lane: Group B				PF Assignment to Port[1]
			L0	L1	L2	L3	L0	L1	L2	L3	
8.4, 8.41	Dual Port External PHY 2x10G XAUI & 2x 1G/100M SGMII	XAUI	P1.L0	P1.L1	P1.L2	P1.L3	P0.L0	P0.L1	P0.L2	P0.L3	PF0 to P1 PF1 to P0 X.01 SGMII
		SGMII	P1.L0				P0.L0				
8.5, 8.51	Dual Port External PHY 2x10G XAUI & 2x 1G/100M SGMII	XAUI	P1.L3	P1.L2	P1.L1	P1.L0	P0.L3	P0.L2	P0.L1	P0.L0	PF0 to P1 PF1 to P0 X.01 SGMII
		SGMII				P1.L0				P0.L0	
8.6, 8.61	Dual Port External PHY 2x10G XAUI & 2x 1G/100M SGMII	XAUI	P1.L3	P1.L2	P1.L1	P1.L0	P0.L0	P0.L1	P0.L2	P0.L3	PF0 to P1 PF1 to P0 X.01 SGMII
		SGMII				P1.L0	P0.L0				
8.7, 8.71	Dual Port External PHY 2x10G XAUI & 2x 1G/100M SGMII	XAUI	P1.L0	P1.L1	P1.L2	P1.L3	P0.L3	P0.L2	P0.L1	P0.L0	PF0 to P1 PF1 to P0 X.01 SGMII
		SGMII	P1.L0							P0.L0	
10.0[3], 10.01	Quad Port External PHY 4x10G KR 4x 100M/1G SGMII	KR	P0.L0	P1.L0	P2.L0	P3.L0					X.01 SGMII
		SGMII	P0.L0	P1.L0	P2.L0	P3.L0					
		AN (KR Only)	P0.L0	P1.L0	P2.L0	P3.L0					
10.13 10.11	Quad Port External PHY 4x10G KR 4x 100M/1G SGMII	KR					P0.L0	P1.L0	P2.L0	P3.L0	X.01 SGMII
		SGMII					P0.L0	P1.L0	P2.L0	P3.L0	
		AN (KR Only)					P0.L0	P1.L0	P2.L0	P3.L0	
10.23 10.21	Quad Port External PHY 4x10G KR 4x 100M/1G SGMII	KR	P1.L0		P2.L0	P3.L0	P0.L0				PF0 to P1 PF1 to P0 X.01 SGMII
		SGMII	P1.L0		P2.L0	P3.L0	P0.L0				
		AN (KR only)	P1.L0		P2.L0	P3.L0	P0.L0				
10.3 10.31	Quad Port External PHY 4x10G SFI 4x1G SGMII	SFI	P0.L0	P1.L0	P2.L0	P3.L0					X.01 SGMII
		SGMII	P0.L0	P1.L0	P2.L0	P3.L0					
10.4 10.41	Quad Port External PHY 4x10G SFI 4x1G SGMII	SFI					P0.L0	P1.L0	P2.L0	P3.L0	X.01 SGMII
		SGMII					P0.L0	P1.L0	P2.L0	P3.L0	
10.5 10.51	Quad Port External PHY 4x10G SFI 4x1G SGMII	SFI	P1.L0		P2.L0	P3.L0	P0.L0				PF0 to P1 PF1 to P0 X.01 SGMII
		SGMII	P1.L0		P2.L0	P3.L0	P0.L0				
11	Dual Port 2x20G-KR2/ 2x10G-KR/ 2x1G-KX/	KR2	P0.L0	P0.L1			P1.L0	P1.L1			
		KR/KX	P0.L0				P1.L0				
		AN	P0.L0				P1.L0				

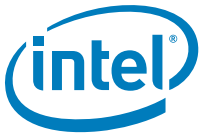


Table 3-51. Possible port to physical lane configurations for various PHY interfaces

Cfg ID	Application	Interface Protocol	Physical lane: Group A				Physical lane: Group B				PF Assignment to Port[1]
			L0	L1	L2	L3	L0	L1	L2	L3	
11.1	Dual Port 2x20G-KR2/ 2x10G-KR/ 2x1G-KX/	KR2	P0.L0	P0.L1	P1.L0	P1.L1					PF1 to P3
		KR/KX	P0.L0		P1.L0						
		AN	P0.L0		P1.L0						

1. See Section 3.2.5.1.
2. Single port backplane configuration: Lane configurations are loaded from the NVM and cannot be changed dynamically, so use single port only for usage models that need fixed lane assignments. Otherwise, for all other usage models, recommend dual 40 Gb/s configuration if either group A and Group B lanes need to be used for high availability.
3. Auto-negotiation should be enabled in KR mode in order to insure validity of electrical training.

3.2.5.1 PF-to-lane mapping

In Table 3-51 there is a “PF Mapping” column that is used to identify cases where on top of the physical arrangement of the device’s SerDes a logical mapping needs to be done in order to map host PF to a physical interface.

The following figure shows one such example.

Note: In SFP mode the default assignment of PF<n> is to Port<n>.

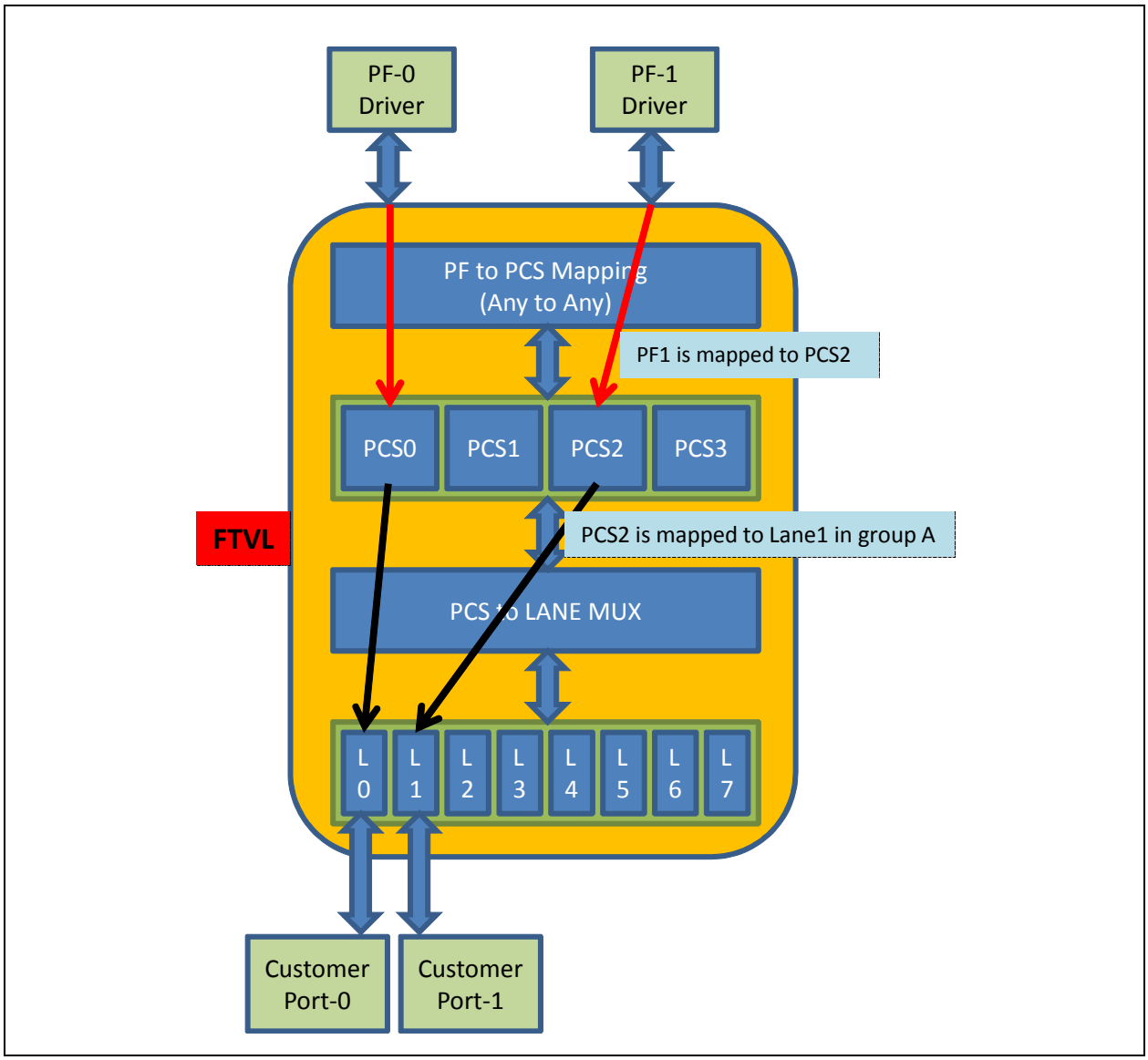
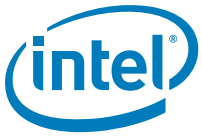


Figure 3-22. PF-to-lane mapping example - Cfg-ID 0.1

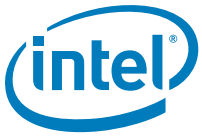


3.2.6 Supported Media Types

The following table describes the supported SFP, SFP+ and QSFP+ setups.

Table 3-52. Supported Media Types

Speed	Media	Description
40G	QSFP+ SR4 / LR4 optical modules	***Note:*** Single and multi-speed • •
	QSFP+ AoC's	• These modules are identified based on the compliance code field. ***Note:*** Some legacy AoC require identification based on additional fields.
10G	SFP+ SR / LR single-speed (10G) and multi-speed (1G / 10G) optical modules	• These modules are identified based on the compliance code field. • •
	SFP+ DA twin-ax cables	• •
	SFP+ AoC's (Active optical Cables)	• These modules are identified based on the compliance code field. • AoC cables are enabled whenever the device is enabled for 10G-SR ***Note:*** Only "Limiting Initialization" cables are supported.
	QSFP+ and breakout DA twin-ax cables or Active Copper cables	• These modules are identified based on the compliance code field. • • ***Note:***



3.3 Non-volatile Memory (NVM)

3.3.1 General Overview

There are new conditions specific to the XL710 that induced a new approach concerning NVM access:

- LAN traffic can be handled only if the EMP code runs
- For flexibility reasons, the main EMP code is retrieved from the NVM and not from ROM.
- Unless authenticated, the EMP firmware code presents a potential security threat for the system:
  - Modification/tampering firmware code is presently undetectable to the rest of the system
  - The XL710's various virtualization capabilities (like SR-IOV with VFs and PFs) can reach into virtually any VM or VMM address space
  - A malicious firmware is capable of sending out the harvested data from the server to unauthorized recipients via DMA reads … again, undetected. The host CPU is not involved in this process.
  - A malicious firmware is capable of injecting a malware into the host, bypassing other platform-level access control.

3.3.1.1 Requirements on NVM access

The basic requirements from NVM access in the XL710 include the following:

1. Guarantee that only Intel provided firmware code (EMP) is run by the XL710.
   a. If the EMP code that presented a security weakness was released to customers and loaded into the XL710, Intel guarantees it cannot be reused once a newer safe EMP code is loaded.
2. Protect NVM update flow from power failure before completion. This implies the image-update procedure of modules uses a double-bank policy.
3. Meet the boot time requirements described in Section 4.2.1.
4. Guarantee that NVM settings that are critical for the XL710's accessibility from the host be provided only by Intel, but can still be replaced in the field with another Intel provided setting if necessary. It relates to the following NVM modules:
   a. PCIe analog
   a. RO PCIR registers auto-load
   b. PHY analog
   c. RO PCIe LCB

Requirements 1. and 4. imply that the contents of several modules be protected via authentication while others be made RO. Refer to Section 3.3.9 for details about NVM authentication.

3.3.1.2 Operational limitations

The NVM protection method selected in the XL710 relies on an authenticate on update concept. It means that the protected modules are not authenticated after initialization, but only before committing to a module update operation. NVM protection is guaranteed by an induction authentication chain, starting from a first secured NVM image and requiring that any step taken later on for modifying the image be secured.

This method induces several limitations and restricted working assumptions:

1. A first good EMP image is loaded into the Flash at the manufacturing site that is assumed to be safe. For example, it can be done via physical means such as nail bed.
   a. It assumes customers (OEM and end-user) know the source of the installed components, the supply chain producing these components is not compromised during manufacturing, and after being deployed the NIC/LOM is physically protected from modification.
   b. The possibility exists that unauthorized firmware be loaded into the XL710 via physical modification post manufacturing, as well as supply chain vulnerabilities. However, firmware updates via programmatic (software) methods are enhanced to require authentication prior to updating NVM settings. Furthermore, host software can independently detect whether the firmware image has an invalid digital signature.
2. In normal operating mode, NVM write accesses are controlled by the XL710 (by EMP) and cannot be performed 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 to a BMC command, which are both handled by the EMP.
3. All supported Flash parts share the same set of opcodes used by the XL710.
4. A blank Flash programming mode is provided (besides the normal programming mode previously mentioned in item 2.) where the Flash can be programmed directly without the EMP being involved via one of the legacy methods:
   a. Memory mapped write via host memory BAR.
   b. Bit banging (GLNVM_FLA register).
5. The blank Flash programming mode is not safe, and must therefore be used only as a last resort to recover from initial mis-configuration. This programming mode is entered either:
   a. When a blank Flash is detected.
   b. When the EMP image loaded (even if authenticated by Intel) does not belong to the XL710.
   c. When the host debug mode is entered via a dedicated JTAG code (0x3E) or via a strapping pins combination, the host can remove address protection of the PF space and write into the register that controls the blank Flash programming mode.

3.3.2 Flash device requirements

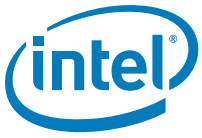
The XL710 merges the 82599 legacy EEPROM and Flash content in a single Flash device. Flash devices require a sector erase instruction if a cell is modified from 0b to 1b. As a result, in order to update a single byte (or block of data) it is required to erase it first. The XL710 supports Flash devices with a sector erase size of 4 KB. Note that many Flash vendors are using the term sector differently. The XL710 Datasheet uses the term Flash sector for a logic section of 4 KB.

The XL710 supports Flash devices that are either write-protected by default after power-up or not. The XL710 is responsible to remove the protection by sending the write-protection removal opcode to the Flash after power up.

The Flash part is clocked by the XL710 at ~25 MHz and no fast read opcode/sequence is used.

The following opcodes are supported by the XL710 as they are common to all supported Flash devices:

1. Write Enable (0x06).
2. Read Status Register (0x05).
3. Write Status Register (0x01). The written data is 0x00 to cancel the Flash default protection.
4. Read Data (0x03). Burst read is supported.
5. Byte/Page Program (0x02). To program 1 to 256 data bytes.
6. 4 KB Sector-Erase (0x20).
7. Chip Erase (0xC7).



8. Read (JEDEC) Identification (0x9F).

3.3.3 Shadow RAM

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

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

   Unlike an EEPROM, Flash devices require rewriting an entire sector even if it comes to updating a single byte. The partial update is first performed against the shadow RAM. Later on, the XL710 commits the entire updated shadow RAM into the Flash.
2. On device-level resets (in contrast to function-level resets), the module (or parts of it) is auto-loaded by the XL710 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 ms) that could have been initiated just before. Flash read accesses are suspended until a Flash sector erase operation completes. NVM auto-load performed further to device-level resets are done from the internal shadow RAM into the registers, without involving Flash read cycles.

The XL710 maintains the first 32 x 4 KB sectors of the Flash 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 entire shadow RAM contents. These banks are referred to as the basic NVM banks. At any time one of these two banks must be valid or else the XL710 is set by hardware default and enters into blank Flash programming mode (refer to Section 3.3.4.2).

Following a Power On Reset (POR), the XL710 copies the valid bank of the Flash device into the internal shadow RAM, 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 shadow RAM content are then copied by the XL710 into the next bank of the NVM, flipping circularly the valid 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 3-23 shows the shadow RAM mapping and interface.

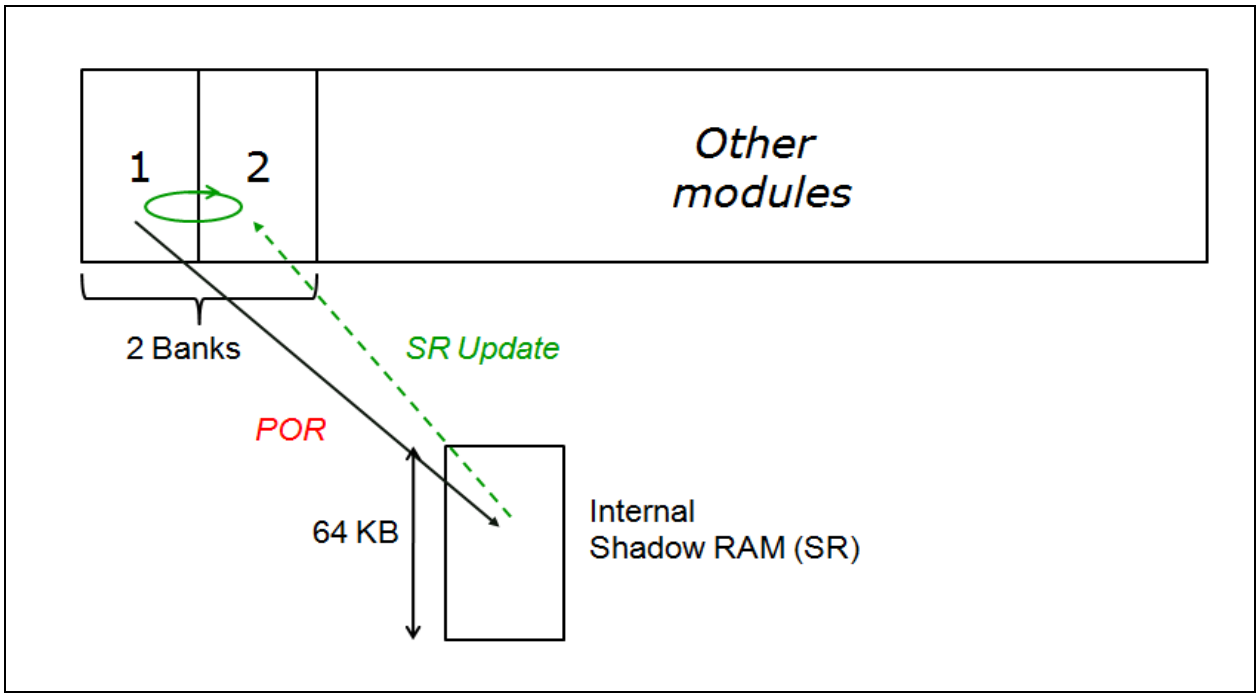


Figure 3-23. NVM shadow RAM

3.3.4 NVM access modes

3.3.4.1 Normal mode

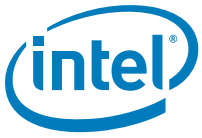
For BIOS read accesses and VPD accesses, any read or write access to the NVM by the host must be preceded by taking ownership of the NVM resource via the Request Resource Ownership admin command. Refer to Section 7.10.12.5. This prevents the following situations:

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

3.3.4.1.1 Normal read access

Read accesses to the NVM do not require the EMP being involved (except if performed via the NVM Read admin command). Available read accesses are as follows:

1. An NVM Read admin command from the PF.
2. VPD register set. The EMP asserts the *Done* bit.
3. Memory mapped read via the memory/expansion 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).
4. GLNVM_SRCTL and GLNVM_SRDATA register set, for reads only and addressed to the shadow RAM contents only.



3.3.4.1.2 Normal write access

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

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

NVM write access attempts performed via the memory/expansion ROM BARs are not performed by the XL710, although PCIe transactions are (apparently) normally completed.

3.3.4.2 Blank Flash programming mode

The XL710 enters blank Flash programming mode based on the following:

a. When a blank Flash is detected. It means that the NVM *Validity* field (NVM Control Word 1) read from the two basic banks is not equal to 01b.
b. When the EMP image read at initialization time does not belong to the XL710.
c. When host debug mode is entered via a dedicated JTAG code (0x3E) or via a strapping pin combination, the host can remove address protection of the PF space and then clear the *GLNVM_FLA.LOCKED* bit. It is recommended that Flash programming platforms at manufacturing sites be provided with the JTAG and/or a strapping option 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 XL710 is in this mode (see steps a. and b. previously described), the EMP code not loaded from the NVM and the EMP remains disabled. The XL710 is not able to exchange any kind of traffic over the lines and no admin command can be posted. The XL710 is in an unknown operational state where only the Flash programming flow is operational.

3.3.4.2.1 Additional NVM accesses

Additional NVM accesses are enabled while in blank Flash programming mode:

1. Direct bit-banging access to the Flash part via GLNVM_FLA register — It can be used for read and write operations of the NVM content or the Flash part Control and Status registers.
2. Memory mapped write to the Flash via memory BAR— When in this mode, the memory BAR access to the Flash is always available since the GLPCI_LBARCTRL.FLASH_EXPOSE CSR bit is set by default.
3. The GLNVM_SRCTL, GLNVM_SRDATA register set for a write into shadow RAM.

The GLNVM_FLA.LOCKED bit — It indicates (when = to 1b) that the GLNVM_FLA register and that the memory-mapped NVM write access is locked to all software (normal programming mode). When = to 0b, software can use the bit-banging and the memory-mapped write accesses to directly access the NVM device. The bit assertion to 1b is controlled by hardware (with no EMP involved) and once asserted it is reset only on the next POR event.

3.3.5 NVM update flows

The flows described in this section only affects normal programming mode. When in the blank Flash programming mode, refer to NVM access procedures described in Section 3.3.8.

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 7-209. Otherwise, the sequence must be fragmented in several shorter sequences, releasing NVM ownership in between.

The software tool is responsible to re-compute and update the software checksum (word 0x3F) each time a change was made to words 0x00-0x3E further to NVM Update commands issued.

3.3.5.1 Flash high level map

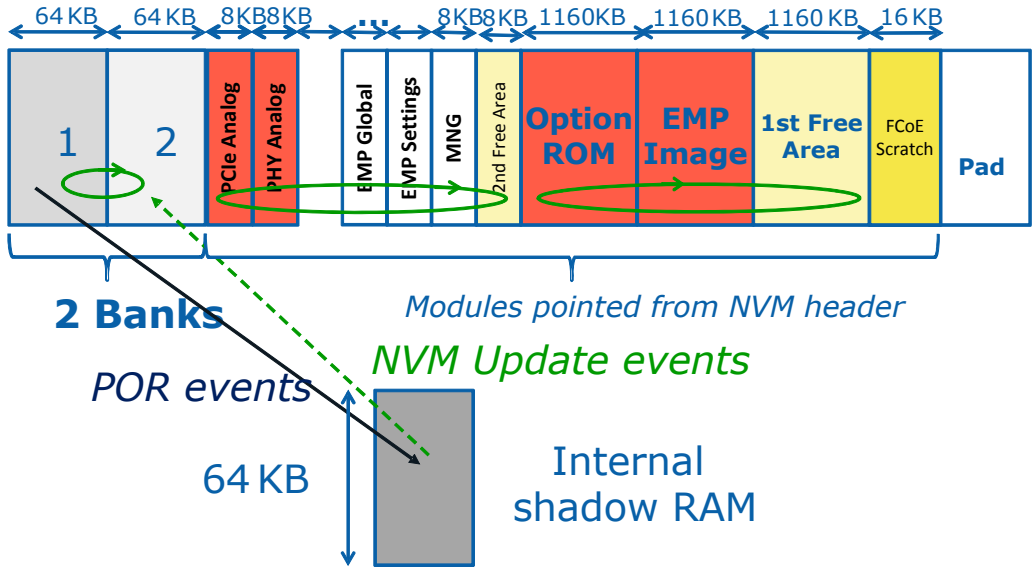
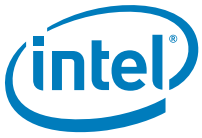


Figure 3-24. Flash high level map

Figure 3-24shows the high level mapping of the NVM in the XL710, while the exact mapping is given in Chapter 6.0. It is made of the following areas:

- Two Basic Banks 1, 2— Among other modules, it contains the VPD area. The current valid bank is mirrored into the internal shadow RAM at POR events. Changes that are made in the shadow RAM are finally dumped into the next bank, which becomes the current valid, and so forth cyclically. Refer to Section 3.3.5.2 and Section 3.3.5.3 for the update flows.
- EMP Image Area— It contains the firmware code to be run by the EMP embedded processor. It requires authentication before an update. Refer to Section 3.3.5.5 for the update flow. The last 4 KB sector of the EMP image is covered by authentication but is not part of the compiled EMP code version. This sector contains commands to EMP for updating NVM RO words and modules (refer to Section 3.3.6). This last EMP image sector is referred as the RO commands section.
- Expansion/Option ROM Area (OROM area)— It contains pre-boot code and settings read by BIOS. Refer to Section 3.3.5.4 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.



- First free area— The free provisioning area used for modifying one of the 1160 KB long areas such as the EMP image area. Once the new module content has been written into the first free area, the old module area is used as the new first free area, and so forth cyclically.
- EMP Global, MNG, and EMP Settings Modules— They contain information to be handled by the EMP. Refer to Section 3.3.5.4 for the update flow.
- PCIe Analog and PHY Analog Modules— These modules are only loaded by the XL710 at power-up events. Refer to Section 3.3.5.5 for the update flow.
- Second Free Area— The free provisioning area used for modifying one of the 8 KB long areas such as EMP settings or MNG areas. Once the new module content has been written into the second free area, the old module area is used as the new second free area, and so forth cyclically.
- FCoE Scratch Pad Area— FCoE drivers intend to use NICs that provide available NVM storage by identifying a minimal area (4 KB per port) to be used for persistent storage of FIP discovery data as well as other potential future debug and/or diagnostic purposes.

3.3.5.2 VPD update

3.3.5.2.1 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 the VPD *Write Enable* bit is set to 1b in NVM Security Control word (offset 0x02), the entire VPD area can be modified later on by a host via the NVM Update AQ command. If VPD tags were modified, it is required to issue a PCIe reset before write accessing the VPD area from PCIe configuration space.

3.3.5.2.2 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 XL710.

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 a 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 VPD-RW area (and not to a VPD RO area).
   a. If not, the EMP clears the VPD *Flag* in the PF configuration space to notify PF VPD software that the transaction completed and then exits the flow.
4. EMP writes the change into shadow RAM — It writes the change into the shadow RAM via the EMPNVM_SRCTL, EMPNVM_SRDATA register set.
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 takes ownership over the NVM resources for a write — The EMP checks that the NVM is not busy and not owned by software. It then marks it internally to be owned by the EMP for a write operation, starting the 3-minute timer. Refer to Table 7-209.
   - If the Flash is busy by a previous sector erase operation or if NVM ownership is held by software, it might that the flow needs to be restarted from step 1. at this stage by successive VPD write accesses initiated by VPD software. That way, successive VPD writes might generate only two next bank erase operations, one for the first and one for the last VPD write in a sequence, thus increasing Flash longevity.

7. The EMP erases the next bank — It erases the contents of the next basic bank sectors, via the procedure described in Section 3.3.8.4.
8. The EMP copies shadow RAM into the next bank — It copies the shadow RAM into the next basic bank sectors with the exception of the *Validity* field, which is left as all ones. It uses the internal EMPNVM_FLCTL, EMPNVM_FLDATA register set.
9. The EMP 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 (such as Flash defect), exit the flow. The EMP releases the NVM ownership (internally). Refer to Section 7.10.12.6.
   b. If the check was successful, the EMP validates the new bank and invalidates the old bank.
   — The *Validity* field of the new bank is set to 01b. The EMP checks that the *Validity* field is read as written in the Flash. If not, then go to the previous sub-step.
   — The EMP toggles the state of the BANK1VAL bit in the GLNVM_GENS register to indicate that the non-valid bank became the valid one and vice versa.
   — The current (old) bank is invalidated by setting its *Validity* field to 00b.
   — The EMP releases the NVM ownership (internally). Refer to Section 7.10.12.6.

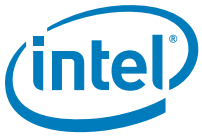
Note: Users must be made aware that dumping the VPD change into the Flash might take a few hundreds of a ms after the VPD transaction completes to software (by clearing the VPD *Flag*). As a result, they must wait few seconds before they can shut down the system.

If the VPD write access is attempted by the host when the XL710 has just started a shadow RAM dump (step 8), then it might be that the write request times out.

3.3.5.3 Updating a RW module or RW words mapped inside shadow RAM

Besides any RW word in the NVM header (refer to the list of RO words in Section 6.1.1), the following NVM modules are affected by this flow:

1. PCIR Registers Auto-load (pointed by NVM word 0x08).
2. PBA Block (pointed by NVM word 0x16).
3. Boot Configuration (pointed by NVM word 0x17).
4. Alternate SAN MAC Addresses (pointed by NVM word 0x27).
5. Permanent SAN MAC Addresses (pointed by NVM word 0x28).
6. VPD Area (pointed by NVM word 0x2F) — if the VPD *Write Enable* bit is set to 1b.
7. PXE Setup Options (pointed by NVM word 0x30).
8. PXE Configuration Customization Options (pointed by NVM word 0x31).
9. Software Alternate MAC Addresses (pointed by NVM word 0x37).
10. POR Registers Auto-load (pointed by NVM word 0x38).
11. GLOBR Registers Auto-load (pointed by NVM word 0x3B).
12. CORER Registers Auto-load (pointed by NVM word 0x3C).
13. Software takes ownership over the NVM resource for a write (see Section 7.10.12.5).
14. Software issues one or several NVM Update commands — It posts the NVM Update admin command to the EMP with the *Authenticated* bit cleared, providing the following parameters (see Section 3.3.10.3):
    - Address in the NVM header of the pointer to the module or 0x0000 for updating a RW word of the NVM header
    - 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 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.

It is assumed that each command in the sequence leaves the shadow RAM with a usable/consistent contents as needed if a device-level reset occurs in the middle of the NVM update sequence.

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

15. The EMP checks that the command is valid and posts a response to software — It performs the following command validity checks and posts a response (ACK/NACK) to software. If one check fails then the EMP flushes the remaining NVM update commands (if any) of the sequence and exits the flow. Otherwise, if no error is encountered, the EMP schedules the NVM command to run in a separate thread, resuming from the next step.
    a. The pointer location is not a RO module or word (refer to RO words in Section 6.1.1). The size of a RO module mapped to the shadow RAM is given at the first word of the module. See Section 6.1.5.1.
    b. The start/end offsets, once applied to the module's location in the basic bank, do not lead to addresses beyond the shadow RAM size.
    c. The start/end offsets, once applied to the module's location in the basic bank, do not overwrite contents of a RO module.
    d. If the pointer location is 0x0000, it means that the NVM update operation might affect fields in the NVM header. In that case, it must be checked that the start/end offsets does not lead to write over RO words.
16. The EMP writes the change into shadow RAM — It writes the change into the shadow RAM via the EMPNVM_SRCTL, EMPNVM_SRDATA register set.
17. If the *Last Command* bit is set in the command — The EMP commits the change into the next basic bank by performing the following steps. Otherwise, it completes the command to software, exits the flow and goes back to step 2. when the next command is posted.
18. The EMP erases next bank — It erases the contents of the next basic bank sectors, via the procedure described in Section 3.3.8.4.
19. The EMP copies shadow RAM into next bank — It copies the shadow RAM into the next basic bank sectors with the exception of the *Validity* field, which is left as all ones. It uses the internal EMPNVM_FLCTL, EMPNVM_FLDATA register set.
20. The EMP 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 a previous step.
    a. If not (such as Flash defect), return an error status — The command completes to software (in the next step) with the Flash defect error bit set.
    b. If the check was successful, the EMP validates the new bank and invalidates the old bank.
    — The *Validity* field of the new bank is set to 01b. The EMP checks that the *Validity* field is read as written in the Flash. If not, then go to the previous sub-step.
    — The EMP toggles the state of the BANK1VAL bit in the GLNVM_GENS register to indicate that the non-valid bank became the valid one and vice versa.
    — The current (old) bank is invalidated by setting its *Validity* field to 00b.
21. The EMP posts an event completion on ARQ to software.
    a. If the NVM ownership timeout for write ends before reaching this step, the EMP flushes the remaining NVM update commands (if any) of the sequence, reporting a timeout error status.
22. Software releases NVM ownership (seeSection 7.10.12.6).
23. Software initiates a system reset (PERST) for loading the modifications into the XL710.

3.3.5.4 Reprogramming a non-authenticated module mapped outside shadow RAM

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

1. EMP Global (pointed by NVM word 0x09)
2. Manageability (pointed by NVM word 0x0E)
3. EMP Settings (pointed by NVM word 0x0F)
4. FCoE Scratch Pad (pointed by NVM word 0x4B)

Modules items 1. to 3. previously listed use the second free area pointer (NVM word 0x46) for their bank swap. This free area is made up of two consecutive 4 KB sectors and it is used for the update of one module at a time.

FCoE scratch pad area uses a single bank (no bank swap) via flat addressing AQ commands (such as a module_pointer field in the command set to 0b).

1. Software takes ownership over the NVM resource for a write— Refer to Section 7.10.12.5.
2. Software issues one or several NVM Erase and/or Update commands — It posts an NVM Erase/Update admin command to the EMP, providing the following parameters (see Section 3.3.10.3):
   - Address in the NVM of the pointer to one of the modules previously listed

     When accessing the FCoE scratch pad area, software must set a null value in the Module_pointer field, which requires the Flash part be addressed as a flat memory from physical address 0x0. Software should do a couple of extra steps to calculate the address of the FCoE scratch pad area:

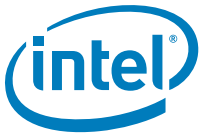
     1. Read data from offset 0x4B, 0x4C in shadow RAM (using GLNVM_SRCTL/SRDATA registers).
     2. Translate the address in 0x4B by removing bit 15 and multiplying by 4 KB.
     3. Use the computed address for a read/update to the Flash content in the scratch area.

     It can also check the size of the scratch area in 0x4C word.
   - Offset inside the module
   - Buffer in host memory
   - Buffer length (maximum supported is 4 KB)

   It is software's responsibility to erase a Flash sector prior to writing it.

   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 Update and/or Erase 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 pointersare updated in 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.
3. The EMP checks that the command is valid and posts a response to software — It performs the following command validity checks and posts a response (ACK/NACK) to software. If one check fails then the EMP flushes the remaining NVM update commands (if any) of the sequence and exits the flow. Otherwise, if no error is encountered, the EMP 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 start/end offsets, once applied to the relevant free area, do not lead to addresses beyond the free area size.



   c. The start/end offsets, once applied to the relevant free area, do not spread over two consecutive 4 KB sectors
4. The EMP erases/writes the free provisioning area — According to the command, the EMP erases the 4 KB sector in the free provisioning area via the procedure previously described in Section 3.3.8.4 or writes the change required by the (elementary) command into it via the EMPNVM_FLCTL, EMPNVM_FLDATA register set.
5. If the update sequence did not end, the EMP posts an event completion on ARQ to software — If the *Last Command* bit is cleared in the (elementary) command, the EMP completes the command and then goes to back to step 3. of this flow for the processing of the next (elementary) command of the sequence.
6. If the update sequence ended, the EMP swaps pointers — If the *Last Command* bit is set in the command, the EMP swaps between the module's pointer and the free provisioning area pointers in the shadow RAM NVM header via the EMPNVM_SRCTL, EMPNVM_SRDATA register set.
7. The EMP erases the next bank **—** It erases the contents of the next basic bank sectors, via the procedure described in Section 3.3.8.4.
8. The EMP copies shadow RAM into the next bank — It copies the shadow RAM into the next basic bank sector switch the exception of the *Validity* field, which is left to all ones. It uses the internal EMPNVM_FLCTL, EMPNVM_FLDATA register set.
9. The EMP checks the Flash write **—** 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 a previous step.
   a. If it is not (like Flash defect), return an error status — The EMP swaps back the two pointers in shadow RAM and the command completes to software (in the next step) with the Flash defect error bit set.
   — If the check was successful, The EMP validates the new bank and invalidates the old bank.The *Validity* field of the new bank is set to 01b. The EMP checks the *Validity* field is read as written in the Flash. If not, it goes to the previous sub-step.
   — The EMP toggles the state of the BANK1VAL bit in the GLNVM_GENS register to indicate that the non-valid bank became the valid one and visa versa.
   — The current (old) bank is in-validated by setting its *Validity* field to 00b.
10. The EMP posts an event completion on ARQ to software.
   a. If the NVM ownership timeout for write ends before reaching this step, the EMP flushes the remaining NVM update commands (if any) of the sequence, reporting a time out error status.
11. Software releases NVM ownership (see Section 7.10.12.6).
12. Software initiates a device reset (PCIR, CORER, or GLOBR) for loading the modifications into the XL710.
13. If the module modified is the manageability or the EMP settings module, then the EMP resets itself according to the flow described in Section 4.1.2.7 for loading the new settings into the XL710.

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

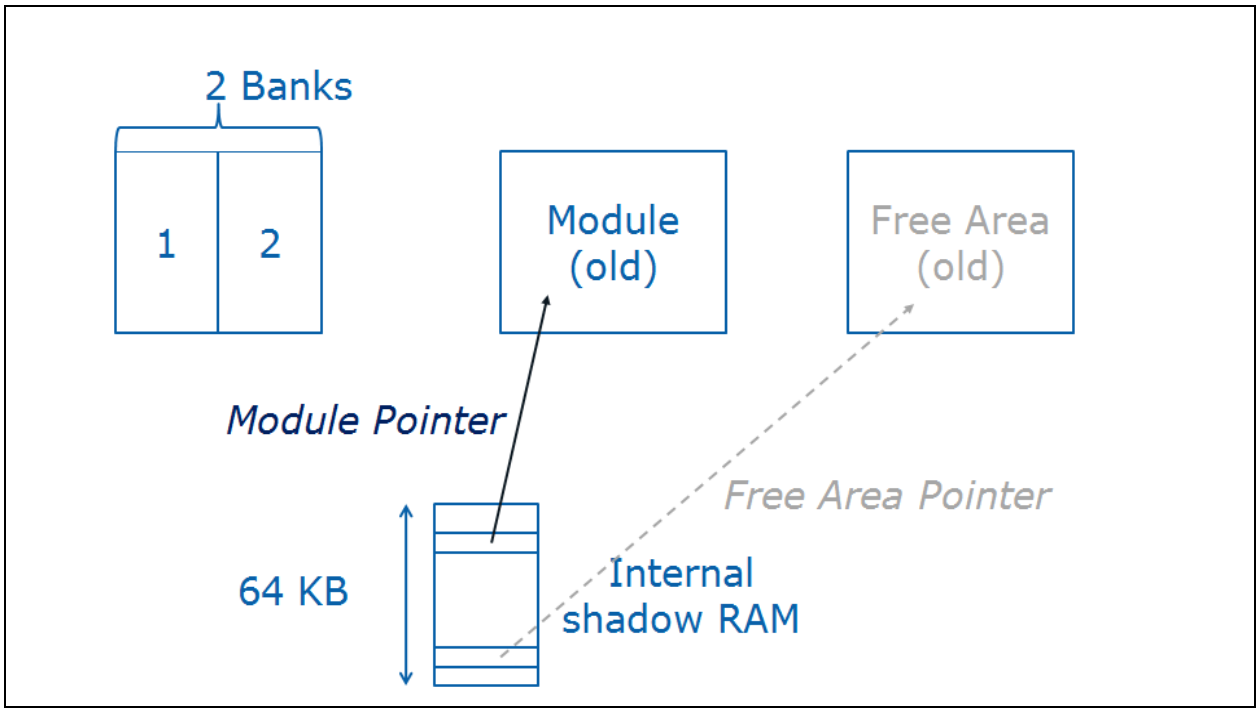


Figure 3-25. Initial state before the NVM update command

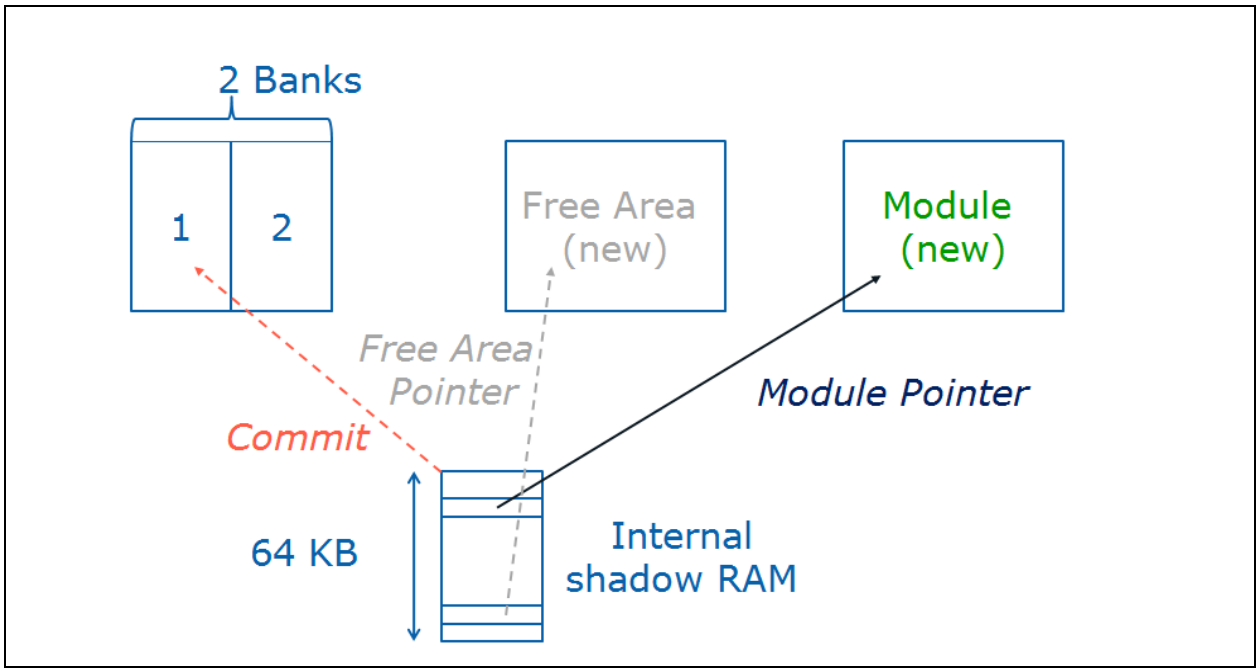
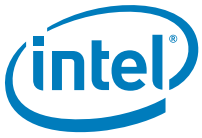


Figure 3-26. Final state after completion of the NVM update command



3.3.5.5 Reprogramming an authenticated module mapped outside shadow RAM

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

1. EMP Image (pointed by NVM word 0x0B)
2. Option ROM (pointed by NVM word 0x05)
3. PCIe Analog (pointed by NVM word 0x03)
4. PHY Analog (pointed by NVM word 0x04)

The first free area pointer (NVM word 0x40) is used for the bank swap of the first three modules. This free area must be sized exactly to 1160 KB.

The last two modules previously listed use the second free area pointer (NVM word 0x46) for their bank swap. This free area is made up of two consecutive 4 KB sectors.

A free area is used for the update of one single module at once.

1. Software takes ownership over the NVM resource for a write— Refer to Section 7.10.12.5.
2. Software issues one or several NVM Erase and/or Update commands — It posts the NVM Erase/ Update admin command to the EMP, providing the following parameters (refer to Section 3.3.10.3):
   - Address in the NVM of the pointer to one of the modules previously listed
   - Offset inside the module
   - Buffer in host memory
   - Buffer length (maximum supported is 4 KB)

   It is software's responsibility to erase a Flash sector prior to writing it.

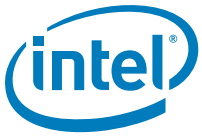
   If the *Last Command* bit is cleared in the command, it means that the command belongs to a complex NVM update operation made up of several elementary NVM update and/or erase commands that are posted in the admin queue. In between completions of elementary commands in a chain, other commands can be posted by a PF, besides other NVM-related commands. The entire 1160 KB free provisioning area must always be written.

   It is software's responsibility to issue the required NVM Erase commands prior to issuing NVM Update commands.

   The *Flush on Error* (FE) bit must be set for all commands of a sequence with the exception of the last one.
3. The EMP checks that the command is valid and posts a response to software — It performs the following command validity checks and posts a response (ACK/NACK) to software. If one check fails then the EMP flushes the remaining NVM Update commands (if any) of the sequence and 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 pointer location belongs to one of the modules in the list.
   b. The start/end offsets, once applied to the relevant free area, do not lead to addresses beyond the free area size.
   c. The start/end offsets, once applied to the relevant free area, do not spread over two consecutive 4 KB sectors
4. The EMP erases/writes the free provisioning area — According to the command, the EMP erases the 4 KB sector in the free provisioning area via the procedure described in Section 3.3.8.4 or writes the change required by the (elementary) command into it via the EMPNVM_FLCTL, EMPNVM_FLDATA register set.
5. If the update sequence did not end, the EMP posts an event completion on ARQ to software — If the *Last Command* bit is cleared in the (elementary) command, the EMP completes the command and then goes back to step 3. of this flow for the processing of the next (elementary) command of the sequence.

6. The EMP checks the security revision — It checks that the security revision (lad_srev field in CSS header) of the new module is greater or equal to the security revision of the current EMP module.
   a. If the check fails, the command is completed to software with the Security Revision Check Fails bit set. The EMP goes to step 15.
7. The EMP checks the XL710 *Blank NVM Device ID* and *Module ID* fields — it checks that the the XL710 *Blank NVM Device ID* field written in the new module is 0x154B. The EMP checks that the 30 LS bits of the *Module ID* field in the CSS header correspond to the module currently being updated.
   a. If one of the checks fails, the command is completed to software with the EACCES bit set. The EMP goes to step 15.
   b. Note that before authenticating the PCIe analog, PHY analog module, or 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. Refer to Section 6.1.5.2 for more details.
8. If an update sequence ended, the EMP checks CRC8 and authenticates the module — It reads the new module from the Flash, makes sure its CRC8 is valid and then authenticates its signature according to the procedure described in Section 3.3.9. This can be done in the course of writing to the Flash using the previous steps.
   a. If one of the checks fail, the command is completed to software with the relevant error bit set, either the Public Key Check Fails bit or the Module Signature Check Fails bit. The EMP goes to step 15.
9. The EMP swaps pointers — The EMP swaps between the module’s pointer and the free provisioning area pointers in the NVM header of the shadow RAM via the EMPNVM_SRCTL, EMPNVM_SRDATA register set.
10. The EMP erases next bank - It erases the contents of the next basic bank sectors, via the procedure described in Section 3.3.8.4.
11. The EMP copies shadow RAM into next bank — It copies the shadow RAM into the next bank sectors with the exception of the *Validity* field, which is left as all ones. It makes use of the internal EMPNVM_FLCTL, EMPNVM_FLDATA register set.
12. The EMP 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 at the previous step.
    — If not (such as Flash defect), return an error status - The EMP swaps back the two pointers in the shadow RAM and the command is completed to software (in the next step) with the Flash defect error bit set.
    b. If the check completed successfully, the EMP validates the new bank and invalidates the old bank
    — *Validity* field of the new bank is set to 01b. The EMP checks the *Validity* field is read as written into the Flash. If not, it goes to the previous sub-step.
    — 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 visa versa.
    — The current (old) bank is invalidated by setting its validity field to 00b.
13. If the module updated was the EMP image.
    a. If the last command in the flow completed with an error, then go to next step to report the error.
14. The EMP posts an event completion on ARQ to software — It posts a completion/response to the last admin command of the sequence.
15. Software releases NVM ownership. Refer to Section 7.10.12.6.
16. If the module updated was the EMP image, EMP resets itself according to the flow described in Section 4.1.2.7 and reloads from the *new* EMP image.
    a. Note that the *new* RO settings described in step 14.b. were loaded into the XL710 before the *new* EMP code started to run.

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



3.3.6 NVM clients and low level interfaces

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

Table 3-53 lists the different accesses to the NVM.

Table 3-53. Clients and access types to the NVM

Client	NVM Access Method	Accessed Performed Against	Logical Byte Address Range	NVM Access Interface (CSRs or Other)
VPD Software	Parallel (32-bits)	Shadow RAM	0x000000 - 0x0003FF from VPD module beginning	VPD Address and Data registers. Any write access is immediately pushed by the XL710 into the Flash.
PF Software	Parallel via memory (CSR) Bar (32-bits read, 8-bits write) Write allowed only in blank Flash programming mode	Flash Part	0x000000 - 0xFFFFFF	The address is relative to the beginning of the flash. See Section 12.1.2.1 for details of the flash offset within the CSR BAR.
	Parallel via expansion ROM BAR (32 bits read only)	Flash Part	0x020000 - 0xFFFFFF	This logical address range is relative to the beginning of the expansion ROM module. Write access to the Flash via expansion ROM BAR is not performed (silently dropped).
	Via AQC	Flash Part/ Shadow RAM	0x000000 - 0x00FFFF	NVM read, NVM erase and NVM update admin commands as described in Section 3.4.10.
	Parallel (16-bits)	Shadow RAM	0x000000 - 0x00FFFF	GLNVM_SRCTL and GLNVM_SRDATA registers for a read from shadow RAM logic. Write into the shadow RAM is allowed via these registers only in blank Flash programming mode.
	Bit banging (1-bit) allowed to software, only when in blank Flash programming mode	Flash Part	0x000000 - 0x01FFFF	GLNVM_FLA. Accessing this range via bit-banging should be avoided during normal operation as it might cause non-coherency between the Flash and the shadow RAM.
			0x020000 - 0xFFFFFFF	GLNVM_FLA.

3.3.6.1 Memory-mapped host interface

The Flash is read (or written when in blank Flash programming mode) by the XL710 each time the host CPU performs a read (or a write) operation to a memory location that is within the Flash address mapping or upon boot via accesses in the space indicated by the Expansion 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 XL710 is in blank Flash programming mode. The Flash size exposed is retrieved from the *GLPCI_LBARCTRL.FL_SIZE* CSR field, and is 8 MB by default (blank Flash programming mode). Refer to Section 11.1.1.2 for more details.

- The Expansion ROM Base Address register (PCIe Control register at offset 0x30). The module address space is always exposed to the expansion ROM BAR unless the Flash is blank. The Flash size exposed is retrieved from the *GLPCI_LBARCTRL.EXROM_SIZE* CSR field, and is 512 KB by default. The XL710 is responsible to map read accesses via the expansion ROM BAR to the physical NVM. Write attempts to the Flash through this BAR are not performed, they are silently dropped. The offset in the NVM of the expansion ROM module is defined by the PCIe expansion/option ROM pointer (Flash word address 0x05). This pointer is loaded by the XL710 from the Flash before enabling any access to the expansion ROM memory space.
  - When modifying the PXE driver section pointer in the NVM, it is required to issue a PCIe reset on which the updated offset is sampled by hardware.
  - If there is no valid NVM validity field in the two basic banks, then the expansion ROM BAR is disabled.

The XL710 controls accesses to the Flash when it decodes a valid access. Attempting memory-mapped write access to the Flash during normal programming mode is ignored. Out of range memory-mapped read access returns arbitrary data.

Note: Refer to Section 3.1.2.2.1 for details on memory/expansion ROM BAR access rules. The XL710 only supports byte writes to the Flash via the memory-mapped host interface (only when in blank Flash programming mode).

Flash read accesses are assembled by the XL710 each time the access is greater than a byte-wide access.

The XL710 byte reads or writes to the Flash take about 2 to 30 μs. The XL710 continues to issue retry accesses during this time.

During normal operation, the host should avoid memory-mapped accesses to the first two basic banks of the Flash because it might be non-coherent with the shadow RAM contents.

When in blank Flash programming mode, memory BAR access to the Flash while GLNVM_FLA.FL_REQ is asserted (and granted) is not supported. This can lead to a PCIe hang because a bit-banging access requires several PCIe accesses.

Prior to initiating an NVM read (or write) cycle via memory mapped access, PF software is required to take ownership over the NVM resources. Refer to Section 7.10.12.5.

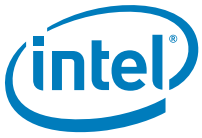
3.3.6.2 CSR mapped interface

All other low-level accesses to the NVM are made via dedicated registers sets, including the bit-banging access mode available in blank Flash programming mode.

Prior to initiating a shadow RAM read cycle via GLNVM_SRCTL and GLNVM_SRDATA registers, PF software is required to take NVM ownership.

3.3.7 Flash access contention

Flash read accesses initiated through PFs might occur concurrently to the EMP modifying the NVM contents. The XL710 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 EMP accesses, these entities are required to make use of the NVM ownership taking/release flows for any read or write access to NVM. Refer to Section 7.10.12.5 and to Section 7.10.12.6 for more details. This is also useful to avoid the timeout of the PCIe transaction made to a memory mapped Flash address while the Flash is currently busy with a long sector erase operation.

However, two software entities cannot use the NVM ownership acquiring/release mechanisms: BIOS and VPD software.

- Since VPD software accesses only the VPD module, which is located in the first valid bank of the NVM, VPD accesses are always performed against the shadow RAM first. In this case, the EMP must take/release ownership over the NVM as if it was the originator of the Flash access. It is then hardware/EMP's responsibility to update the NVM according to the Flash update sequence described in Section 3.3.5.2.
- No contention can occur between BIOS and any other software entity (VPD included) as it accesses the NVM while the operating system is down.

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 expansion ROM BAR returns 0xDEADBEEF.

- It is assumed that the expansion ROM signature check performed by BIOS fails in this case.
- The EMP must avoid initiating sector erase operations at boot time.
- It is assumed and recommended that users do not attempt to update the NVM contents via the BMC while the system is re-booting.
- The BMC should delay PERST# de-assertion or boot running until after the BMC completed any OOB accesses to Flash memory. It is required to route the wake-up signal from the standby button to the BMC and not to the chipset. The BMC issues a system reboot signal to the chipset only after any NVM write access completes.
- If a system reboot is issued by a local user running on the host, there is no technical way to avoid contention in this case.

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

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

3.3.8 NVM access procedures

Any software read/write or EMP 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 EMP in between two ownership takings.

Refer to Section 7.10.12.5 and Section 7.10.12.6 for the NVM ownership taking/releasing procedures as well for the associated timeouts.

3.3.8.1 Auto-load

a. On POR events:

— The XL710 starts a new general auto-load timer with a default hardware value of 300 ms.

— The XL710 starts auto-load of the POR registers auto-load module during which NVM defaults are loaded into the two auto-load timers:

- General auto-load timer in GLNVM_ALTIMERS. GEN_ALTIMER
- PCIe auto-load timer in GLNVM_ALTIMERS. PCI_ALTIMER

— If the auto-load general timer ends before the module auto-load completes, go to step 7.d. Otherwise, go to next step (7.b).

b. Each time the XL710 starts auto-load of one of the PCIe modules, the PCIe autoload timer is re-starts (15 ms by default). It stops each time a module auto-load completes. The following PCIe modules are affected:

- RO PCIR registers auto-load
- RO PCIe LCB
- PCIR register auto-load
- PCIe ALT auto-load

c. Each time the XL710 starts auto-load of any other NVM module, the general auto-load timer re-starts. It stops each time a module auto-load completes.

d. If an auto-load timer ends, the XL710 takes the following actions:

— It raises a new *Auto-load-Error* (AE) bit in GLNVM_ULT register (one such bit per NVM register auto-load module) to report the module for which auto-load has not completed.

— It stops parsing the current NVM module, asserts the HW Done / Conf Done indication bits for the corresponding module and goes to the next module to be parsed, if any; otherwise, it returns to its idle state.

3.3.8.2 Flash erase flow by the host

This flow is available to the software PF device driver only when the XL710 is in blank Flash programming mode.

1. Poll the *FL_BUSY* flag in the GLNVM_FLA register until cleared.
2. Set the *Flash Device Erase* bit (FL_DER) in the GLNVM_FLA register or the *Flash Sector Erase* bit (FL_SER) together with the Flash sector index to be erased (FL_SADDR).

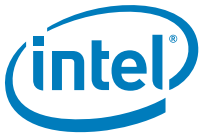
Hardware gets the Erase command from GLNVM_FLA register and sends the corresponding Erase command to the Flash. The erase process then finishes by itself. Software should wait for the end of the erase process before any further access to the Flash. This can be checked by polling the GLNVM_FLA.FL_BUSY bit.

3.3.8.3 Software access to NVM via the bit banging interface

This flow is available to PF software only when the XL710 is in blank Flash programming mode.

To directly access the Flash, software/EMP should follow these steps:

1. Write a 1b to the *Flash Request* bit (GLNVM_FLA.FL_REQ).



2. Read the *Flash Grant* bit (GLNVM_FLA.FL_GNT) until it becomes 1b. It remains 0b as long as there are other accesses to the Flash.
3. Write or read the Flash using the direct access to the 4-wire interface as defined in the GLNVM_FLA register. The exact protocol used depends on the Flash placed on the board and can be found in the appropriate datasheet.
4. Write a 0b to the *Flash Request* bit (GLNVM_FLA.FL_REQ).
5. Following a write or erase instruction, software/EMP should clear the *Request* bit only after it has checked that the cycles were completed by the NVM. This can be checked by reading the BUSY bit in the Flash device STATUS register. Refer to the Flash datasheet for the opcode used for reading the STATUS register.

Note: If software uses the bit banging interface during run time, it should adhere to the following rules:

- Gain access first to the Flash using the flow described in Section 7.10.12.5.
- Minimize the GLNVM_FLA.FL_REQ setting for a single byte/word/Dword access or other method that guarantees a fast enough release of GLNVM_FLA.FL_REQ.

3.3.8.4 Software access to the shadow RAM via GLNVM_SRCTL and GLNVM_SRDATA registers

3.3.8.4.1 Shadow RAM read access

The host initiates a read cycle to the shadow RAM by the following:

1. Poll the *Done* bit in the GLNVM_SRCTL register until its set.
2. Write the address to be read, clear the *Write* bit, and set the *Start* bit in the GLNVM_SRCTL register.

As a response, hardware executes the following steps:

1. The XL710 reads the data from the shadow RAM.
2. Puts the data in *RDDATA* field of the *GLNVM_SRDATA* register.
3. Sets the *Done* bit in the *GLNVM_SRCTL* register.

Note: Any word read this way is not loaded into the XL710's Internal Registers. This happens only at an hardware auto-load event.

3.3.8.4.2 Shadow RAM write access

The host initiates a write cycle to the shadow RAM as follows:

1. Poll the *Done* bit in the *GLNVM_SRCTL* register until its set.
2. Write the data word in the *WRDATA* field of the *GLNVM_SRDATA* register.
3. Write the address, the *Write* bit, and the *Start* bit in the *GLNVM_SRCTL* register.

As a response, hardware executes the following steps:

1. The XL710 writes the data to the shadow RAM.
2. The XL710 sets the *Done* bit in the *GLNVM_SRCTL* register.

Note: Any word written into shadow RAM is not loaded into the XL710's internal registers. This happens only at a hardware auto-load event.

3.3.8.5 Flash programming procedure via the memory interface

This flow is available to PF software only when the XL710 is in blank Flash programming mode.

Software initiates a write cycle via to the Flash via the Memory BAR as follows:

1. Poll the *FL_BUSY* flag in the GLNVM_FLA register until cleared.
2. Write the data byte to the Flash through the Memory BAR.
3. Repeat the steps 2 and 3 if multiple bytes should be programmed.

As a response, hardware executes the following steps for each write access:

1. Set the *FL_BUSY* bit in the GLNVM_FLA register.
2. Initiate autonomous write enable instruction.
3. Initiate the program instruction right after the enable instruction.
4. Poll the Flash status until programming completes.
5. Clear the *FL_BUSY* bit in the GLNVM_FLA register.

Note: Software must erase the sector prior to programming it. Refer to Section 3.3.8.2.

Memory BAR write access to the Flash while GLNVM_FLA.FL_REQ is asserted (and granted) is not supported. This can lead to a PCIe hang as a bit-banging access requires several PCIe accesses.

3.3.8.6 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 3.3.11 for more details.

3.3.9 NVM authentication procedure

The NVM update integrity feature ensures that only Intel-approved firmware code (or another protected NVM module) is able to be updated on XL710 devices after manufacturing. This procedure is performed each time an attempt is made to update one of the protected modules. Refer to NVM update flows in Section 3.3.5 for more details.

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-bits 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, the EMP must first verify that the RSA *Modulus* and RSA *Exponent* fields in the new module loaded are identical to those in the current EMP image. If the RSA *Modulus* and *Exponent* fields are the same, the EMP decrypts the digital signature using the 2048-bit RSA *Modulus* and *Exponent* fields stored in the manifest of the old EMP image to extract the expected SHA256 hash of content (stored hash). The 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.

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 XL710's Flash by the EMP.

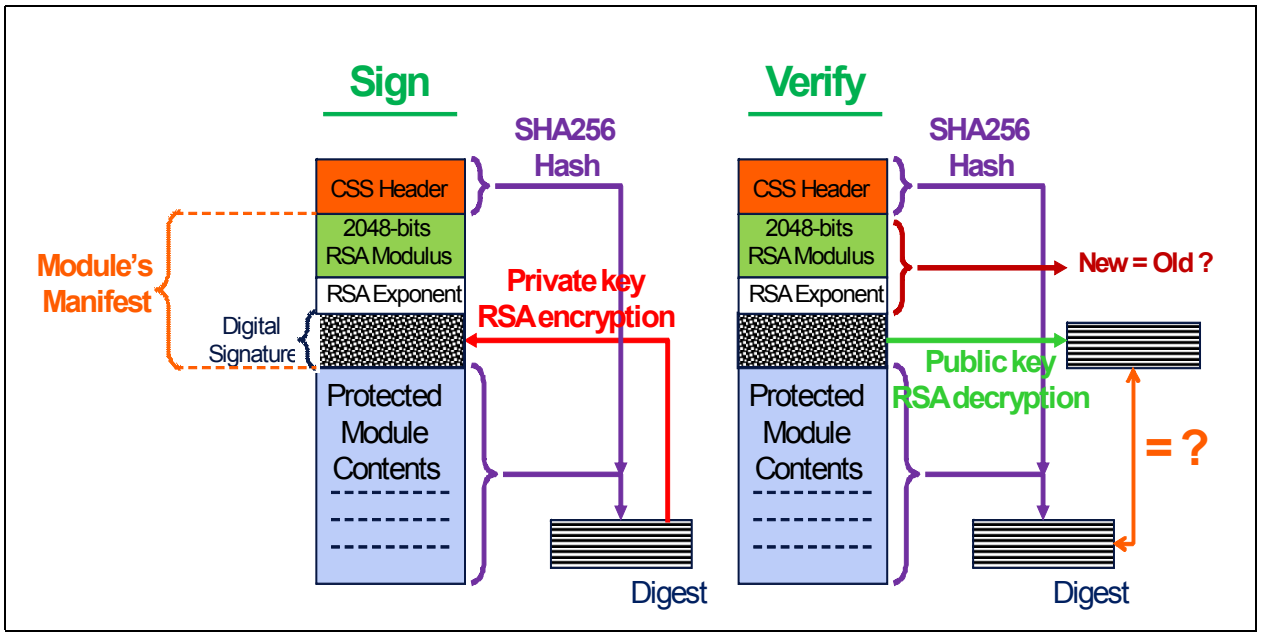


Figure 3-27. Sign and verify procedures for authenticated NVM modules

3.3.9.1 Digital signature algorithm details

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 XL710.

The algorithms used are described in the following locations:

- PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002 - www.rsa.com
- SHA family definition - http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.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 3-27 starts with the XL710's Blank NVM Device ID word of the NVM header described in Section 6.1.5.2 and ends with the last word of the 1160 KB long EMP image area, regardless of the size of the EMP code and to the presence and size of a RO commands section at the last sector of the EMP image area.

3.3.9.2 Intel key generation and Intel code signing system

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

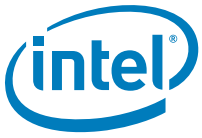
Intel's CSS follows the PKCSv1.5 format with 2048-bit RSA keys and SHA-256 hash. The CSS algorithm requires a standard manifest header to appear at the beginning of all signed modules. The manifest header (represented in C syntax) is as follows:

```
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;                // Reserved CSS field
   uint32 keySize;             // Reserved CSS field
   uint32 modulusSize;         // Reserved CSS field
   uint32 exponentSize;        // Reserved CSS field
   uint32 lad_srev;            // LAD-specific security revision field
   uint32 reserved;         // Reserved for future use
uint32 lad_fw_entry_offset;// LAD-specific offset from start of CSS header of the firmware
code in WORDS
int32 reserved; // Mandatory field, in use by Springville project
uint32 lad_image_unique_id; // LAD-specific unique identifier for the specific NVM image version.
Taken from NVM Software Reserved Words 22, 23 (a.k.a. EETRACK ID)
uint32 lad_module_id; // LAD-specific and per device unique identifier for NVM module. The 30 LS
bits shall be one of the following values:

0x1 for the EMP image

0x3 for PCIe analog

0x4 for PHY analog

0x5 for Option ROM
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 XL710 CSS header includes a *Security Revision* (lad_srev) field. Th*is* 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 XL710 does not allow an NVM image roll back to an earlier version because this might expose previously known vulnerabilities).

Note: 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* and *lad_nvmsize* fields give the entire module size including the CSS header itself. It is expressed in Dwords and in words, respectively. As stated in Section 3.3.9.1, these fields are always set to 0x20000 and 0x40000, respectively.

3.3.9.3 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 XL710 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 XL710. Each module is appended by its own digital signature:

- EMP image
- PCIe Analog PHY
- PHY Analog
- Option ROM

3.3.9.4 Software requirements

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

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

CELO (or equivalent manufacturing diagnostics tool) must implement a test to check the GLNVM_FLA *Locked* bit state as part of manufacturing qualification. If after the NVM is programmed, and the XL710 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 build into the hardware as well as the saved CSS manifest in the NVM image.

3.3.9.5 Manufacturing requirements

3.3.9.5.1 End-of-line verification

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

3.3.9.5.2 Post-manufacturing physical modification countermeasures

For add-in NICs and related form factors, applying a conformal epoxy encapsulation coating around the edges of NVM components (Flash and EEPROM) during manufacturing prevents them from being easily removed.

In addition, epoxy is used to coat any traces or vias that could enable direct-probing of the XL710. The testing requirement for these coatings must be that they cannot be removed using a standard hobby knife or other hand-held tool.

Note that protection by epoxy encapsulation is just a recommendation, not a security requirement for the XL710.

3.3.10 NVM access admin commands

NVM access commands are not supported when the XL710 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 7.10.12.5.

Table 3-54. NVM access admin commands

Command	Opcode	Brief Description	Detailed Description
NVM Read	0x0701	Read a segment from the NVM into a host buffer	Section 3.3.10.1
NVM Erase	0x0702	Erase consecutive 4 KB sectors of the Flash	Section 3.3.10.2
NVM Update	0x0703	Write a segment of the NVM from a host buffer	Section 3.3.10.3

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

3.3.10.1 NVM read

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 3-55. NVM read admin command

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0701	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value/ VFID	6-7		Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value, will be copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value, will be copied by the EMP into the completion of this command.

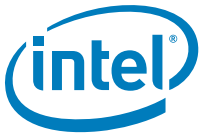


Table 3-55. NVM read admin command (Continued)

Name	Bytes.Bits	Value	Remarks
Command Flags	16		NVM Access Admin Command Parameters bit 0 - Last command bit, used to notify EMP that this is the last admin command of a sequence. bits 1:7 - Reserved, must be zeroed.
Module_pointer	17		Module pointer location in words from the NVM beginning. A value of 0x0000 means that the command is performed over the Flash part seen as a flat memory. In any case, the read is always performed against the Flash part and never from the shadow RAM. The GLNVM_SRCTLand GLNVM_SRDATA register set must be used for shadow RAM reads. This flat memory addressing mode must be the only mode used when accessing the FCoE scratch pad area.
Length	18-19		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.
Offset	20-23		Byte 23= Reserved, must be zeroed. Bytes 22:20= 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.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

Table 3-56. NVM read response

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0701	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
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 that is copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value that is copied by the EMP into the completion of this command.
Reserved	16		Reserved.
Module_pointer	17		Module pointer location and copied from the command.
Length	18-19		Length to be read and copied from the command. 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 modules's end.
Offset	20-23		Byte 23= Reserved, must be zeroed. Bytes 22:20 = Offset in the module, copied from the command.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

3.3.10.2 NVM erase

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

Table 3-57. NVM erase admin command

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.1 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 that is copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value that is copied by the EMP into the completion of this command.
Command_flags	16		NVM Access Admin Command Parameters. bit 0 = Last command bit used to notify the EMP that this is the last admin command of a sequence. bits 7:1 = Reserved, must be zeroed.
Module_pointer	17		Module pointer location in words from the NVM beginning. A value of 0x0000 means that the command is performed over the Flash part seen as a flat memory. This flat memory addressing mode must be the sole mode used when erasing the FCoE scratch pad area. Attempting to erase the basic banks or RO modules area is not allowed in normal operating mode. Besides 0x0000, only the address of a free provisioning area module pointer can be listed here.
Length	18-19		Length of the section to be erased, which is expressed in 4 KB sector units from the offset in the module. The module's beginning must be aligned to a 4 KB sector for the command to be valid. A value of 0xFFFF means that the last 4 KB sector to be erased is the last sector of the free provisioning area (if byte 17 was not set to 0x0000).
Offset	20-23		Byte 23= Reserved, must be zeroed. Bytes 22: 20 = Offset in the module, which is expressed 4 KB sector index from the pointed module's beginning.
Reserved	24-27		Reserved.
Reserved	28-31		

This command 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 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.

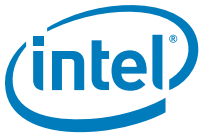


Table 3-58. NVM erase response

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.2 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 free provisioning area or attempt to erase sectors from the basic banks. EINVAL = Out of range offset/length beyond the free area module's limits. ENOENT = The module pointed is not aligned with an 4 KB sector beginning. 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 that is copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value that is copied by the EMP into the completion of this command.
Reserved	16-31		Reserved

Table 3-59. NVM erase completion (on ARQ)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.2 for details.
Opcode	2-3	0x0702	Command opcode.
Datalen	4-5		Reserved.
Return Value	6-7		Return Value: 0x0 = No error (success). EIO = Flash defect.
Cookie High	8-11	Cookie	Opaque value that is copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value that is copied by the EMP into the completion of this command.
Reserved	16		Reserved.
Module_pointer	17		Copied from the command.
Length	18-19		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 modules's end in 4 KB units.
Offsets	20-23	0x0	Byte 23= Reserved, must be zeroed. Bytes 22:20= Copied from the command.
Reserved	24-27		Reserved.
Reserved	28-31		

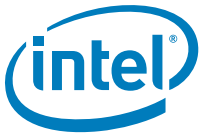
3.3.10.3 NVM Update

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

Table 3-60. NVM update admin command

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 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 that is copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value that is copied by the EMP into the completion of this command.
Command_flags	16		NVM Access Admin Command Parameters. Bit 0 = Last command bit used to notify EMP that this is the last admin command of a sequence. Bits 6:1 = Reserved, must be zeroed. Bit 7 = Flash-only bit. When bit 7 is set, any flat write (null Module_pointer) into the first 128 KB of the Flash is written directly to the Flash and not in shadow RAM. The bit has no effect when attempting a flat write outside the first 128 KB of the Flash.
Module_pointer	17		Module pointer location in words from the NVM beginning. Attempting to write a RO module is not allowed. A value of 0x0000 here means that the command is performed over the Flash part seen as a flat memory. No reset or pointer switch is initiated by the XL710 in such a case. Any flat write attempt to the first 64 KB of the Flash is performed against the shadow RAM first, and then dumped to the next basic bank in the Flash (see Section 3.3.5.3). Any flat write attempt to the second 64 KB of the Flash is rejected as for write attempts to a RO module. A flat write attempt to the area of a RO module or to a RO word is rejected as well. This flat memory addressing mode must be the sole mode used when updating the FCoE scratch pad area.
Length	18-19		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.
Offset	20-23		Byte 23 = : Reserved, must be zeroed. Bytes 22:20= Offset, which is expressed in bytes from the pointed module's beginning. This is the byte offset of the first byte to be written.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

This command 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 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 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 3-61. NVM update response

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0703	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value	6-7		Return Value. 0x0 = No error (success). EPERM = The module pointer location 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. 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.
Cookie High	8-11	Cookie	Opaque value that is copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value that is copied by the EMP into the completion of this command.
Reserved	16-31		Reserved.

Table 3-62. NVM update completion (on ARQ)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0703	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value	6-7		Return Value. 0x0 = No error (success). EIO = Flash defect. EACCES = Security check failed: • Public key check failed • Module digest check failed Module security revision check failed • Device ID check failed • Module ID check failed
Cookie High	8-11	Cookie	Opaque value that is copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value that is copied by the EMP into the completion of this command.
Reserved	16		Reserved.
Provisioning_pointer	17		Location in words from the NVM beginning with the free provisioning pointer used for the command. A value of 0x0000 is returned if the command was performed against the shadow RAM.
Length	18-19		Copied from the command.

Table 3-62. NVM update completion (on ARQ)

Name	Bytes.Bits	Value	Remarks
Offsets	20-23		Byte 23= Reserved, must be zeroed. Bytes 22:20= Copied from the command.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

3.3.11 VPD Support

The Flash image can contain an area for VPD. This area is managed by the OEM vendor and does not influence the behavior of hardware. Word 0x2F of the Flash image contains a pointer to the VPD area in the Flash. A value of 0b in the GLPCI_CAPCTRL.VPD_EN register bit means VPD is not supported and the VPD capability does not appear in the configuration space. The register bit must be set to 1b in NVM only once the VPD area has been programmed. Refer to Section 3.3.5.2.1.

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

Table 3-63. Small resource structure

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

Table 3-64. Large resource structure

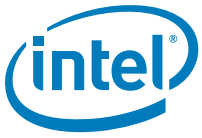
Offset	0	1 — 2	3 — n
Content	Tag = 1xxx,xxxxb (Type = Large(1), Item Name = xxxxxxxx)	Length	Data

The XL710 parses the VPD structure during the auto-load process following PCIe reset in order to detect the read only and read/write area boundaries. The XL710 assumes the following VPD fields with the limitations listed:

Table 3-65. VPD structure

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 XL710 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 3-65, 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 3-65. Write accesses to a RO area or any accesses outside of the VPD area via this structure are ignored. If the VPD *Write Enable* field is set to 1b in the NVM Security Control word, the entire VPD area can be modified via the NVM Update AQ command. Otherwise, the command is completed with an error status.
- VPD area must be mapped into the first valid basic bank of the Flash.
- VPD software does not check the NVM ownership before attempting to access the Flash via dedicated VPD registers (refer to Section 12.3.4). 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 3.3.5.2 for more details.

3.4 General Purpose I/O (GPIO)

The XL710 has a total of 22 GPIOs pins that can be configured as SDPs, LED drivers or dedicated hardware functions such as for connecting to external PHYs or IEEE 1588 auxiliary devices or driving Reset on LAN (RoL). The GPIO pins can also be configured to be associated with any of the physical ports. The following sections describe the possible configurations for the GPIO pins. Many of the GPIO pins are reserved for specific use and hence named by default as SDP, LED or GPIO signals (See Section 2.0). This offers the flexibility to configure any of the GPIO pins, irrespective of their names, to different modes and associated with different ports.

The GPIO pin functionality and other attributes such as I/O direction, default value, interrupt mode etc., can be configured through Global GPIO Control registers. See the Table 3-66 for the list of GPIO pins and their corresponding configuration control registers. The GLGEN_GPIO_CTL has a PIN_FUNC field to configure the pin functionality such as SDP, LED, RoL or 1588 Timesync modes. The PRT_NUM field is used to configure the port number associated with the GPIO pin. The INT_MODE field is used to configure the GPIO pin to generate an interrupt on the rising edge, falling edge or on both edges. The PHY_PIN_NAME is used to indicate how the GPIO is connected to an external PHY. Seethe GLGEN_GPIO_CTL register description for additional configuration fields and options. The default value for this register is loaded from the NVM and is typically dependent on the board layout/configuration, number of ports, and PHYs present on the board.

Table 3-66. GPIO pin configuration registers

GPIOn Index (n= 0..29)	Pin Name	GPIO Register
GPIO0	GPIO_0	GLGEN_GPIO_CTL[0]
GPIO1	GPIO_1	GLGEN_GPIO_CTL[1]
GPIO2	GPIO_2	GLGEN_GPIO_CTL[2]
GPIO3	GPIO_3	GLGEN_GPIO_CTL[3]
GPIO4	SDP0_0	GLGEN_GPIO_CTL[4]
GPIO5	SDP0_1	GLGEN_GPIO_CTL[5]
GPIO6	SDP0_2	GLGEN_GPIO_CTL[6]
GPIO7	SDP0_3	GLGEN_GPIO_CTL[7]
GPIO8	SDP1_0	GLGEN_GPIO_CTL[8]
GPIO9	SDP1_1	GLGEN_GPIO_CTL[9]
GPIO10	SDP1_2	GLGEN_GPIO_CTL[10]
GPIO11	SDP1_3	GLGEN_GPIO_CTL[11]
GPIO12	SDP2_0	GLGEN_GPIO_CTL[12]
GPIO13	SDP2_1	GLGEN_GPIO_CTL[13]
GPIO14	SDP2_2	GLGEN_GPIO_CTL[14]
GPIO15	SDP2_3	GLGEN_GPIO_CTL[15]
GPIO16	SDP3_0	GLGEN_GPIO_CTL[16]
GPIO17	SDP3_1	GLGEN_GPIO_CTL[17]
GPIO18	SDP3_2	GLGEN_GPIO_CTL[18]
GPIO19	SDP3_3	GLGEN_GPIO_CTL[19]

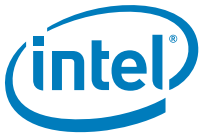
The following registers are used to read and write to the GPIO pins. The GLGEN_GPIO_STAT register is used to read the status of the GPIO pins. This register returns the actual value (0b = high, 1b = low) on the pin. The GLGEN_GPIO_TRANSIT register is used to latch any transition (low-to-high or high-to-low) on the GPIO pins since the last time the register was cleared. The GLGEN_GPIO_SET register is used to set the value (0b = low, 1b = high) of the GPIO output pins when configured as an SDP. The PF (software) is expected to write to a GPIO pin only if it owns it. Note that in MFP mode, software is expected to request ownership of GPIO resources before writing to it. See Section 7.10.12.5 and Section 7.10.12.6 for details on requesting ownership of shared resources. When any of the GPIO pins are configured to generate interrupts, the PFINT_GPIO_EN and EMPINT_GPIO_EN registers are used to enable or disable the interrupts to the PF or EMP that owns the pins.

3.4.1 LEDs

The XL710 designates eight pins named LED0_0 through LED3_3 for driving LEDs for ports 0 through 3 as described in Section 2.2.6.1. However, depending on the board layout, these might be replaced or complemented with any of the GPIO pins by configuring the GPIO to be a LED driver through the GLGEN_GPIO_CTL register as described in Section 3.4.3.

In order to configure a GPIO pin as an LED driver using the GLGEN_GPIO_CTL register:

- Set PIN_FUNC to LED



- Set PIN_DIR to output
- Use the PORT_NUM field to assign a port number to the LED pin

The output polarity of the LED pin (active high or active low) can be configured through the LED_INVRT field and the blink mode (blinking versus steady) is configured through the LED_BLINK field. The LED outputs can be individually configured to indicate a particular event, state, or activity by using the LED_MODE field. See Table 3-67 for more information on configuring the LED source mode field. The hardware default configuration for GPIO pins configured as LED outputs can be specified via NVM fields thereby supporting LED displays configurable to a particular OEM preference. Note that at run-time there is usually no need to override the LED setting, the only exception would be to force a LED to blink in order to physically identify a port.

Table 3-67. LED source modes

LED_MODE Value	Mode	Condition[1]
LED_MODE[4] = 0		
0000b	LED_OFF	Always off.
0001b	LINK	True while link is held.
0010b	LINK_40G	True while the XL710 has 40 Gb/s link.
0011b	LINK_10G	True while the XL710 has 10 Gb/s link.
0100b	LINK_1G	True while the XL710 has 1 Gb/s link.
0101b	LINK_100	True while the XL710 has 100 Mb/s link.
0110b	LINK_40G/10G	True while the XL710 has 40 Gb/s or 10 Gb/s link.
0111b	LINK_40G/1G	True while the XL710 has 40 Gb/s or 1 Gb/s link.
1000b	LINK_10G/1G	True while the XL710 has 10 Gb/s or 1 Gb/s link.
1001b	LINK_10G/100M	True while the XL710 has 10 Gb/s or 100 Mb/s link.
1010b	Combined Ports Activity	Active when any one of the XL710's port has an established link with packets being transmitted or received. In this mode LED_BLINK must be set.
1011b	Reserved	Reserved.
1100b	LINK_ACT	Asserted steady when link is established and there is no transmit or receive activity. Blinking when there is link and receive or Transmit activity. In this mode LED_BLINK must be cleared at 0b.
1101b	MAC_ACT	Active when link is established and packets are being transmitted or received. In this mode, the LED_BLINK must be set.
1110b	FILTER_ACT	Active when link is established and packets are being transmitted or received that passed MAC filtering. In this mode, the LED_BLINK must be set.
1111b	LED_ON	Always true. Overrides all other settings.
LED_MODE[4] = 1 (Firmware LED) ***Note:*** The following LED Modes do not support LED_BLINK and the configuration is ignored when a LED is configured as one of the following		
0000b	AGGREGATE_LINK_10G	Asserted when ALL enabled ports have 10G link
0001b	COMBINED_LINK	Asserted when ANY of the enabled ports has link
0010b	COMBINED_LINK_40G	Asserted when at least one of the enabled ports has 40G link
0011b	COMBINED_LINK_10G	Asserted when at least one of the enabled ports has 10G link

Table 3-67. LED source modes (Continued)

LED_MODE Value	Mode	Condition[1]
0100b	COMBINED_LINK_1G	Asserted when at least one of the enabled ports has 1G link
0101b	COMBINED_LINK_10G_NOT_ALL	Asserted when at least one of the enabled ports has 10G link but not ALL ***Note:*** This LED will never be asserted when only one port is enabled
Other	Reserved	

1. When the condition is true, the LED might blink or be constantly on, depending on the value of LED_BLINK field.

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

The LED_BLINK bit controls whether the LED should blink while the LED source is asserted. The blink control can be especially useful for ensuring that certain events, such as ACTIVITY indication, cause LED transitions, which are sufficiently visible to the human eye. The global blink mode bit in GLGEN_LED_CTL register is used to select between blinking every 200 ms or every 83 ms.

Note: LED_ON is not affected by the LED_BLINK control.

Note: The LINK/ACTIVITY source functions slightly different from the others when BLINK is enabled. The LED is:

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

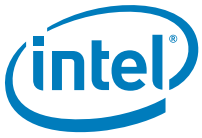
Note: In both MFP mode and non-MFP mode, only one software device driver should override any of the LED settings at any time (host-wide). Since this is an administrator specified override, hardware does not enforce this. Typically, there is no need to override the LED hardware defaults loaded from the NVM in a specific board configuration.

3.4.2 Software-Definable Pins (SDPs)

The XL710 allows any of the GPIO pins to be configured as SDPs through the GLGEN_GPIO_CTL register as described in Section 3.4.3. The XL710 has designated 16 GPIO pins, SDP0_0 (GPIO4) through GPIO3_3 (GPIO19), as SDPs for use with ports 0 through 3, as described in Section 2.2.6.2. However, depending on the board layout, any of the GPIO pins, irrespective of their name, could be configured as SDPs. In order to configure a GPIO pin for SDP use, set PIN_FUNC to SDP and PIN_DIR to either input or output in the GLGEN_GPIO_CTL register. Use the PORT_NUM field to assign a port number to be associated with the SDP.

The XL710 SDP pins can be used for hardware connectivity to low-speed, optical-module interfaces, external PHY control or software controllable purposes. The SDP pins can also be configured for use as external interrupt sources. The SDP pins, port ownership, direction and their functions are bound to specific board layout/configuration. The default values for GPIO control registers (GLGEN_GPIO_CTL[n]) are loaded from the NVM. Typically, there is no need for software to change the default configuration assigned for a specific board layout. Pins assigned for PHY management are owned by firmware and should not be accessed directly by software.

The XL710 SDP pins can also be configured for use as general purpose external interrupt sources (GPI). To act as GPI pins, the desired pins must be configured as inputs and enabled by the INT_MODE field in the GLGEN_GPIO_CTL register. The INT_MODE field can be used to configure the pin to generate an



interrupt on the rising edge, falling edge or on both edges. Rising or falling edge detection occurs by comparing values sampled at the internal clock rate, as opposed to an edge-detection circuit. When detected, a GPIO interrupt is indicated in the PFINT_ITR0 register. The PFINT_GPIO_ENA register is used to enable interrupts to PFs (software) and the EMPINT_GPIO_ENA register is used to enable an interrupt to the EMP (firmware). GLGEN_GPIO_STAT and/or GLGEN_GPIO_TRANSIT registers can be used to read the status of the GPIO pins that caused the interrupt. Software is expected to enable interrupts only for pins that are owned by the PFs.

Software or firmware can read/write to a GPIO/SDP pin as described in Section 3.4.3.

Note: In MFP mode, the software device driver must take ownership of a port's SDP pin before using it. To take ownership, the software device driver must request ownership over the GPIO/SDP resources (See Section 7.10.12.5 and Section 7.10.12.6 for details about requesting ownership of shared resources). Note that The XL710 firmware can use any SDP as an interrupt source.

The proposed use of SDPs for controlling external PHY devices or modules is explained in Section 3.2.2.9.

3.4.3 Global GPIO pins

The XL710 has designated six general purpose I/O pins as Global GPIO pins named GPIO_0 through GPIO_5, as listed in Section 2.2.6.3. These pins are not assigned for a specific usage or port. The mode and port association for these pins can be configured as described in Section 3.4. These pins can be used as SDP, LED, or 1588 Timesync modes or as general purpose interrupt sources. System designers have the flexibility to assign these GPIO pins for specific use depending on the board configuration.

3.5 BMC interconnects

The XL710 supports three sideband interfaces:

- SMBus
- Network Controller-Sideband (NC-SI)
- PCIe (together with MCTP)

The SMBus and NC-SI interfaces are described in the sections that follow. The PCIe interface is described in Section 3.1.

3.5.1 SMBus

SMBus is an optional interface for pass-through and/or configuration traffic between an external BMC and the XL710. The SMBus channel behavior and the commands used to configure or read status from the XL710 are described in Section 10.5.

The XL710 also enables reporting and controlling itself using the MCTP protocol over SMBus. The MCTP interface is used by the BMC to only control the NIC and not for pass through traffic. All network ports are mapped to a single MCTP endpoint on SMBus. For further information, see Section 10.7.

3.5.1.1 Channel behavior

The SMBus specification defines a maximum frequency of 100 KHz. However, when acting as a slave, the XL710 can receive transaction with a clock running at up to 1 MHz. When acting as a master, it can toggle the clock at 100 KHz, 400 KHz or 1 MHz. The speed used is set by the *SMBus Connection Speed* field in the *SMBus Notification Timeout and Flags* NVM word.

3.5.2 NC-SI interface

The NC-SI interface in the XL710 is a connection to an external BMC defined by the DMTF NC-SI protocol. It operates as a single interface with an external BMC, where all traffic between the XL710 and the BMC flows through the interface.

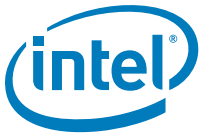
The XL710 supports the standard DMTF NC-SI protocol for both pass-through and control traffic as defined in Section 10.6.

3.5.2.1 Electrical characteristics

The XL710 complies with the electrical characteristics defined in the NC-SI specification.

The XL710’s NC-SI behavior is configured at power-up in the following manner:

- The *Multi-Drop NC-SI* NVM bit defines the NC-SI topology (point-to-point or multi-drop; the default is point-to-point).



The XL710 dynamically drives its NC-SI output signals (NC-SI_DV and NC-SI_RX) as required by the sideband protocol:

- At power-up, the XL710 floats the NC-SI outputs.
- If the XL710 operates in point-to-point mode, it starts driving the NC-SI outputs some time following power-up.
- If the XL710 operates in a multi-drop mode, it drives the NC-SI outputs as configured by the BMC.

3.5.2.2 NC-SI transactions

The NC-SI link supports both pass-through traffic between the BMC and the XL710 LAN functions, as well as configuration traffic between the BMC and the XL710's internal units as defined in the NC-SI protocol. See Section 10.6.2 for more details.

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

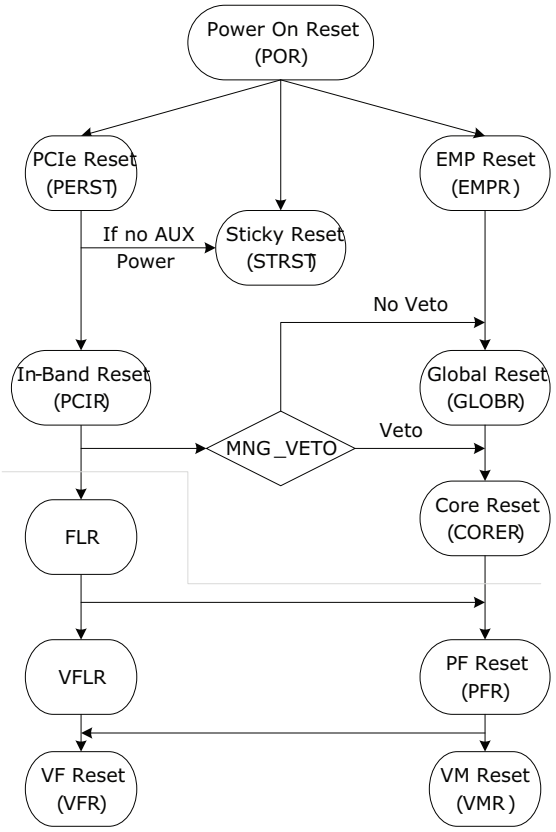
4.0 Initialization

4.1 Reset operation

The following sections lists the hardware and software reset sources that initialize the entire portions of the XL710 and functions 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 XL710 while the complete list of initialized logic is listed in Table 4-1. 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)	Internal power on detection circuitry LAN_PWR_GOOD input signal enabled by POR_BYPASS input
Sticky Reset (STRST)	Internal reset that clears PCIe sticky registers following PCIe reset
PCIe Reset (PERST)	PE_RST_N input signal
In-Band Reset (PCIR)	In-Band PCIe message signalling
FLR / VFLR	OS sets the Initiate FLR flag in the PF or VF PCIe configuration registers
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
PF Reset (PFR)	FLR assertion via the PCIe configuration space D3 to D0 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 register
PF Level reset (FLR)	Operating sytem sets the FLR bit in the PF PCIe config
VF Level reset (VFLR)	Operating system sets the FLR bit in the VF PCIe config or clearing the VFE bit in the parent's PF PCIe config

Figure 4-1. Hierarchical reset tree

Power On Reset (POR) — The XL710 has an internal POR signal that transit from low to high when the device power sources are above the normal operation voltage and the internal clocks are stable. As long as the internal POR signal is at the low level, the entire device is held at the reset state. On a transition of the POR to high level, the XL710 starts an internal initialization sequence that impacts the entire device. The device also has a LAN_PWR_GOOD input signal that might be used instead of the internal POR depending on the POR_BYPASS input signal setting. If the POR_BYPASS is set to 0b the internal POR is used. Otherwise, the external LAN_PWR_GOOD signal is used.

PCIe Reset (PERST) — The PCIe reset signal is kept at the low level when the system is at power down state and at system boot. Transit of PCIe reset to low generates an internal reset signals. If there is no AUX power, the PERST generates an internal STRST signal that clears PCIe sticky bits as listed in Table 4-1. PERST also triggers internally a PCIR, which is detailed later in this section.

Sticky Reset (STRST) — Sticky reset is internal signal triggered by PCIe reset when the XL710 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) — The 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.

Function Level Reset (FLR) — The XL710 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.

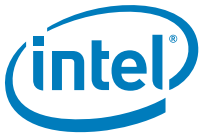
PF Reset (PFR) — PFR initializes the resources and data path of the PF and its VFs with no impact on other PFs, VFs and the EMP subsystem. Any further master cycles of the PF are not initiated while some packets that were already fetched completely might still be sent out. The PFR is generated by one of the following four causes: (1) D0 to D3hot transition, which is also known as ACPI reset; (2) FLR; (3) PF software sets the PFSWR bit in the PFGEN_CTRL register; (4) De-assertion of the *Bus Master Enable* flag in the PCI configuration space.

VF Level Reset (VFLR) — The XL710 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 and 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: (1) VFLR or clearing the *VFE* bit of the parent PF; (2) PF software sets the *VFSWR* bit in the VPGEN_VFRTRIG register of its VF.

VM Reset (VMR) — There are 384 VSIs where up to 256 of them can be VMDq2 VSIs (see Section 7.4.5.2.1.1). Such VSIs are associated with VMs. VMR is a mechanism to reset a VMDq2 VSI. VMR initializes the resources and data path of the VM with no impact on other PFs, VFs, VMs and 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.

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 to with NACK during the initialization cycle. This reset is not expected to be used other than as an escape mechanism in case



the XL710 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.

Global Reset (GLOBR) — GLOBR is a super-set 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 F*orce TCO Reset Disable* bit in the NVM).

EMP Reset (EMPR) — EMPR initializes the resources and data path connected to the EMP including its firmware reload. 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.

Table 4-1. Device logic affected by the reset sources

Reset Activation	POR	STRST / PCIR / PERST	EMPR	GLOBR	CORER	PFR/ FLR	VMR	VFR/ VFLR
Load the NVM to the shadow RAM in the hardware and clear the alternate structure	+							
Load the EMP firmware from the NVM	+		+					
Load device settings from NVM shadow and alternate structure (see details listed in Table — 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 the Programming Interface section 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 the Programming Interface section for the reset source of each register.	+	+	+	+	+	PF and its VFs		VF only
Cache contexts (filters, queue context, FCoE 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	+	+	+	+	+			

Table 4-1. Device logic affected by the reset sources

Reset Activation	POR	STRST / PCIR / PERST	EMPR	GLOBR	CORER	PFR/ FLR	VMR	VFR/ VFLR
Tx and Rx queue disable	+	+	+	+	+	PF and its VFs	by the PF SW	VFs
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
RSS key and table	+	+	+	+	+	PF		VF
Invalidate FD filters	+	+	+	+	+	PF		

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 and the WoL MAC addresses are loaded only following the de-assertion of the PCIe reset.The MAC and PHY are cleared only if the MNG_VETO flags in all four PRTPM_GC registers are cleared. If neither WUC filter is active and no port is needed by FW/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 12.2.1).
3. For exception list of registers that are not cleared on PFR see Section 12.2.1.
4. The XL710 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 XL710 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_SRLD 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	+	+	+				
RO PCIe LCB	+	+					
CORER Registers Auto-load	+	+	+	+	+	+	
GLOBR Registers Auto-load	+	[1]	1	+	+		
PHY Analog Configuration	+						
EMPR Registers Auto-load	+			+			

1. GLOBR registers auto-load only if the MNG_VETO flags in all four PRTPM_GC registers are cleared.

4.1.2 Reset Flows

This section describes the reset flows in the XL710 and software interaction.

4.1.2.1 POR Flow

For POR flow see Section 4.2.

4.1.2.2 PCI Reset and InBand PCI Reset Flow

The internal reset flow is shown in Figure 4-2 and described by the text that follows.

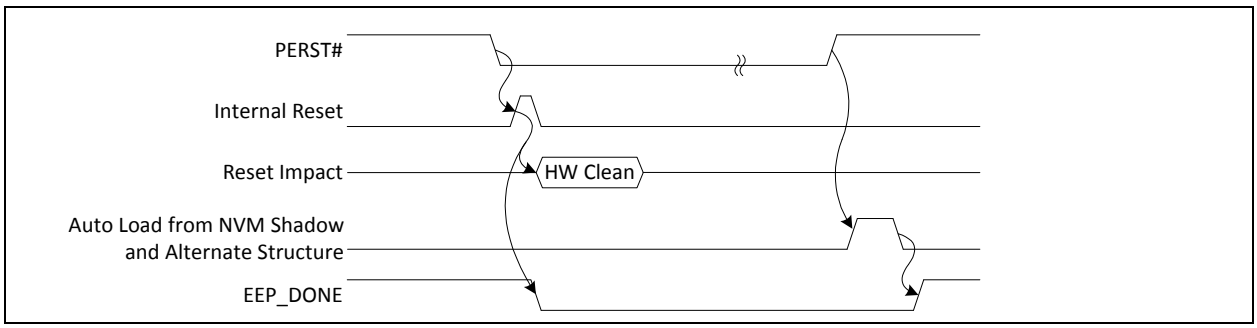


Figure 4-3. PCI reset flow

- Avoid any further master accesses on the PCIe bus and discard any completions for the PF.
- Initialize PCIe registers and core registers as listed in Table 4-1.
- Invalidate the LAN queues mapping tables of the PF and its VFs: VFQTABLE's and VSIQTABLE's.
- Invalidate the FPM tables of the PF and its VFs (function private memory).
- Disable all transmit, receive and admin queues. All packets in the XL710 are lost. EMP packets might be lost as well for a very short period (in the range of 1 µs).
- Invalidate all filters in the FD table and internal caches.
- Invalidate all internal caches: transmit and receive queue contexts; FCoE DDP contexts.
- Invalidate all associated VSI contexts.
- Clear interrupt settings of all functions.
- The following flags are cleared by the XL710 at the beginning of the reset flow:
  - HW_*_DONE bits in GLNVM_SRLD register (those ones that are listed in Table 4-2 in the respective PCIR or PERST columns)
  - The EMP_*_DONE bits in GLNVM_EMPLD register are cleared (those ones that are listed in Table 4-2 in the respective PCIR or PERST columns) if the respective GLNVM_EMPRQ.EMP_*_REQD bit is set.
  - The CONF_*_DONE bits in GLNVM_ULD register are cleared if the respective HW_*_DONE bit or EMP_*_DONE bit are 0b.
- The XL710 repeats the following flow for all previous *_*_DONE flags:
  - If the NVM is not valid:
    - Set the HW_*_DONE flag in GLNVM_SRLD register.
  - If the NVM is valid:
    - Load the matched block from the shadow memory and set the HW_*_DONE flag
    - If EMP_*_REQD is set:

Load any parameters from the alternate structure into the matched block(s)

Perform any configuration of the matched block(s)

Set the EMP_*_DONE flag

— For CORER and GLOBR modules, once HW_*_DONE and EMP_*_DONE flags are set, the matched CONF_*_DONE flag is set as well. For other modules, once HW_*_DONE flag is set, the matched CONF_*_DONE flag is set as well.

— At this point, the EMP might start responding the Get Version Admin command, which is blocked until this stage.

- When checking for hardware configuration to complete, software should either wait for a response to a Get Version Admin command or poll on the CONF_*_DONE bits.

Following these steps, hardware is ready to accept operating system configuration cycles.

4.1.2.3 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 4-4. 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.

Before initiating the PFR, software is expected to disable all transmit and receive queues of the PF as described in the following pseudo code:

disable_pf_queues()// disable all Tx and Rx queues of the PF

```
{

1.Disconnect all Tx and Rx queues from the interrupt link lists as follows

  Set the FIRSTQ_INDX field to 0x7FF in all PFINT_LNKLSTx register of the PF

  Set the FIRSTQ_INDX field to 0x7FF in all VPINT_LNKLSTx register of the VFs of the PF

2.last_q = PFLAN_QALLOC.LASTQ - PFLAN_QALLOC.FIRSTQ // “last_q” is a local variable used in the
  steps below. It is the index of the last queue of the PF and its VFs (in the PF space)

3.Set the SET_QDIS flag in the GLLAN_TXPRE_QDIS registers for all transmit queues of the PF and its
  VFs. The queue indexes are global ones starting by PFLAN_QALLOC.FIRSTQ and up to
  PFLAN_QALLOC.LASTQ.

4.Clear QRX_ENA[register index=0,… last_q] // Disable all Rx queues of the PF

5.Clear QTX_ENA[register index=0,… last_q] // Disable all Tx queues of the PF. Software must
  guarantee a gap of at least 400usec from step 3 to this step

6.Wait for the last Rx queue to be disabled and wait for all Tx queues to be disabled or time
  expires (expiration time is defined by the ENDLESS_XOFF_THRESH parameter). Note that if the
  software is not required to take any special actions in case the time is expires and can continue
  to the next step initiating the PFR.

}
```

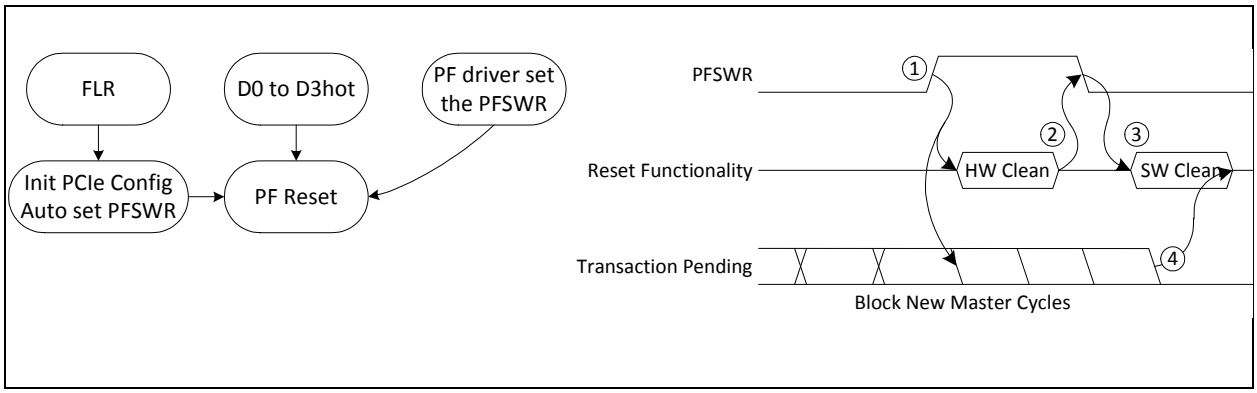


Figure 4-4. PFR flow

The PF hardware response to PFR is listed in Table 4-1. Note to the following specific events:

- Emulate internal VFR to all its VFs (the hardware response to VFR is described in Section 4.1.2.5.3 with the exception that the VF's CSRs are not gated on read).
- Avoid any further master accesses on the PCIe bus and discard any completions for the PF and its VFs.
  - 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.
  - 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).
- Disable all transmit receive and admin queues of the PF and its VFs (note that some transmit packets that are already fetched completely might be transmitted).
  - Disabling the transmit queues might take some time until the XL710 processes all transmit descriptors that are already fetched. Because many queues are active, this process might take longer. The transmit completion might be further delayed when the TC is paused by flow control. It can happen if other functions have active transmit queues on the same TC as the function under reset. Hardware includes a timeout mechanism that prevents indefinite latency as described in Section 7.7.1.2.8.
  - Disabling the receive queues might take some time as well until the packets of the function's queues are flushed from the shared receive data path. The completion might be further delayed when there are packets of other functions in the pipe that belongs to no-drop TC with no valid receive descriptors. Hardware includes a timeout mechanism that prevents indefinite latency as described in Section 7.7.1.2.8.
- Clear interrupt settings of the PF and its VFs (excluding the CEQ and LAN transmit and receive cause control registers and any interrupt context in the PCIe configuration space). Write 1b to also clear the VFLRE flags in the *GLGEN_VFLRSTAT* registers of all the VFs owned by the PF.

After all the previous steps are completed, hardware does the following:

- Clears the *PFSWR* bit in the *PFGEN_CTRL* register.
- Releases bus master cycles for the PF and its VFs.

The PF software polls the *PFSWR* bit until it is cleared (indicating that hardware completed its reset flow). On top of it, 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.

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). Once the Transactions Pending is found cleared, the PF software does the following:

- Clears the *VFSWR* flag in the *VPGEN_VFRTRIG* registers of its VFs.
- Sets the VFR_STATE in the *VFGEN_RSTAT* registers of its VFs to VFR completed. At this point, the memory structures of the VF can be released and the VF software can start its initialization flow.

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):

- Query the reset counters (*CORERCNT, GLOBRCNT* and *EMPRCNT*) in the GLGEN_RSTAT register
- ...
- Interrupt enable flow
- 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" then go to the global reset flow
  - If Device state = "Device active" then
  - 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, then go to the global reset flow

4.1.2.4 FLR flow

FLR resets a specific PF. In the case of SR-IOV, it also resets the VFs of this PF.

- The following steps are optional:
  - The operating system is expected to clear the BME bit in the Command register in the PCI configuration register of the PF.
  - 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.
  - As a response, hardware avoids any new master cycles of the PF and its VFs (including MSI-X initiation).
  - The operating system should poll the Transaction Pending bit in the Device Status register of the PF until it is cleared.
  - 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 that all the previous steps are expected and recommended but not enforced by the PCI specification.
- 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.
- The PF hardware response as follows:
  - Initialize the PCIe configuration space of the PF including clearing the BME bit of the PF and the VFE bit.

- By clearing the BME, hardware avoids any further master accesses on the PCIe bus and trash any completions for the PF.
- By clearing the VFE bit, all VFs of the PF avoid any further master accesses on the PCIe bus and 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 *Transaction Pending* bits in the PCIe configuration space are cleared.

— 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).

4.1.2.5 VFR/VFLR flows

The VF reset flow is shown in Figure 4-5 and detailed in the sections that follow.

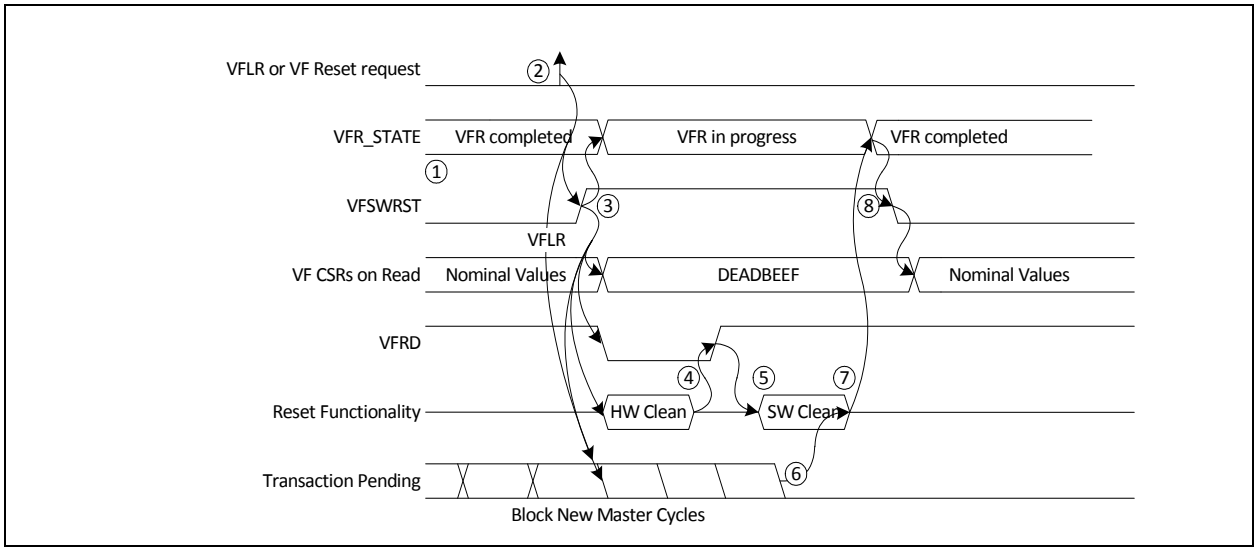


Figure 4-5. VF reset (VFR) flow

4.1.2.5.1 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 4-5. See also Section 4.1.2.5.3 that 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 that the VF software gets a control over its VF only after any prior VF reset flow is completed (step 1 in Figure 4-5).

- 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 mailbox (Admin command) or any other software based sideband channel (step 2 in Figure 4-5).
- After that, the VF polls the VFR_STATE in the VFGEN_RSTAT register until it is set by the PF to VFR completed (step 8 in Figure 4-5 and explained in Section 4.1.2.5.3).
- The VF software proceeds activating the function.

4.1.2.5.2 VF reset request by the operating system (VFLR)

The VF reset initiation by the operating system is described as follows and shown in Figure 4-5.

- The following steps are optional:
  - Clear the BME bit in the VF Command register.
  - As a response, hardware avoids any new master cycles of the VF (including MSI-X initiation). Old completions are trashed by hardware.
  - The operating system polls the *Transaction Pending* bit in the VF Device Status r4egister until it is cleared.
- The operating system sets the *FLR* bit in the VF Device Control register (step 2 in Figure 4-5).
- The operating system is required by PCIe specification to wait 100 ms before it can assume that the VFLR sequence is completed by hardware.
- 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).
- The VF software proceeds activating the function.

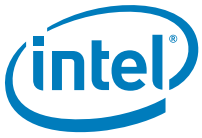
Hardware responses to VFLR are as follows:

- Initialize the PCIe configuration space of the VF including the *Bus Master Enable* flag. The *Transaction 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).
- Set internally the VFSWR bit in the VPGEN_VFRTRIG register of the VF, which initiates a VFR (described in Section 4.1.2.5.3).
- Set the matched VFLRE flag in the GLGEN_VFLRSTAT registers and initiate an interrupt to the parent PF. The PF response to the VFLR interrupt is described in Section 4.1.2.5.3.

4.1.2.5.3 VF reset flow by the PF software driver

The PF software is called to initiate the VFR by the VF software driver or as a result of a VFLR interrupt. See Figure 4-5.

- Specific step for a VFLR interrupt:
  - The PF software queries the VFLRE flags in the GLGEN_VFLRSTAT registers (of the VFs that it owns). It then writes 1b to clear, only to the bit(s) of the matched VF(s) that generated the VFLR and owned by the PF. Note that the GLGEN_VFLRSTAT registers are composed of 128 bits for the 128 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.
- Specific step for a VF reset request by the VF driver:
  - Set the *VFSWR* bit in the VPGEN_VFRTRIG register of the VF (step 3 in Figure 4-5).



- Poll the VFRD flag in the VPGEN_VFRSTAT register of the VF until it is set (indicating that hardware completed the VFR flow) (step 4 in Figure 4-5).

The hardware response to setting the *VFSWR* bit is listed in Table 4-1. Note that as part of the VF registers, the VFGEN_RSTAT register is cleared as well reflecting VFR in progress state. On top of it, hardware also gates read accesses to the CSRs of the VF returning DEADBEEF or DEADBEAF values and also gates any master accesses.

- When the reset flow is completed, the hardware sets the VFRD flag in the VPGEN_VFRSTAT register.

Once the VFRD flag in the VPGEN_VFRSTAT register is found active, the PF software proceeds with the VF reset flow (step 5 in Figure 4-5).

- Disable the receive queues of the VF following the fast queue disable flow per each queue as described in Section 8.3.3.1.3. Note that the hardware auto-disable the transmit queues of the VF.
- Clear the interrupt settings of the VF.
- Clear the queue mapping tables of the VF's VSIs (VSIQTABLEs).
- Clear the filters in the FD table that were assigned by the PF for the VF.
- Remove all the MAC/VLAN filters from the VSIs of the VF and then remove these VSIs.
- 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 6 in Figure 4-5).
- 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):
  - Add the VSIs for the VF (including its Transmit and Receive queues and its Scheduler).
  - Add the MAC/VLAN filters for the VSI.
  - Enable the VFLAN_QTABLE by setting the VPLAN_MAPENA and then program the VFLAN_QTABLE to the queues of the VF.
- The PF software completes the flow by notifying the VF that the reset flow is completed. It sets the VFR_STATE in the VFGEN_RSTAT register to VFR completed and clears the *VFSWR* bit in the VPGEN_VFRTRIG register (step 7 in Figure 4-5).

Hardware response to cleared *VFSWR* flag:

- 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 4-5).
- Master cycles are not blocked anymore by the reset logic.
- The hardware is ready to be used by the VF.

4.1.2.6 VMR flow

Before initiating a VM reset the PF software should disable the interrupts associated with this VM. The VM interrupt causes might share the same interrupt with the PF. In this case the PF software need only to remove the VM interrupts causes from those interrupt linked list as described in Section 7.5.3.1.3. Then the PF software can initiate the VM reset and re-enable the interrupts of the PF.

The PF sets the *VMSWR* bit in the VSIGEN_RTRIG register:

- Hardware response to VMR is the same as its response to VFR (other than invalidating the VFQTABLE's and VFxxx registers, which the VM does not own).

The PF polls the VMR done indication in the VSIGEN_RSTAT register and then executes the following complementary steps (as the VFR flow):

- Disable the transmit and receive queues using the fast queue disable flow.
- Optionally clear the queue mapping tables, FD filters and VSI context.
- The PF 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.
- The PF completes the flow by clearing the *VMSWR* bit in the VSIGEN_RTRIG register.

4.1.2.7 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 XL710 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 4-6 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.

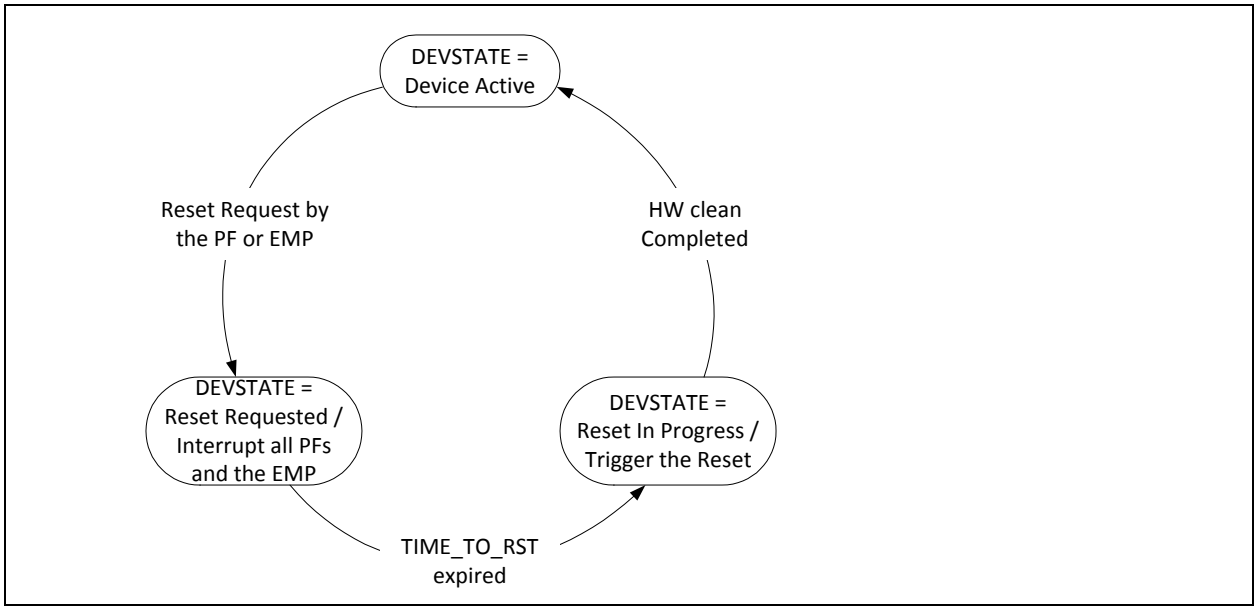


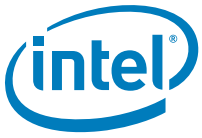
Figure 4-6. Global resets flow

4.1.2.7.1 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; reset in progress.



RESET_TYPE The RESET_TYPE reflects the last reset cause initiated to the XL710. 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 and EMPR.

RST_CNT The GLGEN_RSTAT reflects the following counters: CORERCNT, GLOBRCNT and EMPRCNT. These fields are 2-bit counters each. They count the matched reset completion events since POR.

GRSTDEL GRSTDEL in the GLGEN_RSTCTL register is the delay from reset request to its initiation by hardware. The GRSTDEL is loaded from the NVM and defined in 100 ms units up to ~6.5 seconds.

TIME_TO_RST The TIME_TO_RST is a down counter, loaded by the GLGEN_RSTCTL.GRSTDEL following a reset request. When the TIME_TO_RST reaches a zero value hardware initiates the requested reset.

4.1.2.7.2 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 XL710 state in the GLGEN_RSTAT register.

Hardware response:

- Changes the GLGEN_RSTAT register are as follows:
  - Set DEVSTATE to reset requested.
  - Set RESET_TYPE to CORER indicating the requested reset.
  - Set TIME_TO_RST by the GLGEN_RSTCTL.GRSTDEL value and start counting down.
- Further write accesses to the GLGEN_RTRIG register do not re-trigger the TIME_TO_RST. Still, if a stronger reset is set, it overrides this reset and is reflected in the RESET_TYPE field.
- Initiates a GRST interrupt to all PFs and EMP that the reset is about to be fired by hardware.
- Once the GLGEN_RSTAT.TIME_TO_RST gets to zero, hardware triggers the internal core reset and changes the GLGEN_RSTAT.DEVSTATE to reset in progress.
- The following flags are cleared by the XL710 at the beginning of the reset flow:
  - HW_*_DONE bits in GLNVM_SRLD register (those ones that are listed in Table 4-2 in the respective PCIR or PERST columns) are cleared.
  - The EMP_*_DONE bits in GLNVM_EMPLD register are cleared (those ones that are listed in Table 4-2 in the respective PCIR or PERST columns) if GLNVM_EMPRQ.EMP_*_REQD is set.
  - The CONF_*_DONE bits in GLNVM_ULD are cleared if the respective HW_*_DONE bit or EMP_*_DONE bit are set to 0b.
- Clear the entire transmit and receive data path, 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).
- Reset internal device logic excluding EMP cluster, PCI i/f and MAC/PHY cluster as listed in Table 4-1.
- Once the reset flow is completed, update the GLGEN_RSTAT register as follows:
  - Set DEVSTATE to device active.
  - Increment the CORERCNT by one.
- The XL710 repeats the following flow for all previous *_*_DONE flags:
  - If the NVM is not valid:
    - Set the HW_*_DONE flag in GLNVM_SRLD register.
  - If the NVM is valid:
    - Load the matched block from the shadow memory and set the HW_*_DONE flag

- If EMP_*_REQD is set:

  Load any parameters from the Alternate Structure into the matched block(s)

  Perform any configuration of the matched block(s)

  Set the EMP_*_DONE flag

— For CORER and GLOBR modules, once HW_*_DONE and EMP_*_DONE flags are set, the matched CONF_*_DONE flag is set as well. For other modules, once HW_*_DONE flag is set, the matched CONF_*_DONE flag is set as well.

— At this point, the EMP might start responding the Get Version Admin command, which is blocked until this stage.

Following the GRST interrupt, all PFs and the EMP poll the XL710 state in the GLGEN_RSTAT register.

- Keep polling the register as long as DEVSTATE is not equals to device active. Note that as long as DEVSTATE equals to reset requested, the TIME_TO_RST indicates the remaining delay to the internal reset triggering by hardware. Note the PF drivers should avoid any CSR slave accesses other than the one required to query the XL710 state.
- The GLGEN_RSTAT 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 would be good practice to avoid too frequent global resets to avoid such cases.
- Check the RESET_TYPE that indicates the reset that was initiated and follow the required initialization flow.
  — Note that also the function that initiated the reset should check the RESET_TYPE since it might reflect a stronger reset initiated by another function.
- 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.
  — When checking for the hardware configuration to be done, software should either wait for a response to a Get Version Admin command or poll on the CONF_*_DONE bits.
- The PFs proceeds with their software initialization flow.

4.1.2.7.3 GLOBR flow

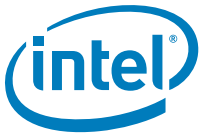
Global reset is initiated by the PFs or the EMP by setting the GLOBR flag in the GLGEN_RTRIG. The GLOBR flow is identical to the CORER flow with the following changes:

- 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 XL710 setting from the NVM is listed by the GLOBR column in Table 4-2.

4.1.2.7.4 EMPR flow

EMP reset is expected to be used by the EMP as a mechanism to resolve potential hardware locks or potential lost 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.

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 XL710 setting from the NVM is listed by the EMPR column in Table 4-2.

4.2 Cold reset initialization

This section describes the flow of the XL710 power up following a cold reset (application of power to the XL710). The initialization sequence for the XL710 is broken down into phases: power on, BIOS initialization, and device driver load.

- Power on (Section 4.2.1) 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 4.2.2) begins when option ROM code is loaded by BIOS in order to provide boot services for PXE, iSCSI, or FCoE. Other option ROM capabilities include configuration images for certain OEM environments such as SMASH/CLP.
- The final phase of the XL710 initialization is device driver Load, where the operating system loads the various device drivers, and the XL710 has been initialized for its post-boot operation. See Section 4.2.3.

4.2.1 Power on

Figure 4-7 and Figure 4-8 describe the stages that make up the power up sequence. Note that two procedures take place in parallel following initialization of internal clocks and loading the hardware-managed units from the NVM:

- The on-die processors are been initialized, starting with the EMP
- Once PERST# is de-asserted, the PCIe link goes through its initialization flow

Figure 4-7 shows Case I, where APM or pass-through manageability are enabled at pre-boot. In that case, the Ethernet link is brought up once the EMP has initialized, independent of the de-assertion of PERST#.

Figure 4-8 shows Case II, where APM and pass-through manageability are both disabled pre-boot. In that case, the Ethernet link kept down as long as PESRT# is kept asserted. Once PERST# is de-asserted, the Ethernet link is brought up (but not before EMP completed initialization).

Note that in both cases, the Ethernet link is brought up only after the EMP completes its initialization.

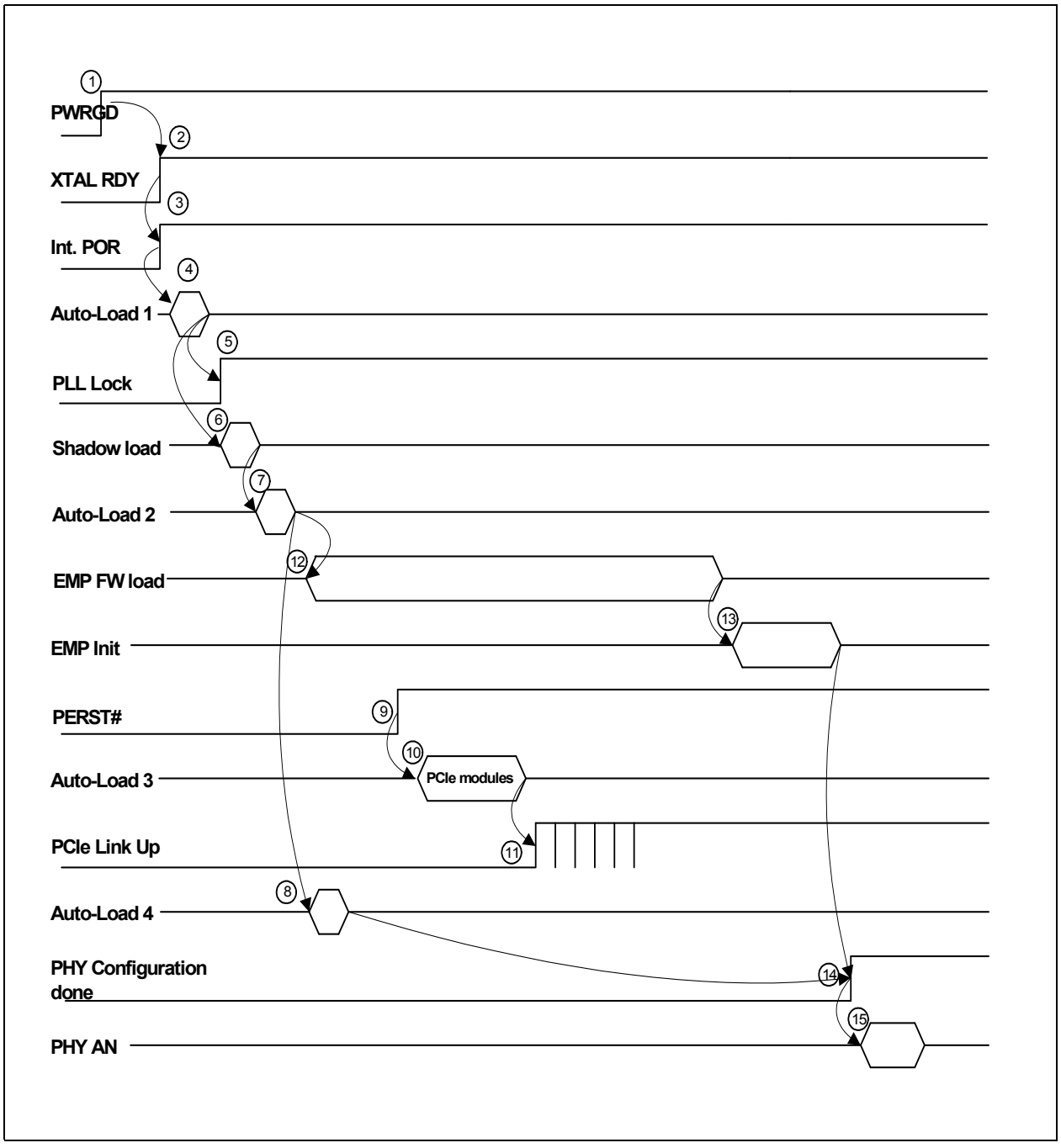


Figure 4-7. Power-on sequence - case I

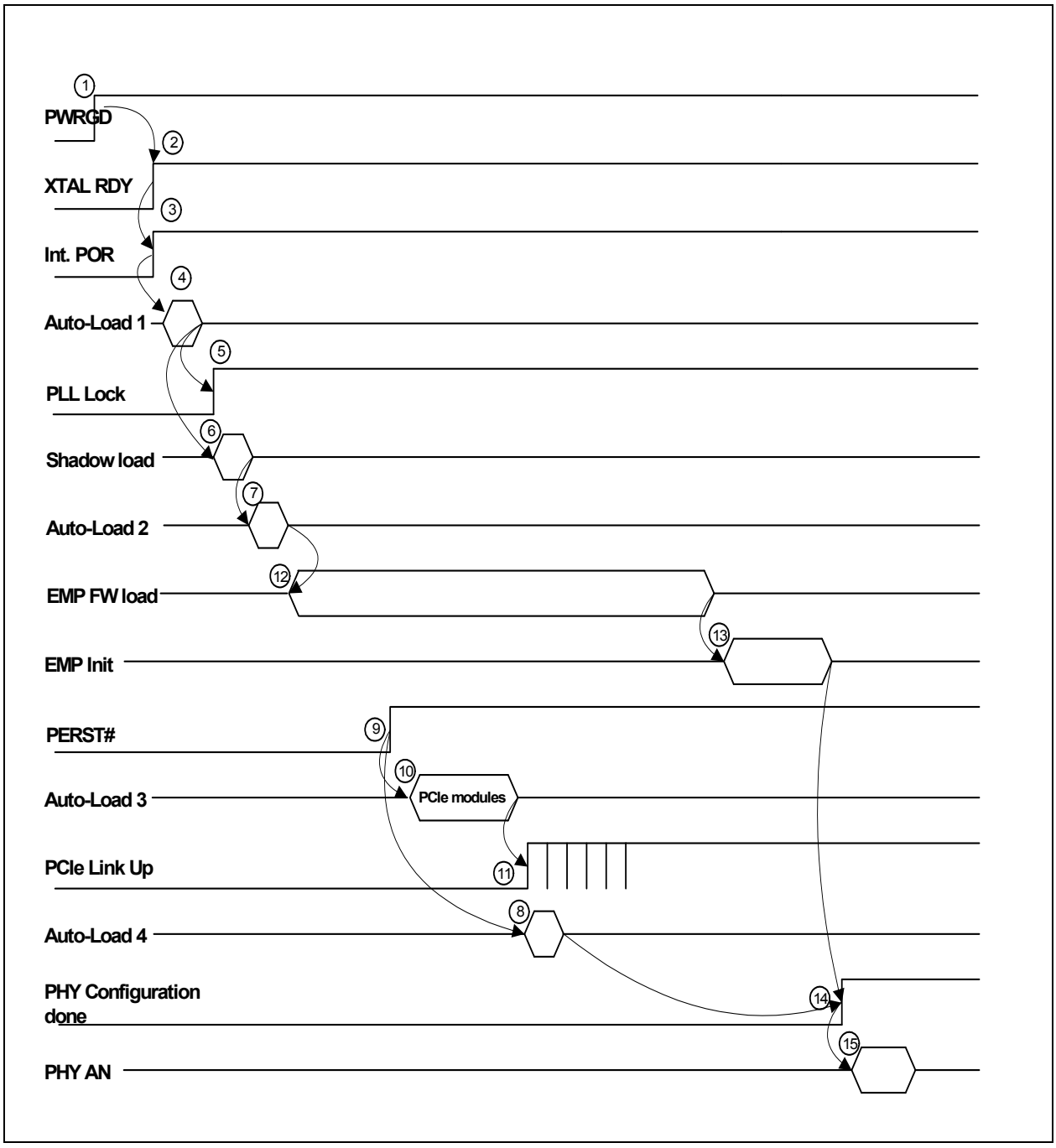
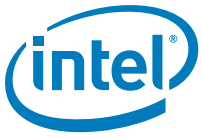


Figure 4-8. Power-on sequence - case II

Table 4-3 lists the relationship between steps and the timing of each stage.

Legend:

- Triggering event - this is the event that starts the corresponding step.



- Duration - Time it takes to complete the respective step (usually defines the maximum duration).
- Completion Time from Start - The time the respective step ends, counting from beginning of the power-up sequence. It usually defines the latest time this step might end.

Example: EMP firmware load

- Step - EMP firmware load
- Triggering event - Auto-load 2 complete (an earlier step)
- Duration - it might take up to 280 ms to complete this step
- Completion Time from Start - Auto-load 2 ends at time 79 ms. With this step taking up to 280 ms, completion would be maximum of 359 ms from start

Table 4-3 Power On Sequence stages

	Step	Triggering Event	Duration (ms)	Completion Time From Start (ms)
1	Power is applied to the XL710.	-	0	0
2	Xtal clock ready.	Power ramp	35	35
3	Internal POR goes active following clock.	Clock stable	0	35
4	Auto-load 1 completes - NVM validity is checked and NVM contents are loaded into the analog sections (PCIe and internal PHY).	Internal POR	21	56
5	The internal PLL is up.	Auto-load one complete	2	58
6	NVM contents are loaded into shadow RAM.	Internal PLL locked	21 [1]	79
7	Auto-load two completes - POR and EMP modules are loaded from shadow RAM.	Shadow RAM ready	~0	79
8	Auto-load four completes - units on core and MAC/PHY clock are loaded from shadow RAM. Case I - if APME or manageability is enabled, it is done following auto-load two. Case II - if both APME and manageability are disabled, it is done following PERST# de-assertion. **Note:** the order between auto-load three and four is not deterministic.	Auto-load two complete/PCIe PLL is up	~0 **(TBD)**	79/110
9	PERST# is de-asserted.	Power ramp	0	≥100
10	The PCIe PLL is up. The PCIe unit is loaded from NVM shadow RAM (auto-load three) and PCIe enters a detect state.	PERST# de-assert	10 **(TBD)**	110
11	PCIe specification requirement - PCIe link must enter a detect state.	PERST# de-assert	20	120
	PCIe specification requirement - PCIe link is ready to process configuration cycles.	PERST# de-assert	100	200
12	EMP firmware load - both manageability and other EMP code.	Auto-load two complete	280	359
13	EMP firmware initialized.	EMP firmware load completed	20	379
14	MAC/PHY configuration by EMP is done (either internal PHY or external PHY) and can bring up link (GLNVM_ULD.CONF_GLOBAL_DONE is set).	EMP firmware initialized	~0	379
15	Ethernet link established	MAC/PHY completes configuration	See Note [2]	See Note 2

1.
2. Specific to the PHY media used.

4.2.1.1 LAN_PWR_GOOD support

It is possible to begin device initialization based on the assertion of the LAN_PWR_GOOD input rather than based on main power ramp. When the POR_BYPASS input pin is set to 1b, the XL710 disables the internal POR circuit and uses the LAN_PWR_GOOD pin as a POR indication. Note that LAN_PWR_GOOD should be asserted during pre-boot time (before the operating system boots).

Table 4.2.1.2 and Figure depict the initialization flow. Other events listed in Table 4-4 are shifted relative to Internal POR and PERST#.

Table 4-4. Power-on sequence with a LAN_PWR_GOOD signal

	Step	Triggering Event	Duration (ms)	Completion Time From Start (ms)
1	Power Ramp - Must meet the sequence defined in Section 14.3.1.1. Time = 0 ms is when all power rails are at 90% of nominal voltage.	Power ramp	0	0
2	LAN_PWR_GOOD - must not be asserted before specified duration.	Power ramp	40	40
3	Xtal clock ready.	LAN_PWR_GOOD	1	41
4	Internal POR goes active following Xtal ready.	Xtal clock ready	0	41
5	PERST# must not be de-asserted before specified duration. Note that the PCIe specification still holds that PERST# is de-asserted at least 100 ms following power ramp.	LAN_PWR_GOOD	50	91

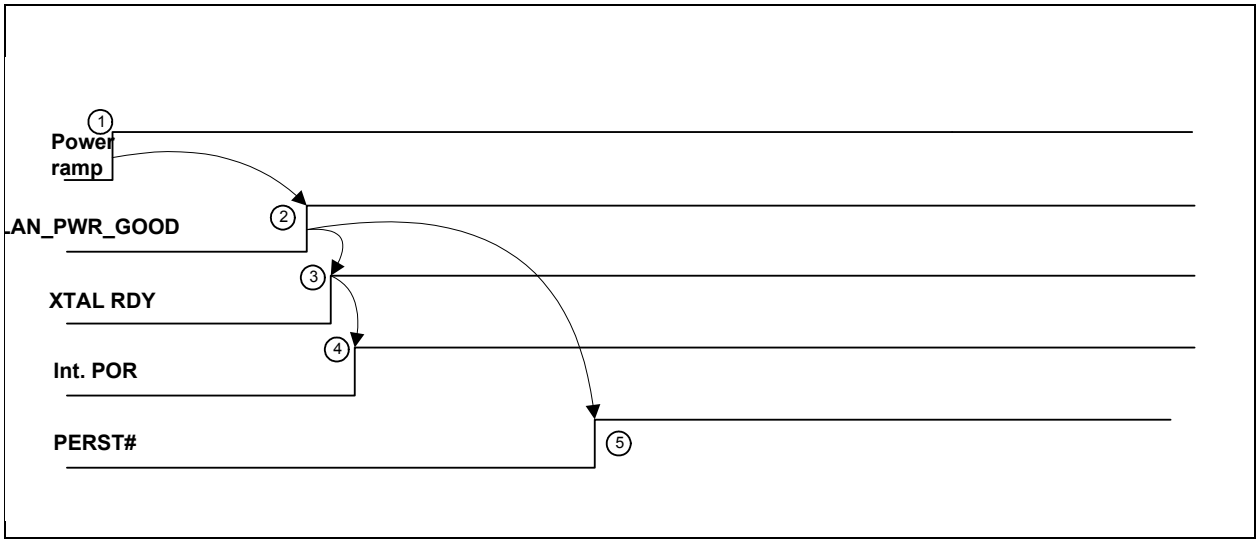
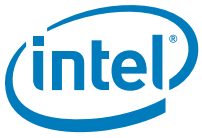


Figure 4-9. Power-on sequence with a LAN_PWR_GOOD signal

4.2.1.2 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 3.3.4.2 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, the GLNVM_GENS.FL_AUTO_RD bit is set. The bit is also set when the NVM is found blank.

4.2.1.3 Auto-load into device units

- This stage loads the NVM configuration into the device units, either directly from the NVM or indirectly through shadow RAM.

Loading the following NVM modules is tracked and reported as:

- PCIe analog
- PHY analog
- PCIR registers auto load
- RO PCIe LCB
- PCIe Transaction Layer (TL) shared
- PCIe ALT auto load
- CORER registers auto load
- GLOBR registers auto load
- POR registers auto load
- EMPR registers auto load

The following CSR fields track the progress of loading the NVM modules following power on and the various resets:

- GLNVM_GENS.FL_AUTO_RD, when set, indicates that the shadow RAM was loaded from the NVM and is ready for use.
- GLNVM_SRLD.HW_*_DONE bits (one per relevant module previously identified), when cleared, indicate that the respective module needs to be loaded from shadow RAM into the XL710. When set, it means that the module is not required to load or has already been loaded.
- EMP_*_REQD bits in the GLNVM_EMPRQ register (one bit per relevant module) indicate whether the EMP involvement is required during the initialization of the module.
- EMP_*_DONE bits in the GLNVM_EMPLD register (one bit per relevant NVM module) indicate when the corresponding module completed initialization by the EMP.
- GLNVM_ULD.CONF_*_DONE bits (one per module previously identified), when set, indicate that the respective units are initialized and ready for use.

Default hardware values:

- GLNVM_GENS.FL_AUTO_RD = 0b
- GLNVM_SRLD.HW_*_DONE = 0b
- GLNVM_EMPRQ.EMP_*_REQD = 0b
- GLNVM_EMPLD.EMP_*_DONE = 0b
- GLNVM_ULD.CONF_*_DONE = 0b

The flow during a power-on stage is as follows:

- If the NVM is found blank:
  - All GLNVM_SRLD.HW_*_DONE bits and all GLNVM_ULD.CONF_*_DONE bits are set by the XL710
- If the NVM is found valid:
  - GLNVM_GENS.FL_AUTO_RD is set when the shadow RAM was loaded from the NVM
  - GLNVM_EMPRQ.EMP_*_REQD is loaded from the NVM
  - When a module completes loading from the shadow RAM to the XL710, it sets the respective GLNVM_SRLD.HW_*_DONE bit. If the respective GLNVM_EMPRQ.EMP_*_REQD bit is 0b, then the respective GLNVM_ULD.CONF_*_DONE is set to 1b by hardware.
  - After EMP firmware loads, during its init sequence, it sets the GLNVM_EMPLD.EMP_*_DONE bit (for each of CORER and GLOBLR domains). As a result, hardware sets the GLNVM_ULD.CONF_*_DONE bit to 1b.

4.2.1.4 Firmware initialization

Once EMP firmware loads, the following XL710 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 XL710 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 XL710 for device driver initialization (see Section 7.10.3).
- HMC default profile has been configured

The following NVM fields are loaded by EMP firmware to control its operation:

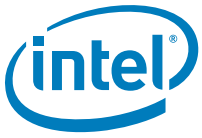
- The *OEM capabilities* and *OEM technologies enabled* words in the *EMP Shadow-RAM Module Header* section determines OEM-specific technologies
- The *Feature Enable.EVB Enable* field in the *EMP Module Header* section determines if EVB is enabled

Once EMP firmware is up and running, the XL710 can be configured via its management interfaces, and certain device capabilities are then enabled or disabled (such as soft skewing).

4.2.1.5 MAC address initialization

Two sets of station LAN MAC addresses are supported:

- A station MAC address per physical function. These addresses are used by the internal switch for L2 filtering of Rx packets, by the RX classification filters, and for WoL purposes. The addresses are loaded from the NVM.
  - The station MAC addresses for WoL are loaded by the EMP into the *PRTPM_SAL* and *PRTPM_SAH* registers (see also Section 5.4.3)
    - Default - A single PF address is loaded per port into *PRTPM_SAL[0,Port]* and *PRTPM_SAH[0,Port]*



- A station MAC address per Ethernet port. These addresses are used for link-level functionality such as flow control frames.
  - The *PRTGL_SAL* and *PRTGL_SAH* registers contain these addresses for the four 10 Gb/s MACs.
  - The *PRTMAC_HSEC_CTL_TX_SA_PART1* and *PRTMAC_HSEC_CTL_TX_SA_PART2* registers contain these addresses for the 40 Gb/s MACs.

The following table lists how each address can be set or read and where it is stored:

MAC Address Type	Where Stored	How Reported	How Modified
LAN MAC address - factory	NVM PF allocations section		N/A
LAN MAC address – current	NVM PF allocations section; alternate RAM	Manage MAC address read (PF LAN SA)	Write alternate AQ command: Alternate LAN MAC address (LS) Alternate LAN MAC address (MS) or NC-SI Set Address command or manage MAC address write (update LAA only)
SAN MAC address - factory	NVM permanent SAN MAC address section		N/A
SAN MAC address – current	NVM permanent SAN MAC address section; alternate RAM	Manage MAC address read (PF SAN SA)	Write alternate AQ command: Alternate SAN MAC address (LS) Alternate SAN MAC address (MS) or NC-SI Set Address command
Port MAC address	GLOBR registers auto-load module (PRTGL_SAL/H)	Manage MAC address read (port SA)	Manage MAC address write AQ command (update port address) **Note:** Every function within a port is allowed to use this command – the last command takes precedence.
LLDP MAC address	Use the port MAC address for SA, chassis and port IDs.		
WoL MAC address	NVM PF allocations section, alternate RAM (PRTPM_SAL/H)	Manage MAC address read (PF WoL SA)	Manage MAC address write AQ command (update LAA and WoL address)

The following rules apply:

- The *PF_NUM* field in *PRTPM_SAH* is deducted from the PF issuing the command
- The *MC_MAG_EN* field in *PRTPM_SAH* is loaded from the NVM and is not affected by 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.

4.2.1.5.1 Manage MAC address read command

This command is used by the PF driver to read the per-PF station MAC address.

Indirect command

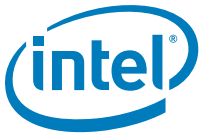
Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Table 7.10.5.2.1 for details.
Opcode	2-3	0x0107	Command opcode.
Datalen	4-5	0x00	Must be 0x0, value is ignored.
Return Value/ VFID	6-7	0x00	Return value. Zeroed by the device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Command Flags	16-17	Reserved	Zeroed by the device driver.
SAH	18-19	See remarks	Read operation: Reserved. Write operation: High 16 bits of the MAC address (big endian order). Example: if MAC address = 11:22:33:44:55:66 then SAH = 0x1122.
SAL	20-23	See remarks	Read operation: Reserved. Write operation: Low 32 bits of the MAC address (big endian order). Example: if MAC address = 11:22:33:44:55:66 then SAL = 0x33445566.
Data Address High	24-27	Buff Addr	High bits of return buffer address.
Data Address Low	28-31	Buff Addr	Low bits of return buffer address.

4.2.1.5.2 Manage MAC address read response

A firmware acknowledge to the Manage MAC Address Read command

Indirect Response

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Table 7.10.5.2.2 for details.
Opcode	2-3	0x0107	Command opcode.
Datalen	4-5	0x00	Must be 0x0, value is ignored.
Return Value/ VFID	6-7	Return Value	Return value. Firmware supplies in the *Return Value* field indication on the completion of the Manage LAA command. 0x0 - No error. Others - Error detected in the command.
Cookie High	8-11	Cookie	Opaque value is copied by firmware from the manage LAA command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware from the manage LAA command.



Name	Bytes.Bits	Value	Remarks
Command Flags	16-17.3	Reserved	Zeroed by the EMP.
	16.4	LAN Address Valid	
	16.5	SAN Address Valid	
	16.6	Port Address Valid	
	16.7	Wol Address Valid	
	17	Reserved	
Reserved	18-23	0x0	Reserved.
Data Address High	24-27	Buff Addr	High bits of return buffer address.
Data Address Low	28-31	Buff Addr	Low bits of return buffer address.

Note: All MAC addresses are in big endian order.

Address	Offset	Description
PF LAN SA	0-5	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.
PF SAN SA	6-11	Current device value of the PF SAN MAC address. Validated by the *SAN Address Valid* flag.
Port SA	12-17	Current device value of the Port MAC address. Validated by the *Port Address Valid* flag.
PF WoL SA	18-23	Current device value of the PF WoL MAC address. Validated by the *WoL Address Valid* flag.

4.2.1.5.3 Manage MAC addresses write command

This command is used by the PF driver to write the per-PF station MAC address.

Direct command.

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Table 7.10.5.1.1 for details.
Opcode	2-3	0x108	Command opcode. 0x0107 is used for reads; 0x0108 is used for writes.
Datalen	4-5	0x00	Must be 0x0, value is ignored.
Return Value/ VFID	6-7	0x00	Return value. Zeroed by the device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.

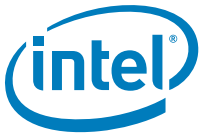
Name	Bytes.Bits	Value	Remarks
Command Flags	16-17.5	Reserved	Zeroed by driver
	17.7:6	Write Type	00b = Update LAA only. 01b = Update LAA and WOL address. 10b = Update port address. 11b = Reserved.
SAH	18-19	See Remarks	High 16 bits of the MAC address (big endian order). Example: if MAC address = 11:22:33:44:55:66 then SAH = 0x1122.
SAL	20-23	See remarks	Low 32 bits of the MAC address (big endian order). Example: if MAC address = 11:22:33:44:55:66 then SAL = 0x33445566.
Data Address High	24-27	Buff Addr	High bits of return buffer address.
Data Address Low	28-31	Buff Addr	Low bits of return buffer address.

4.2.1.5.4 Manage MAC address write response

A firmware acknowledge to the Manage LAA command.

Direct response.

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Table 7.10.5.1.2 for details.
Opcode	2-3	0x0108	Command opcode.
Datalen	4-5	0x00	Must be 0x0, value is ignored.
Return Value/ VFID	6-7	Return value	Return value. Firmware supplies in the *Return Value* field indication on the completion of the Manage LAA command. 0x0 - No error. Others - Error detected in the command.
Cookie High	8-11	Cookie	Opaque value is copied by firmware from the manage LAA command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware from the manage LAA command.
Command Flags	16-17.5	Reserved	Zeroed by the EMP.
	17.7:6	Write Type	00b = Update LAA only. 01b = Update LAA and WOL address. 10b = Update port address. 11b = Reserved.
Reserved	18-23	0x0	Reserved.
Data Address High	24-27	Buff Addr	High bits of return buffer address.
Data Address Low	28-31	Buff Addr	Low bits of return buffer address.



4.2.1.6 Power-on device state

This section describes the specific setting of each of the XL710'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, then enabled LAN ports are brought up by firmware once EMP firmware initialization completes. Else, 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 7.10.3 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 or for MFP functionality.
  - See Section 7.4.9.4.2 for more details.
- **DCB** - The NVM loads the LLDP and DCBx setting into the XL710. Once EMP firmware initializes and link is up, firmware engages in DCBx negotiation. Firmware then makes the appropriate changes in the XL710's configuration based on the outcome of the DCBx protocol. At this stage, DCB capabilities (TCs, ETS, PFC) might be enabled.
- **Tx Scheduler** - As the switch VSIs are enabled, firmware allocates Tx scheduler queue sets per each PF VSI based on the NVM configuration. Each queue set is configured to default behavior. Firmware also generates the required handles to enable system software to update the configuration of each queue set. If DCBx protocol runs, the outcome of the protocol exchange might translate into changes in scheduler configuration.
  - See Section 7.8.4.1 for more details.
- **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 XL710 might be enabled for APM wake during power up. See Section 5.4.1 for more details on how APM Wake is configured.
  - EEE might be enabled once EMP firmware is initialized. See Section 5.3.1.4 for more details.
  - In addition, various configuration fields are loaded from the NVM to set up later operation of power management capabilities such as DMA coalescing. This capability is not enabled pre-boot.
- **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.

- **FCoE** - Some configuration data is loaded from the NVM during power on. FCoE offload capabilities are not enabled during power on and BIOS initialization stages.

4.2.2 BIOS init

This section describes how BIOS code might update the XL710's configuration and the capabilities for network boot. The following sections are provided:

- Section 4.2.2.2 describes how BIOS code might change the XL710's configuration on each boot
- Section 4.2.2.3 describes some aspects of supporting network boot
- Section 4.2.2.4 describes the specific setting of each of the XL710's components required for the BIOS Init stage

4.2.2.1 Initial state

The state of the XL710 when BIOS begins to access it is described in Section 4.2.2.4.

BIOS code must check that the EMP completed device initialization. This is done through the Get Version AQ command described in Section 7.10.12.1.

4.2.2.2 Non-persistent configuration

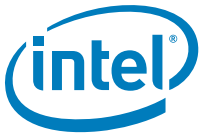
During power up, the XL710 is factory configured from the NVM. However, the factory configuration might be overridden in two possible ways:

- Persistent configuration - System tools (such as software agents and SMCLP commands) might write into the NVM and change the factory defaults. It is also possible to add alternate values into the NVM so that the factory defaults are preserved and might be restored at a later time.
- Non-persistent configuration - An on-die alternate RAM structure is provided for alternate configuration that is not maintained between cold resets. 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). The information in the alternate structure is kept during any reset other than cold reset, and is loaded into the XL710 following the appropriate resets (see the paragraphs that follow), overriding the respective configuration loaded previously from the NVM. This section describes the details of how the alternate structure is handled.

The following mechanisms are provided:

- The configuration is written into an on-die alternate structure:
  - The alternate structure is 8 KB, partitioned into 32-bit entries. Accessing the structure is done by addressing 32-bit entries. An address is therefore 11 bits (2 K Dwords), where address 0x000 points to the beginning of the 8 KB memory.
- The alternate structure is loaded into the device functional units in the following cases:
  - As part of the SMASH/CLP programming
    - Legacy BIOS: once the Init function is called for the 1st time in the flow
    - UEFI: once a Done Alternate Write command is received for each enabled LAN port

Note:



  - — As part of executing the following resets: PERST#, PCIR, EMPR, global reset, core reset. In other words, the contents of the alternate structure are only reset on cold resets
- SMASH/CLP configuration might not be followed by a PCIe reset, and designers can't rely on a PCIe reset to load the alternate module into the XL710.
- Option ROM code and UEFI drivers communicate with the XL710 via a set of AQ commands:
  - — The Read Alternate - Direct and Read Alternate - Indirect commands read from the alternate structure
  - — The Write Alternate - Direct and Write Alternate - Indirect commands write into the alternate structure
  - — The Done Alternate Write command indicates to the XL710 that the CLP strings phase is done (for the entire device in Legacy BIOS mode and per LAN port in UEFI mode).
  - — The Set OEM Mode command sets the XL710 to a specific operating mode (used in Legacy BIOS mode only)

4.2.2.2.1 Alternate RAM structure

The alternate structure is 8 KB, partitioned into 32-bit entries. Accessing the structure is done by addressing 32-bit entries. An address is therefore 11-bits (2 K Dwords), where address 0x000 points to the beginning of the 8 KB memory.

The structure is partitioned as follows:

Section	Address (DW)	Size (DW)	Content
Per PF	0 – 1023	1024 - 64 per PF	16 per-PF sections, one per PF. These sections are described in Section 4.2.2.2.1.1 and can be accessed by each PF.
EMP	1024 - 1279	256	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.
Boot configuration	1280 - 2047	768	A boot configuration section used for SAN boot configuration. This section is for software use only and the XL710 is not aware of its internal content. All PFs can access this section (no enforcement by the XL710).

4.2.2.2.1.1 Per PF sections

Table 4-5. Per PF Alt RAM content

Scope	Address (DW)	Contents	Used by	Reserved	PCIe ALT Module in Shadow RAM	GLOBR Registers Auto-Load	CORER Registers Auto-Load
PF	0	Current LAN MAC Address (LS)	HW, SW				X
PF	1	Current LAN MAC Address (MS)	HW, SW				X
PF	2	Current FCoE MAC Address (LS)	HW, SW				
PF	3	Current FCoE MAC Address (MS)	HW, SW				
PF	4	Current WWNN Prefix	SW				
PF	5	Current WWPN Prefix	SW				
Reserved	6-63	Reserved	N/A				

Note: The following registers are not reset by VFLR and need to be configured by the PF or VF driver in case of a change to a new configuration (such as VF OS transition): VFRDH/T, VFTDH/T, VFPSRTYPE, VFSRRCTL, VFRXDCTL, VFTXDCTL, VFTDWBAL/H, VFDCA_RXCTRL, VFDCA_TXCTRL.

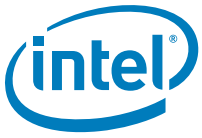
Each entry is associated with one or more NVM modules and is loaded into the XL710 following reset events that load these modules. Loading into the PCIe units is an exception. Parameters to be loaded into PCIe units are written by the EMP into the PCIe ALT auto-load module in shadow RAM. This is done as part of the BIOS initialization flow. On later resets, the XL710 automatically loads the module into the PCIe units, saving the need for the EMP to intervene.

The PCIe ALT auto-load module is a regular hardware type 1 auto-load section as described in Section 6.1.3.1. The content is dynamically created according to the registers that requires an auto-load value different than the default value or the value loaded from the NVM.

4.2.2.2.1.1.1 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

4.2.2.2.1.1.2 Current SAN MAC address (offset 0x2, 0x3)

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.

4.2.2.2.1.1.3 Current WWNN prefix (offset 0x4)

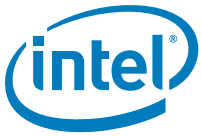
Field	Bit(s)	Description
WWNN	15:0	Contains the current WWNN prefix to be used to create the WWNN of the station.
Reserved	30:16	Reserved.
Valid	31	Valid Bit. 0b = WWNN is invalid and should be skipped. 1b = WWNN is valid and should be used.

4.2.2.2.1.1.4 Current WWPN prefix (offset 0x5)

Field	Bit(s)	Description
WWPN	15:0	Contains the current WWPN prefix to be used to create the WWNN of the station.
Reserved	30:16	Reserved.
Valid	31	Valid Bit 0b = WWPN is invalid and should be skipped 1b = WWPN is valid and should be used.

4.2.2.2.1.1.5 Min BW (offset 0xE)

Field	Bit(s)	Description
PF Protocol	2:0	PF Protocol - Determines if the function is enumerated and its type 000b - Function is not enumerated 001b - Ethernet only 010b - Ethernet 011b - iSCSI 100b - FCoE Else - reserved
Reserved	30:3	Reserved
Valid	31	Valid bit 0b - DW is invalid and should be skipped 1b - Entry is valid and should be processed



Field	Bit(s)	Description
Min BW	8:0	If *Relative BW* = 0x0: Min BW - Minimum guaranteed bandwidth for the PF in multiples of 128 Mb/s: 0x00 = 0 Mb/s. 0x01 = 128 Mb/s. ... 0x190 and above - 40 Gb/s. If *Relative BW* = 0x1: Minimum Tx bandwidth allocation for the PF expressed in percent of the maximum physical port link speed. The percent value ranges from 0 to 100: 0x00 = 0%. 0x01 = 1%. ... 0x64 and above = 100%.
Reserved	29:9	Reserved.
Relative BW	30	0b = Absolute BW. 1b = Relative BW.
Valid	31	Valid Bit. 0b = Dword is invalid and should be skipped. 1b = Entry is valid and should be processed.

4.2.2.2.1.1.6 Max BW (offset 0xF)

Field	Bit(s)	Description
Min BW	8:0	If *Relative BW* = 0x0: Max BW - Maximum guaranteed bandwidth for the PF in multiples of 128 Mb/s: 0x00 = Reserved. 0x01 = 128 Mb/s. ... 0x190 and above = 40 Gb/s. If *Relative BW* = 0x1: Max BW - Maximum Tx bandwidth allocation for the PF expressed in percent of the maximum physical port link speed. The percent value ranges from 0 to 100: 0x00 = 0%. 0x01 = 1%. ... 0x64 and above = 100%.
Reserved	29:9	Reserved.
Relative BW	30	0b = Absolute BW. 1b = Relative BW.
Valid	31	Valid Bit: 0b = Dword is invalid and should be skipped. 1b = Entry is valid and should be processed.

4.2.2.2.2 AQ commands

The Table 4-6 lists the different AQ commands used to manage the alternate structure.

Table 4-6. List of AQ commands for the Alternate Structure

Command	Opcode	Brief Description	Detailed Description
Write Alternate - Direct	0x0900	Write up to two parameters into the alternate structure.	Section 4.2.2.2.2.1
Write Alternate - Indirect	0x0901	Write a block of parameters into the alternate structure.	Section 4.2.2.2.2.2
Read Alternate - Direct	0x0902	Read up to two parameters from the alternate structure.	Section 4.2.2.2.2.3
Read Alternate - Indirect	0x0903	Read a block of parameters from the alternate structure.	Section 4.2.2.2.2.4
Done Alternate Write	0x0904	Indication that all CLP strings (for the entire XL710 in legacy BIOS mode and per LAN Port in UEFI mode) have been sent to the XL710.	Section 4.2.2.2.2.5
Set OEM Mode	0x0905	Transition the device to a specific OEM mode (used in legacy BIOS mode).	Section 4.2.2.2.2.6
Clear Port Alternate	0x906	Clear content of alternate RAM relevant to this port.	

4.2.2.2.2.1 Write alternate - direct

The Write Alternate - Direct command writes to the alternate structure up to two parameters.

Table 4-7. Write alternate - direct command

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0900	Command opcode.
Datalen	4-5	0x00	N/A
Return Value/ VFID	6-7	0x00	N/A
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
First Parameter Address	16-19		Address should be within the range allocated to the function inside the alternate structure. Accessing the alternate structure is done by addressing 32-bit entries.
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		

Table 4-8. Completion for the write alternate - direct command

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0900	Command opcode.
Datalen	4-5	0x00	N/A

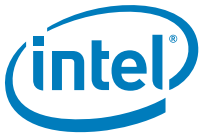


Table 4-8. Completion for the write alternate - direct command

Name	Bytes.Bits	Value	Remarks
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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Reserved	16-31		Might contain the values sent in the original command.

4.2.2.2.2.2 Write alternate - indirect

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 4-9. Write alternate - indirect command

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0901	Command opcode.
Datalen	4-5		Size of buffer accompanying the command (in bytes).
Return Value/ VFID	6-7	0x00	N/A
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Alternate Structure Address	16-19		Lowest address to be written into the alternate structure. Accessing the alternate structure is done by addressing 32-bit entries.
Alternate Structure Length	20-23		Number of Dwords to be written into the alternate structure.
Data Address High	24-27	Buff Addr	High bits of buffer address.
Data Address Low	28-31	Buff Addr	Low bits of buffer address.

The following completion is sent for the Write Alternate - Indirect command:

Table 4-10. Completion for the write alternate - indirect command

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0901	Command opcode.
Datalen	4-5	0x00	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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Reserved	16-31		

4.2.2.2.2.3 Read alternate - direct

The Read Alternate - Direct command reads from the alternate structure up to two parameters.

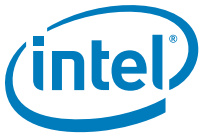
Table 4-11. Read alternate - direct command

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0902	Command opcode.
Datalen	4-5	0x00	N/A
Return Value/ VFID	6-7	0x00	N/A
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
First Parameter Address	16-19		Address should be within the range allocated to the function inside the alternate structure. Accessing the alternate structure is done by addressing 32-bit entries.
Reserved	20-23	0x00	
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	0x00	

The following Completion is sent for the Read Alternate - Direct command:

Table 4-12. Completion for the read alternate - direct command

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0902	Command opcode.
Datalen	4-5	0x00	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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
First Parameter Address	16-19		Copied from 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.



4.2.2.2.2.4 Read alternate - indirect

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.

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0903	Command opcode.
Datalen	4-5		Size of buffer accompanying the command (in bytes).
Return Value/ VFID	6-7	0x00	N/A
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Alternate Structure Address	16-19		Lowest address to be read from the alternate structure. Accessing the alternate structure is done by addressing 32-bit entries.
Alternate Structure Length	20-23		Number of Dwords to be read from the alternate structure.
Data Address High	24-27	Buff Addr	High bits of buffer address.
Data Address Low	28-31	Buff Addr	Low bits of buffer address.

The following completion is sent for the Read Alternate - Indirect command:

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0903	Command opcode.
Datalen	4-5	0x00	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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmwae 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 DWs read from the alternate structure.
Data Address High	24-27	Buff Addr	High bits of buffer address.
Data Address Low	28-31	Buff Addr	Low bits of buffer address.

4.2.2.2.2.5 Done alternate write

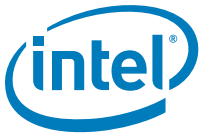
The Done Alternate Write command indicates to the XL710 that the CLP strings have been sent to it:

- Legacy BIOS mode - sent once per device after all CLP strings have been sent to the XL710 (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 XL710's functional units.

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0904	Command opcode.
Datalen	4-5	0x00	N/A
Return Value/ VFID	6-7	0x00	N/A
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
BIOS Mode	16.0	See remarks	0b = Legacy BIOS. 1b = UEFI.
Reserved	16.7 - 16.1	0x00	Reserved.
Reserved	17-31	0x00	Reserved.

The following completion is sent for the Done Alternate Write command:

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0904	Command opcode.
Datalen	4-5	0x00	N/A
Return Value/ VFID	6-7		ENOSPC - more than 128 VFs are requested
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Reserved	16.0	0b	Reserved.
Return Flags	16.1		Reset Needed. When set, indicates that software should do a global reset for the alternate RAM content to take effect.
Reserved	16.2-31	0x00	



4.2.2.2.2.6 Set OEM mode

The Set OEM Mode command sets the XL710 to a specific operating mode. For example, standard mode or some OEM specific mode.

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0905	Command opcode.
Datalen	4-5	0x00	N/A
Return Value/ VFID	6-7	0x00	N/A
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
OEM Mode	16-19		0x0 - No_CLP.
Reserved	20-31	0x00	

The following completion is sent for the Set OEM Mode command:

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0905	Command opcode.
Datalen	4-5	0x00	N/A
Return Value/ VFID	6-7		
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Reserved	16-31	0x00	

4.2.2.2.2.7 Clear port alternate write

The Clear Port Alternate command indicates to the XL710 that the alternate sections of all PF tied to the port. The port is inferred from the PF that sent the command.

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0906	Command opcode.
Datalen	4-5	0x00	N/A
Return Value/ VFID	6-7	0x00	N/A

Name	Bytes.Bits	Value	Remarks
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Reserved	16-31	0x00	

The following completion is sent for the Clear Port Alternate command:

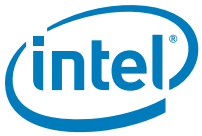
Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0905	Command opcode.
Datalen	4-5	0x00	N/A
Return Value/ VFID	6-7		
Cookie High	8-11	Cookie	Opaque value is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
Reserved	16-31	0x00	

4.2.2.2.3 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 XL710 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 XL710 (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)

        Send "Set OEM Mode" command with a "No_CLP" value

    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
```

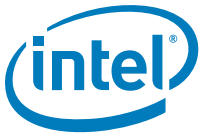
```
    End-if

    // 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

Continue with INIT code...

End SMCLP INIT Entry Point
```

4.2.2.2.4 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 XL710 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 XL710 (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-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
```

4.2.2.2.5 Processing the alternate structure

If a Done Alternate Write AQ command is received, the *GLNVM_GENS.ALT_PRST* bit is set by the XL710. This bit indicates that an alternate structure exists in the XL710. As a result of each Done Alternate Write AQ command, the EMP loads the relevant alternate structure parameters into the XL710 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

4.2.2.3 Network boot

Each PF can be a boot function and can independently support PXE, iSCSI boot or FCoE boot:

- All functions are PXE boot (such as up to 16 such functions)
- Up to four functions might be iSCSI or FCoE boot functions

4.2.2.4 Device state

This section is limited to changes in the XL710's state made in the BIOS initialization stage. For units not mentioned here, behavior is as described in Section 4.2.1.6.

4.2.2.4.1 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.

4.2.2.4.2 LAN

Expansion ROM code might set LAN QPs for network boot. The sequence of setting a LAN QP is described in Section 8.2.2.

Since DCB is required for FCoE boot, expansion ROM code should probe the status of DCBx and read the DCBx results needed for FCoE operation. See Section 7.7.3 for the exact flow.

4.2.2.4.3 Interrupts

Interrupts need to be set if used for admin queue or LAN queues operation. See Section 7.5.1.1 for the exact flow.

4.2.3 Driver load

4.2.3.1 Introduction

4.2.3.1.1 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 XL710 to initialize it for normal operation. The major initialization steps are:

1. Device driver probes the XL710 for resource allocations.
2. Disable interrupts.
3. Initialize the admin queue - see Section 7.10.3.
4. Initialize HMC - see Section 7.9.2, and Section 7.9.3.
5. Initialize MAC/PHY - see Section 3.2.2.9.
6. Initialize power management - see Section 5.3.1.4 (EEE), Section 5.4.5 (Wake Up).
7. Initialize DCB (including Rx-PB).
8. Initialize the switch and Tx scheduler - see Section 7.4.9.4 and Section 7.8.4.19.
9. Initialize statistics by reading the counter’s initial values to serve as a baseline.
10. Initialize 1588.
11. Enable interrupts.

At this point, the XL710 is ready to initialize VSIs and LAN/FCoE flows. These are done dynamically during operation:

- Initialize VSI.
- Initialize LAN QP; including its interrupts (see Section 8.2.2) or initialize FCoE exchange (see Section 9.4.2) or configure filters.

4.2.3.1.2 Driver load (SR-IOV)

The XL710 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 require such functionality. Figure 4-10 shows the high-level initialization flow for virtualized device drivers that use a VF in a guest operating system.

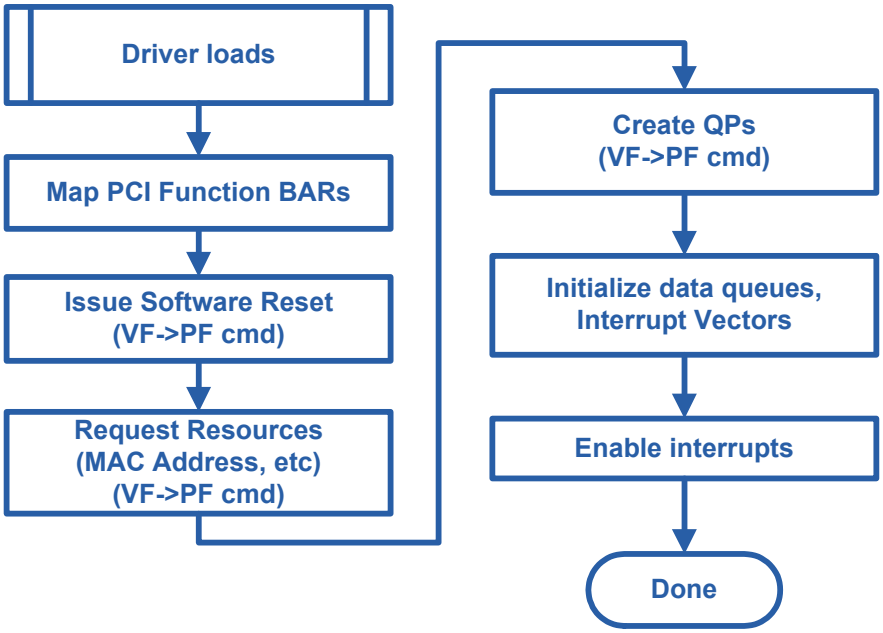


Figure 4-10. Virtualized device driver initialization flow

4.2.3.1.2.1 PF initialization details

In addition to the regular device driver initialization flow described in Section 4.2.3.1.1, 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 7.4.9.5.3.1) in order to get the SEID number to which this PF is connected. This SEID might be the port or an S-channel.
   b. Create a VEB/VEPA using the Add VEB command (Section 7.4.9.5.6.1). The control port can be either the original VSI of the PF or a dedicated VSI.
   c. Connect the switch in place of the current VSI using the Insert Element command (Section 7.4.9.5.6.1).
   d. Connect default and mirror ports as needed using the Add VSI and Connect Elements commands (Section 7.4.9.5.4.1 and Section 7.4.9.5.4.3).
   e. Define mirroring rules using the Mirroring Rules commands (Section 7.4.9.5.9).
   f. Activate malicious driver protection through the GL_MDCK_RX GL_MDCK_TDAT and GL_MDCK_TCMD registers
   g. If the PF owns the port, then it can activate the storm control using the "Set Storm Control Configuration command (Section 7.4.9.5.10.1).

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 7.8.4).
   c. If the virtual port is a VF, allow VF access to the network by clearing the associated bit in *GL_VIRT_VFLRE* register.

4.2.3.1.2.2 VF initialization details

This section describes the flow used to initialize a VF. It refers to various stages shown in Figure 4-10. Only stages involving the hardware are detailed.

4.2.3.1.2.2.1 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 4.1.2.5.

4.2.3.1.2.2.2 Request resources and create initialize data queues

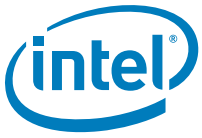
On top of the resources statically allocated to a VF (interrupts, RSS table, FCoE contexts, etc.), a VF might have a set of VSIs. Each VSI contains objects such as:

- User priorities (transmit queue groups).
- Queue pairs
- Queuing filters

VSIs can be requested either for LAN or FCoE.

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	Notes
1	VSI and UPs	Allocate VSIs according to allocation policies using the Add VSI AQ command (Section 7.4.9.5.4.1).	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 MAC and VLAN and MAC, VLAN pairs using the forwarding table configuration commands (Section 7.4.9.5.8).	
3	Data queue pairs	Create the requested queue contexts as described in Section 8.4.3.1 and Section 8.3.3.1.	



4.3 Device/port/function disable

4.3.1 General

The XL710 provides several mechanisms to disable PCI functions and Ethernet ports:

- Through NVM configuration
- Bases on power management policy
- Through strapping pins

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 changes 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.

4.3.2 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 the XL710 PCI devices, PCI functions, or external Ethernet ports and the associated PCI function. It enables end users more control over system resource management and avoids conflicts with add-in NIC solutions. The XL710 provides support for selectively enabling or disabling one or more LAN PCI device(s) in the system.

The XL710 provides strapping pins to disable its external Ethernet ports and/or PCI functions:

— One 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.

— One 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.

Some guidelines on usage of DEV_DIS_N and PCI_DIS_N:

- During power up, the PCI_DIS_N and DEV_DIS_N pins are ignored until the NVM is read. From that point, the XL710 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 XL710 in order to re-enable any disabled PCI functions or external Ethernet ports.
- The PCI_DIS_N and DEV_DIS_N pins should maintain their state during system reset and system sleep states. It should also insure the proper default value on system power up. For example, one could use a GPIO pin that defaults to 1b (enable) and is on system suspend power (such that it maintains its state in S0-S5 ACPI states).

4.3.3 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
- Soft SKU commands
  - Soft SKU commands might enable or disable LAN ports. 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 XL710 (like before the first PCIe configuration cycle to the XL710).
  - EMP updates the configuration of the ports enabled or disabled through the Soft SKU command by writing to the PRTGEN_CNF.PORT_DIS register bits
- Power management
  - If the Ethernet ports are not required in Dr state (such as if manageability is disabled and APME WoL is disabled as well), then all ports are disabled during Dr state. See Section 5.2.3.4.
- 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.

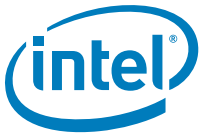
The result of the previous mechanisms is reflected in the PRTGEN_STATUS.PORT_VALID bit (per port), which denotes if the port is enabled. A bit is cleared (port disabled) if at least one of the previous mechanisms disables the port. The port is then powered down (including MAC, PCS, PHY).

If all ports are disabled, the XL710 is put in a power-down, reset state. Specifically, the XL710 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).

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, re-activating it, does not require any reset.

A port is active (PRTGEN_STATUS.PORT_ACTIVE = 1b) based on the following rules:

1. PRTGEN_STATUS.PORT_VALID = 1b, as previously described above AND the XL710 is in D0 state (PMCSR.PowerState = D0).
   - If PRTPM_GC.EMP_LINK_ON is set to 1b (such as the interface is being used for manageability) then the link is kept on. Specifically, hardware ignores disabling the link using the PRTGEN_CNF.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.

2. PRTGEN_STATUS.PORT_VALID = 1b, as previously described, AND the XL710 is in Dr state (PMCSR.PowerState = D3).
   — If port is enabled for wake-up, follow the description in Section 5.4.
   — Else, behavior is defined by the EMP_LINK_ON bit:
   - The hardware default value for PRTPM_GC.EMP_LINK_ON is 0b (link should be down as it is not required for EMP functionality)
   - BMC might enable manageability by sending the appropriate command (see Section 10.7.2.2 and Section 10.5.9). As a result, EMP transitions to pass-through manageability-on state, enables the respective port (if disabled) by setting the PRTPM_GC.EMP_LINK_ON bit for the port, and starts the link bring-up sequence.
   - If the channel is disabled by the BMC (by sending the appropriate command (see Section 10.7.2.2 and Section 10.5.9), and EMP has no other needs for the LAN port in Dr state (like proxy), then the EMP reverts the PRTPM_GC.EMP_LINK_ON bit for the port 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).

Another related per-port configuration is:

- The PFGEN_PORTNUM.PORT_NUM per-PF register field (loaded from the NVM) that associates a PF with a port.

4.3.4 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 hardwired to enabled).
  — When cleared, the respective function is enabled
  — The hardware default value is for function 0 to be enabled and other functions to be disabled
- Strapping through PCI_DIS_N
  — When the PCI_DIS_N pin is asserted low, the PFPCI_FUNC.ALLOW_FUNC_DIS per-PF register bit (loaded from the NVM) determines if the function is disabled
  — If this bit is set to 0b, PCI_DIS_N has no effect on the PCI function
  — The hardware default value is 0b (keep enabled)
- Strapping through DEV_DIS_N
  — When the DEV_DIS_N pin is asserted low, the PFPCI_FUNC.DIS_FUNC_ON_PORT_DIS per-PF bit (loaded from the NVM) determines if the function is disabled
  — If this bit is set to 0b, DEV_DIS_N has no effect on the PCI function
  — The hardware default value is 0 (do not disable)
- Soft SKU commands
  — The soft SKU commands are executed either before or after PERST# was de-asserted and in any case, before the first access by BIOS to the XL710 (such as before the first PCIe configuration cycle to the XL710)
  — 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 hardwired to enabled, so is not affected by this mechanism
- Another configuration involved with enabling functions is:
  - — The GLGEN_PCIFCNCNT register reflects the number of enabled PFs as loaded from the NVM (determined by PFPCI_FUNC.FUNC_DIS only). The value is loaded from the NVM.

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

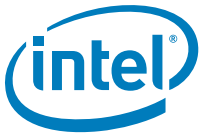
The Ethernet ports associated with disabled PCI functions are still available for manageability purposes.

4.3.5 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. 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 XL710 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 XL710 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.



4.3.5.1 Multi-function advertisement

If all but one of the PCI functions are disabled, the XL710 is no longer a multi-function device. The XL710 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 XL710 reports a 0x0 in this field to signify single-function capability.

4.3.5.2 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 XL710 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, then 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.

4.3.5.3 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 XL710 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.

4.4 Shared resource management

The XL710 is a device with multiple external Ethernet ports and that supports multiple PCI functions per external port. Several on-chip resources are shared between device drivers that load the PCI functions exposed by the XL710. Additionally, some resources for SR-IOV VFs are managed by the associated PFs. In general, the XL710 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.

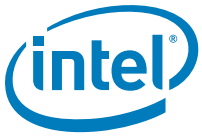
The XL710 handles shared resources using a number of allocation mechanisms and/or access control mechanisms. Table 4-13 lists the supported mechanisms. Table 4-13 lists the mechanism used by each shared resource.

Table 4-13. Supported Mechanism

Class	Description
Dedicated	This resource is associated with a particular XL710 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 XL710 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 XL710 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.

4.4.1 Resource profiles

The concept of resource profiles is used in order to make the XL710's resource allocation mechanisms easier to understand. Resource profiles dictate how resources listed in Table 4-13 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 4-13). 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 4-13 lists which input parameters effect the allocation of each resource.



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

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.

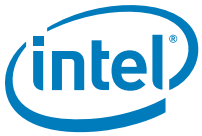
The table below lists a few sample sets of typical input parameters that are used with the XL710. The initial values of these resources are found in the NVM but might be overridden by OEM specific configuration mechanisms followed by a PCI reset. Note that the device driver usage model can be changed and locked by the first device driver to load following a PCI reset. All other inputs are set at PCI reset.

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	128	Default
Dual Port Default	2	2	PF0 -> Port 0 PF1 -> Port 1	128	Default
Quad Port Default	4	4	PF0 -> Port 0 PF1 -> Port 1 PF2 -> Port 2 PF3 -> Port 3	128	Default
Dual Port Multiple PF Per Port	2	8	PF0 -> Port 0 PF1 -> Port 1 PF2 -> Port 0 PF3 -> Port 1 PF4 -> Port 0 PF5 -> Port 1 PF6 -> Port 0 PF7 -> Port 1	128	Default
Single Port SR-IOV VF Primary	1	1	PF0 -> Port 0	128	SR-IOV PF Primary
Single Port SR-IOV Even Distribution	1	1	PF0 -> Port 0	128	SR-IOV Even Distribution

The table below lists the resulting resource distribution for each of the input scenarios listed in the table above. 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 was to change while an active device driver was running, unpredictable results can occur. For example, re-partitioning the host memory cache segment descriptor table while a device driver was active causes

the XL710 to improperly interpret the host memory used for caching resource objects. The XL710 provides various mechanisms (such as the HMC resource profile locking mechanism) to prevent software from unknowingly causing these types of reconfigurations.

Resource	# LAN QPs Per PF	#LAN QPs Per VF	#MSI-X Vectors Per PF	#MSI-X Vectors Per VF	Default Internal Switch Topology	HMC Resources Per PF	HMC Resources Per VF
Single Port Default	512	8	129	5	PF0 -> Port 0	4096 SDs	0 SDs
Dual Port Default	256	8	129	5	PF0 -> Port 0 PF1 -> Port 1	2048 SDs	0 SDs
Quad Port Default	128	8	129	5	PF0 -> Port 0 PF1 -> Port 1 PF2 -> Port 2 PF3 -> Port 3	1024 SDs	0 SDs
Dual Port Multiple PF Per Port	64	8	65	5	PF0 -> VSI0 PF1 -> VSI1 PF2 -> VSI2 PF3 -> VSI3 PF4 -> VSI4 PF5 -> VSI5 PF6 -> VSI6 PF7 -> VSI7 VSI0,2,4,6 -> S-Comp0 VSI1,3,5,6 -> S-Comp1 S-Comp0 -> Port 0 S-Comp1 -> Port 1	512 SDs	0 SDs
Single Port SR-IOVVF Primary Distribution	128	8	129	5	PF0 -> Port 0	139 SDs	123 SDs
Single Port SR-IOV Even Distribution	128	8	129	5	PF0 -> Port 0	349 SDs	220 SDs



5.0 Power Management

This section defines how Power Management is implemented in the XL710.

5.1 Power Targets and Power Delivery

See Section 14.4 for the current consumption and see Section 14.3.1 for the power supply specification.

5.2 PCIe Power Management

5.2.1 Auxiliary Power Usage

The XL710 uses the AUX_PWR pin as an indication that auxiliary power is available to it. The XL710 uses this indication to advertise D3cold 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 amount of power required for the function, which includes the entire Network Interface Card (NIC) is advertised in the Power Management Data register, which is loaded from the NVM.

If AUX power usage during D3cold is supported (as indicated below), all The XL710's sticky bits will preserve their values and will only get reset by the power up reset (detection of power rising).

AUX Power usage is during system low power state (D3cold).

When AUX power is applied to The XL710, the actual usage of AUX power during D3cold is controlled through the below configuration:

NVM loaded bits:

- EMP_LINK_ON - Bit per port loaded from NVM to PRTPM_GC register indicates if the relevant port link should be established for EMP functionality. The XL710 will provide 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 XL710 will provide the proper clocking to establish the required port link/s at this state.

PCIe configuration bits:

PME_En - The PME_En bit of the PMCSR PCI config space is used to determine if The XL710 is allowed to consume AUX power for WoL.

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

5.2.2 PCIe Link Power Management

The PCIe link state follows the power management state of the device. Since the XL710 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 XL710 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.

The XL710 support for Active State Link Power Management (ASPM) is reported via the PCIe *Active State Link PM Support* field in GLPCI_PMSUP register loaded from NVM.

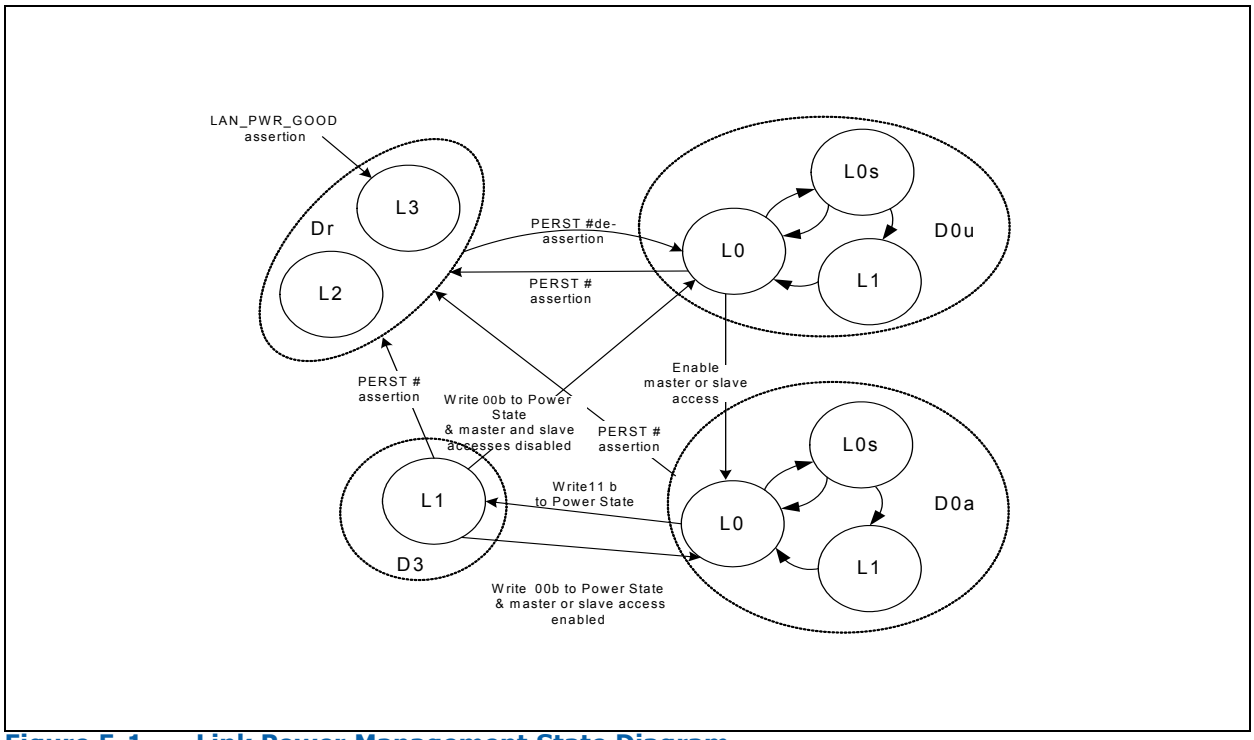


Figure 5-1. Link Power Management State Diagram

While in L0 state, the XL710 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. The default value is loaded on any reset of the PCI configuration registers from/to the GLPCI_PMSUP.ASPM_SUP NVM/register field (LS bit).
- 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 XL710 shares the same reference PCIe clock with its partner across the link, and when the XL710 uses a different reference clock than its partner across the link. The XL710 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 XL710 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 which is loaded from RO PCIe LCB module in NVM. The XL710 will then transition 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 enable - Indicates support for ASPM L1 in the PCIe configuration space (loaded into the Active State Link PM Support field). The default value is loaded on any reset of the PCI configuration registers from/to the GLPCI_PMSUP.ASPM_SUP NVM/register field (MS bit).

- 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.

5.2.3 Power States

The XL710 supports the D0 and D3 architectural power states as described earlier. Internally, the XL710 supports the following power states:

- D0u (D0 un-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 D3cold state. Dr state is entered when PE_RST_N is asserted or a PCIe in-band reset is received.

Figure 5-2 shows the power states and transitions between them.

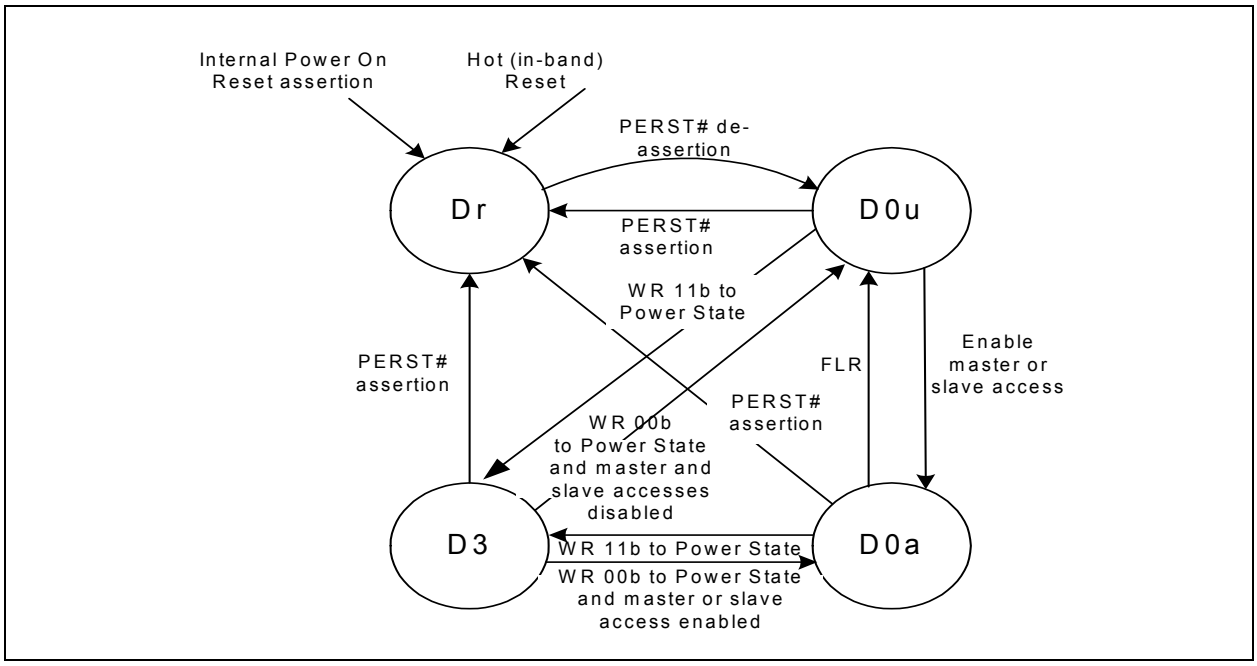
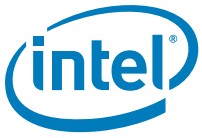


Figure 5-2. Power Management State Diagram

5.2.3.1 $D0_{uninitialized}$ State

The D0u state is an architectural low-power state.

When entering D0u:



- The PCIe Auto-load section of the NVM is read and wake up is disabled. However APM wake up is enabled (See additional information in Section 5.4.1), if all of the following register bits are set:
  — The *PFPM_APM.APME* bit is set to 1b for the appropriate PF.
  — The *PFPM_WUC.EN_APM_D0* bit is set to 1b.

5.2.3.1.1 Entry to a D0u State

D0u is reached from either the Dr state (on de-assertion of internal PE_RST_N) or the $D3_{hot}$ 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 XL710 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 link. Once this is done, configuration software can access the XL710.

On a transition from D3 to D0u state, XL710 PCI configuration space is not reset (since the No_Soft_Reset bit in the *PMCSR* register is fixed to 1b). However following a move to D0a state, the XL710 requires that the software driver perform a full re-initialization of the function.

5.2.3.2 D0active State

A XL710 Function enters the $D0_{active}$ state whenever any single or combination of the Function's Memory Space Enable, I/O Space Enable, or Bus Master Enable bits have been enabled by system software 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.

Note: Two notes:

1. If the *PFPM_WUC.EN_APM_D0* is cleared to 0, then:
   a. 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.
   b. Any APM wake up previously active remains active when moving from D3 to D0.
   c. WAKE# pin is never toggled for an APM wake event when a function is in D0.
2. 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 will not be enabled. Following entry into D0 software device driver can activate other wake-up filters by writing to the Wake Up Filter Control (*PFPM_WUFC*) register.

5.2.3.2.1 Entry to 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.

5.2.3.3 D3 State (PCI-PM D3hot)

The XL710 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 XL710 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 XL710 does not generate master cycles.

Configuration, message requests, and MCTP over PCIe VDMs are the only TLPs accepted by a function in the D3hot state. All other received requests must be handled as unsupported requests, and all received completions are handled as unexpected completions. If an error caused by a received TLP (such as an unsupported request) is detected while in D3hot, and reporting is enabled, the link must be returned to L0 if it is not already in L0 and an error message must be sent. See section 5.3.1.4.1 in the PCIe Base Specification.

5.2.3.3.1 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 XL710; it masks all interrupts, 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 has been enabled Software device driver should set up the required wake up functionality using the flow described in Section 5.4.5 prior to the D3 transition.

Note:

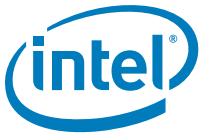
1. The *PMCSR.PME_En* bit setting can be overridden via the *PFPM_APM.APME* bit.
2. If operation during $D3_{cold}$ is required, even when Wake capability is not required (e.g. for manageability operation), system should also set the *Auxiliary (AUX) Power PM Enable* bit in the PCIe *Device Control register*.

If all PCI functions are programmed into D3 state, the XL710 attempts to brings its PCIe link into the L1 link state. As part of the transition into L1 state, the XL710 suspends scheduling of new host TLPs, MCTP over PCIe VDMs will not be suspended. the XL710 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 D3cold state, the device driver might disable up to three LAN ports out of four (LAN disable) and/or transition the remaining link(s) to GbE or 100M speed (if supported by the network interface).

5.2.3.3.2 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 D3cold 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 XL710 is always set to 1b, to indicate that The XL710 does not performs an internal reset on transition from D3hot to D0 so that transition will 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 D3hot to D0 (the Function will be in the D0 state).

The Function will be reset if the Link state had transition to the L2/L3 Ready state, on transition from D3cold to D0, if FLR is asserted or if transition D3hot to D0 is caused by assertion of PCIe reset (PE_RST pin).

5.2.3.4 Dr State (D3cold)

Transition to Dr state is initiated on several occasions:

- On system power up — Dr state begins with the assertion of the internal power detection circuit (LAN_PWR_GOOD) and ends after the de-assertion of PE_RST_N and completion of the NVM auto read.
- On transition from any state — During operation the system can assert PE_RST_N at any time. In an ACPI system, a system transition to the G2/S5 state causes a transition to Dr state.

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 XL710 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 the following section.

5.2.3.4.1 Dr Disable Mode

The XL710 enters a Dr disable mode on transition to D3cold state when it does not need to maintain any functionality. The conditions to enter either state are:

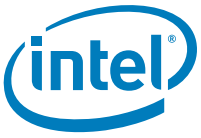
- The device (all PCI functions) is in Dr state.
- APM WOL is inactive for all PCI functions.
- Pass-through manageability is disabled.

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 or Proxying 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 XL710 power. If the platform removes the XL710 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 XL710.

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.



5.2.3.4.2 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, then APM wake up is enabled (for each port independently).
- If the *MNG Enable* bit in the NVM word is set, pass-through manageability is not enabled.
- Each of the LAN ports can be enabled if required for WoL or manageability. See Section 5.3 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.

- FW_allows_PG_in_S5 =
- (Manageability Mode = None) OR
- (Redirection Sideband Interface = “MCTP over SMBus” AND OEM Capabilities.CEM Capable)

5.3 Network Interfaces Power Management

The XL710 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 5.2.1.

When the XL710 is in low-power state, the XL710 asserts the respective TX_DIS pin (Connected to an SDP pin) to enable powering down an external PHY or optical module as well.

5.3.1 Energy Efficient Ethernet (EEE)

the XL710 enters EEE (Energy Efficient Ethernet) Low Power Idle (LPI) mode on transmit or receive independently whenever the device detects no data is scheduled for transmission, or the link partner indicates no data is pending for reception

EEE (Energy Efficient Ethernet) Low Power Idle (LPI) mode defined in IEEE802.3az enables power saving by switching off part of XL710 functionality when no data needs to be transmitted or/and received. Decision on whether XL710 transmit path should enter Low Power Idle mode or exit Low Power Idle mode is done according to need to transmit. Information on whether Link Partner has entered Low Power Idle mode is detected by XL710 and utilized for power saving in the receive circuitry.

When no data needs to be transmitted a request to enter transmit Low Power Idle is issued on the internal xxMII TX interface causing the Backplane PHY to transmit sleep symbols for a predefined period of time followed by a quiet period. During LPI the Backplane PHY periodically transmits refresh symbols that are used by the link partner to update adaptive filters and timing circuits in order to maintain link integrity. This quiet-refresh cycle continues until transmission of 'normal inter-frame' encoding on the internal xxMII TX interface. The Backplane PHY communicates to the link partner the move to link active state by sending Wake symbols for a predefined period of time. The PHY then enters normal operating state where data or idle symbols are transmitted.

In the receive direction, entering Low Power Idle mode is triggered by the reception of sleep symbols from the link partner. This signals that the link partner is about to enter Low Power Idle mode. After sending the sleep symbols, the link partner ceases transmission. When the Link partner initiates the move to LPI the Backplane PHY indicates "assert low power idle" on the internal xxMII RX interface and the XL710 receiver disables functionality to reduce power consumption.

Figure 5-3 and Table 5-1 illustrate the general principles of EEE LPI operation on the Ethernet Link.

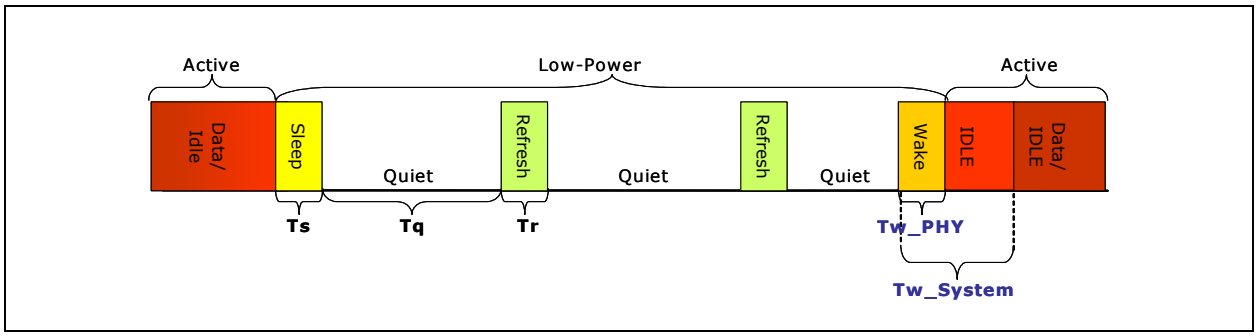


Figure 5-3. Energy Efficient Ethernet Operation

Table 5-1. Energy Efficient Ethernet Parameters

Parameter	Description
Sleep Time (Ts)	Duration PHY sends Sleep symbols before going Quiet.
Quiet Duration (Tq)	Duration PHY remains Quiet before it must wake for Refresh period.
Refresh Duration (Tr)	Duration PHY sends Refresh symbols for timing recovery and coefficient synchronization.
PHY Wake Time (Tw_PHY)	Minimum duration PHY takes to resume to Active state after decision to Wake.
Receive System Wake Time (Tw_System_rx)	Wait period where no data is expected to be received to give the local receiving system time to wake up. The XL710 has no plans to utilize this value for active power reduction.
Transmit System Wake Time (Tw_System_tx)	Wait period where no data is transmitted to give the remote receiving system time to wake up.

5.3.1.1 Conditions to Enter EEE TX LPI

In the transmit direction when network interface is 1000BASE-KX, 10GBASE-KX4 or 10GBASE-KR entry into EEE Low Power Idle (LPI) mode of operation is triggered when one of the following conditions exist:

1. No host transmission is pending, Management does not need to transmit, internal Transmit buffer is empty, the *PRTPM_EEER.TX_LPI_EN* bit is set to 1 and EEE auto negotiation was completed successfully.
   a. When moving out of TX LPI to transmit a 802.3x flow control or priority flow control frame the XL710 will refrain from re-entering into LPI state until the flow control/priority flow control event indication for any UP in that port was released.

When the above condition to enter TX LPI state are detected "assert low power idle" is transmitted on the internal xxMII interface and the XL710 Backplane PHY transmits Sleep symbols on the network interface to communicate to the link partner entry into TX Low Power Idle link state. After Sleep symbols transmission, the Backplane PHY immediately enters the low power quiet mode. In this state the Backplane PHY periodically transitions between quiet link state, where link is idle, to sending refresh symbols until a request to transition link back to normal (active) mode is transmitted on the internal xxMII TX interface (See Figure 5-3).

Note: PRTPM_EEER.TX_LPI_EN configuration is set by FW after FW finished the EEE timer configurations.

Note: Backplane Ethernet LPI allows each link direction to enter sleep, refresh or wake states asymmetric from the other direction.

Note: EEE LPI Status of a XL710 port can be found in the *PRTPM_EEE_STAT* register.

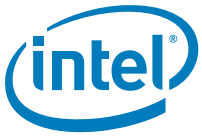
5.3.1.2 Transition from TX LPI to Active Link State

The XL710 will exit TX LPI link state and transition link into active link state when none of the conditions defined in Section 5.3.1.1 exist. To transition into active link state the XL710 transmits:

1. Normal 'inter-frame' encoding on the internal xxMII TX interface for a pre-defined link rate dependant period time of Tw_sys_tx-min as defined by PRTPM_EEEC.TW_PHY . As a result Backplane PHY will transmit Wake symbols for a Tw_phy duration followed by Idle symbols.
2. If the Tw_System_tx duration defined in the *PRTPM_EEER.Tw_system* field is longer than Tw_sys_tx-min the XL710 will continue transmitting the 'inter-frame' encoding on the internal xxMII interface until the time defined in the *PRTPM_EEER.Tw_system* field has expired, before transmitting the actual data. During this period the Backplane PHY will continue transmitting Idle symbols.

Note: When moving out of TX LPI to transmit a 802.3x flow control frame the XL710 will wait only the Tw_phy duration before transmitting the flow control frame. It should be noted that even in this scenario actual data will be transmitted only after the Tw_System_tx time defined in the *PRTPM_EEER.Tw_system* field has expired.

Note: When moving out of TX LPI to transmit a 802.3x flow control or priority flow control frame the XL710 will refrain from re-entering into LPI state until the flow control/priority flow control event indication for any UP in that port was released.



5.3.1.3 EEE Auto-Negotiation

Backplane Auto-Negotiation provides the capability to negotiate Energy Efficient Ethernet capabilities with the Link partner using the Next page mechanism defined in IEEE802.3 Annex 73A. IEEE802.3 Auto-Negotiation is performed at power up, on command from SW, upon detection of a PHY error or following link re-connection.

During the link establishment process, both link partners indicate their EEE capabilities using IEEE802.3 Auto-negotiation. If EEE is supported by both link partners for the negotiated PHY type then the EEE function may be used independently in either direction.

When operating in 1000BASE-KX, 10GBASE-KX4 or 10GBASE-KR modes the XL710 supports EEE Backplane auto-negotiation. EEE capabilities advertised during Auto-negotiation can be modified via the Backplane PHY registers.

Auto-negotiation of EEE capabilities is also supported when the XL710 ports are configured to operate in SGMII mode (see Section 5.3.1.7.2).

5.3.1.4 EEE Link Level (LLDP) Capabilities Discovery

When operating in 1000BASE-KX, 10GBASE-KX4 or 10GBASE-KR modes and link auto-negotiation resolved with EEE support, XL710 supports LLDP negotiation via Software or Firmware, using the EEE IEEE802.1AB LLDP (Link Layer Discovery Protocol) Type, Length, Value (TLV) fields defined in IEEE802.3az clause 78 and clause 79. LLDP negotiation enables negotiation of increased System wake time (Transmit T_w and Receive T_w) to enable improving system energy efficiency.

During pre-boot LLDP is done by FW according to the following flow:

1. EMP determines whether EEE has to be enabled or disabled over a LAN port by reading the "EEE Capability" bit in the corresponding PHY Capability Data Structure in the EMP Settings NVM module.
2. If enabled, EMP loads the settings for the EEE LLDP variables from the EMP Settings module. The relevant variables are: Tw_sys_tx_local, Tw_sys_rx_local, and Tw_sys_rx_fb. These fields can be modified post boot using admin commands. For more information on LLDP protocol, activation and flow refer to section 7.12.
3. If enabled, the PHY goes to an auto-negotiation cycle to define the EEE support with the link partner.
4. Once done, EEE is operational on both Rx and Tx (if enabled on Tx) using default values.

It is assumed that the default latency values are sufficient for system support.

5.3.1.4.1 LLDP Negotiation Actions

Following negotiation of a new System Wake Time Via EEE LLDP negotiation, The PRTPM_EEER.Tw_system field should be updated by FW (when FW does the LLDP) or by SW By sending a *Set Phy Config Command* (See Section 3.2.4.1.1) via the Admin queue (when SW does the LLDP) with the negotiated Transmit T_w time value, to increase the duration where idle symbols are transmitted following move out of EEE TX LPI state before actual data can be transmitted.

In MFP mode a PF SW configuration can only decrease the value of Tw_system_tx using the *Set Phy Config Command* (See Section 3.2.4.1.1) via the Admin queue this is to ensure that minimum LPI exit latency is still preserved for the PFs that requested it.

— Value placed in *PRTPM_EEER.Tw_system* field does not effect transmission of flow control packets. Depending on the technology (1000BASE-KX, 10GBASE-KX4 or 10GBASE-KR) flow

control packet transmission is delayed following move out of EEE TX LPI state only by the minimum Tw_sys_tx time as defined in IEEE802.3az clause 78.5.

Note: The format of the EEE TLV and the meaning of the various TLV parameters can be found in IEEE802.3az clause 78 and clause 79.

5.3.1.5 Programming for Backplane EEE Operation

The following fields should be programed to enable EEE operation on a LAN port when operating in 1000BASE-KX, 10GBASE-KX4 or 10GBASE-KR modes:

1. Backplane PHY registers, if default EEE advertised Auto-negotiation values need to be modified.
2. *Set the EEER.TX_LPI_EN* bit to 1 to enable EEE LPI support, if result of Auto-negotiation at the specified link speed enables entry to LPI. By sending a *Set Phy Config Command* (See Section 3.2.4.1.1) via the Admin queue. In MFP mode if the EEER.TX_LPI_EN was disabled by default or by one of the PFs No PF will be able to enable EEE functionality.

   a.

Following EEE activation an EEE LLDP negotiation cycle should be initiated and the appropriate registers should be programmed (See Section 5.3.1.4).

NOTES:

1. The XL710 waits for at least 1 second following Auto-negotiation (due to reset, Link disconnect or link speed change) and link-up indication before enabling link entry into EEE TX LPI state to comply with the IEEE802.3az specification.
2. Fast Retrain and Local/Remote Fault indication are not considered link disconnect, do not require waiting for 1 second before enabling link to enter into EEE TX LPI.

5.3.1.6 EEE Statistics

The XL710 supports reporting number of EEE LPI TX and RX events via the *PRTPM_RLPIC* and *PRTPM_TLPIC* registers.

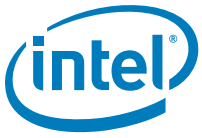
5.3.1.7 External PHY EEE Support

The XL710 supports EEE when connected to an external PHY via a SGMII, XAUIor a SFI interface.

5.3.1.7.1 Programming for External PHY EEE Operation

The following fields should be programed to enable EEE operation on a LAN port when connected to an external copper PHY:

1. After verifying that the external copper PHY supports EEE. Relevant external PHY EEE related registers can be modified, if default EEE advertised Auto-negotiation values need to be updated.
   a. *Set the PRTPM_EEER.TX_LPI_EN* bit to 1 to enable EEE LPI support, if result of Auto-negotiation at the specified link speed enables entry to LPI. By sending a *Set Phy Config Command* (See Section 3.2.4.1.1) via the Admin queue. In MFP mode if the EEER.TX_LPI_EN was disabled by default or by one of the PFs No PF will be able to enable EEE functionality.



2. When operating in XAUI or SFI operating modes the corresponding PRTMAC_PCS_AN_CONTROL4 register bits should be set to 1b to enable EEE operation on the interface regardless of the (external) PHY auto-negotiation results. This is required for external PHYs where EEE auto-negotiation results are not reported to the MAC over the MAC-PHY interface.

Following EEE activation an EEE LLDP negotiation cycle should be initiated and the appropriate registers should be programmed (See Section 5.3.1.4).

1. The XL710 waits for at least 1 second following Auto-negotiation (due to reset, Link disconnect or link speed change) and link-up indication before enabling link entry into EEE TX LPI state to comply with the IEEE802.3az specification.
2. Fast Retrain and Local/Remote Fault indication are not considered link disconnect, do not require waiting for 1 second before enabling link to enter into EEE TX LPI.

5.3.1.7.2 SGMII EEE Operation

When port is configured for SGMII operating mode the XL710 detects external PHY EEE support during SGMII Auto-negotiation when bit 9 of the received Auto-negotiation Link Control word is set to 1b (See Section 3.2.2.5.2 for additional information).

If the external PHY reports support of IEEE802.3az operation, the *PRTPM_EEER.TX_LPI_EN* bit is set to 1b and External PHY operating mode is either Full duplex 100BASE-TX or Full-duplex 1000BASE-T, the XL710 will transmit IEEE802.3az clause 36 /LI/ PCS ordered sets to initiate move of External PHY into EEE TX LPI state when conditions to enter EEE TX LPI state exist (See Section 5.3.1.1). When entering the EEE LPI state the XL710 powers down relevant circuitry to reduce power consumption.

If the external PHY reports support of IEEE802.3az operation, the External PHY operating mode is either Full duplex 100BASE-TX or Full-duplex 1000BASE-T, the XL710 will detect reception of IEEE802.3az clause 36 /LI/ PCS ordered sets and move internal circuitry into EEE RX LPI state.

Note: Even if external PHY does not report support of EEE during Auto-negotiation the XL710 can be forced to enable EEE operation by setting the corresponding bits in PRTMAC_PCS_AN_CONTROL4 register.

5.3.1.7.3 XAUI EEE Operation

When port is configured for XAUI operating mode and external 10GBASE-T PHY supports IEEE802.3az operation the corresponding bits in PRTMAC_PCS_AN_CONTROL4 register should be set to 1b to enable EEE operation on the XAUI interface (For further information on XAUI operating mode see Section 3.2.2.4.1).

If the external 10GBASE-T PHY connected to the XAUI interface supports IEEE802.3az EEE operation, the corresponding bits in PRTMAC_PCS_AN_CONTROL4 register are set to 1b and the *PRTPM_EEER.TX_LPI_EN* bit is set to 1b, the XL710 will transmit IEEE802.3az clause 48 ||LPIDLE|| PCS code groups to initiate move of External PHY into EEE TX LPI state when conditions to enter EEE TX LPI state exist (See Section 5.3.1.1). When entering the EEE LPI state the XL710 powers down relevant circuitry to reduce power consumption.

If the external10GBASE-T PHY connected to the XAUI interface supports IEEE802.3az EEE operation, the corresponding bits in PRTMAC_PCS_AN_CONTROL4 register are set to 1b, the XL710 will detect reception of IEEE802.3az clause 48 ||LPIDLE|| PCS code groups and move internal circuitry into EEE RX LPI state.

5.3.1.7.4 SFI EEE Operation

When port is configured for SFI operating mode and external 10GBASE-T PHY supports IEEE802.3az operation the corresponding bits in PRTMAC_PCS_AN_CONTROL4 register should be set to 1b to enable EEE operation on the SFI interface (For further information on SFI operation see Section 3.2.2.4.4).

If the external 10GBASE-T PHY connected to the SFI interface supports IEEE802.3az EEE operation, the corresponding bits in PRTMAC_PCS_AN_CONTROL4 register are set to 1b and the *PRTPM_EEER.TX_LPI_EN* bit is set to 1b, the XL710 will transmit IEEE802.3az clause 49 /LI/ PCS control characters to initiate move of External PHY into EEE TX LPI state when conditions to enter EEE TX LPI state exist (See Section 5.3.1.1). When entering the EEE LPI state the XL710 powers down relevant circuitry to reduce power consumption.

If the external 10GBASE-T PHY connected to the SFI interface supports IEEE802.3az EEE operation, the corresponding bits in PRTMAC_PCS_AN_CONTROL4 register are set to 1b, the XL710 will detect reception of IEEE802.3az clause 49 /LI/ PCS control characters and move internal circuitry into EEE RX LPI state.

5.3.2 Dx Low Power Backplane Auto-Negotiation (LPLU)

Backplane 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 XL710 is connected to an auxiliary power supply and main power is down, power saving may be more important than performance. The XL710 supports a mode where it Auto-negotiates to the Lowest Common Denominator (LCD) link speed. LCD Auto-negotiation is attempted in the following order:

1. 1000BASE-KX.
2. 10GBASE-KR or 10GBASE-KX4.
3. 40GBASE-KR4.

Initially in this mode the XL710 will attempt to negotiate to the lowest link rate. If Link partner doesn't support the lowest link rate the XL710 will Auto-negotiate to the lowest link rate advertised by the link partner.

Different behavior is defined for the D0 state and Dx state as a function of the per port *PRTPM_GC.LCDMP*, *PRTPM_GC.RATD* and *PRTPM_GC.EMPVETO* register bits.

Software driver can enable Dx Low Power Backplane Auto-Negotiation when operating in a single PF per port mode using the *Set Phy Config Command* (See Section 3.2.4.1.1) via the Admin queue. While in multiple PFs per port mode (MFP mode) if LPLU is enabled by default in the NVM a PF can only disable LPLU by sending a *Set Phy Config Command* (See Section 3.2.4.1.1) via the Admin queue.

The XL710 can initiate auto-negotiation without direct driver command in the following cases:

- When the state of MAIN_PWR_OK pin changes.
- When the PRTPM_GC.*MNG_VETO* bit value changes.
- When all functions on a transition from D0 (D0a or D0u) state to Dx state, or from Dx state to D0 state.

Note: MAIN_PWR_OK pin is de-asserted at least 10 ms before power drops below its 90% value. This allows enough time to the PHY to drop the link and restart auto-negotiation before auxiliary power takes over. Note that since auto-negotiation is restarted the PHY power consumption is already cut down.

Figure 5-4 and Figure 5-5 show the XL710 behavior when entering a low power state and during power state transitions.

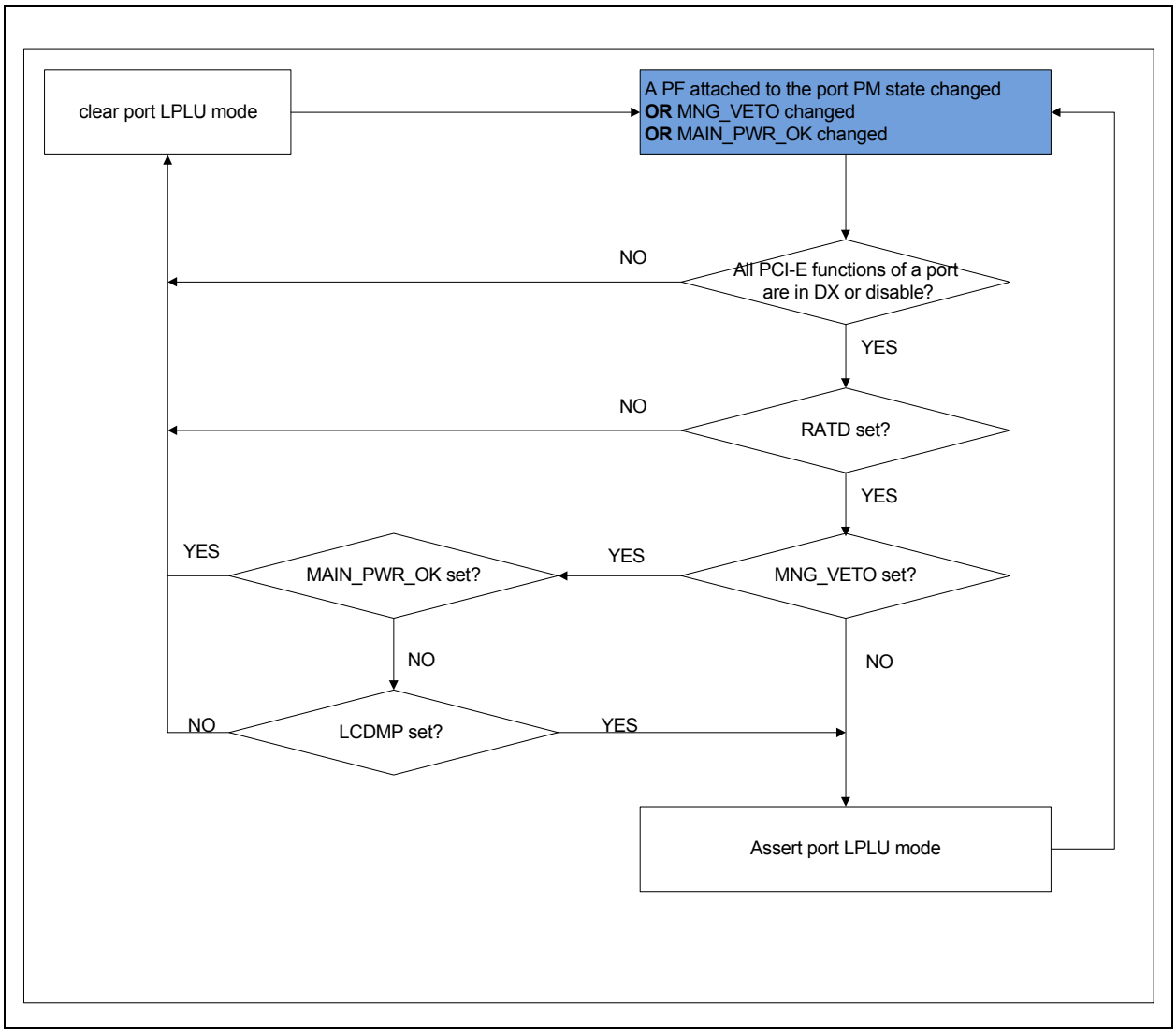


Figure 5-4. LPLU Assert/Clear When Entering Dx Power Down Mode

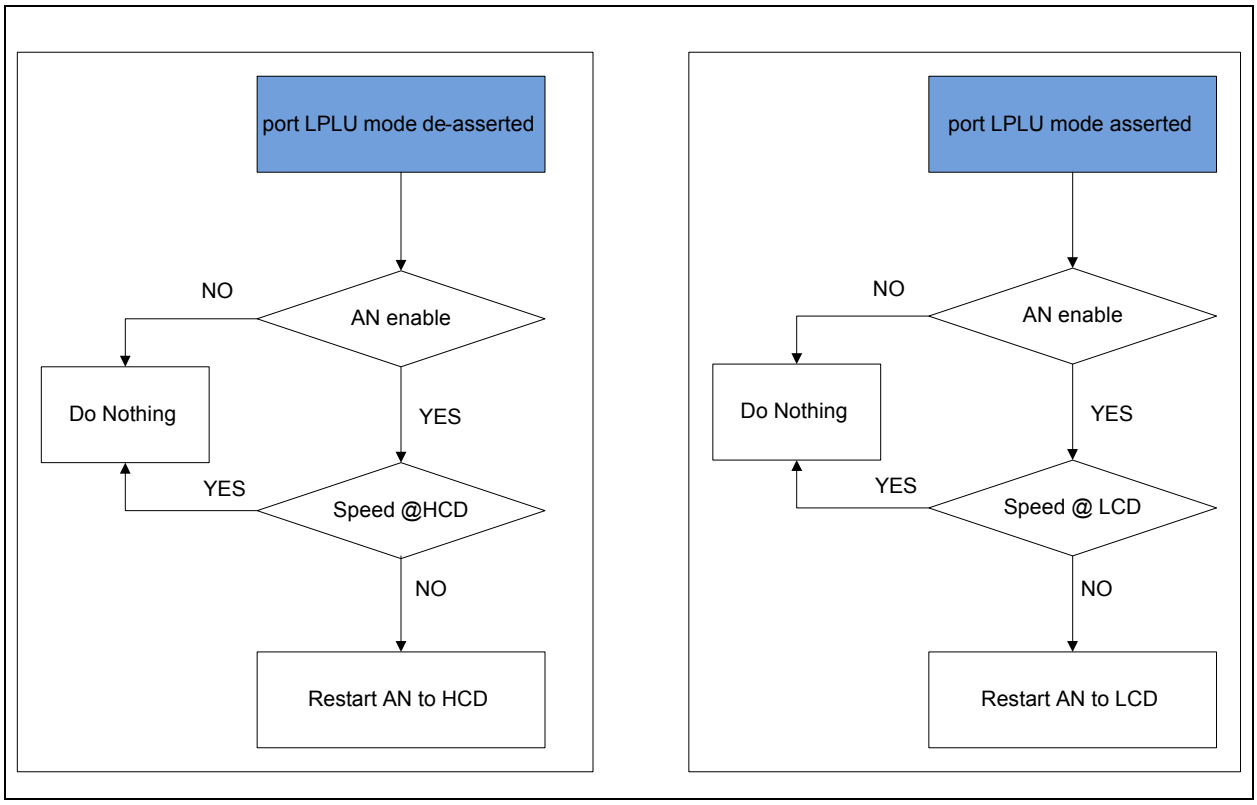


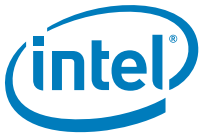
Figure 5-5. Link Speed Change on LPLU Assert/Clear

5.4 Wake Up

The XL710 supports:

1. Advanced Power Management (APM) wake up, which means support for Wake from S5 (Soft Off) states using the well-known Magic Packet wake packet format.
2. Link State Change (LNKC) wake up, which means the ability to wake up the host from S5 when a LAN link state goes up.
3. Firmware Reset (FW_RST_WK) wake up, which means the ability to wake up the host from S5 when an EMPR event occurs.
4. 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 actually defined for this option.

These wake up mechanisms are basically controlled by the PF via the corresponding bit in PFPM_WUFC and PFPM_WUS registers. It suffices that one PF enables a wake up mechanism for the wake up to be enabled for the concerned port.



5.4.1 Advanced Power Management Wake Up

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 XL710 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 and wake-up status is reported in the per PF *Wake Up Status* (*PFPM_WUS*) register. At power up, the XL710 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 5.4.5.1 and Section 5.4.5.2 respectively.

When APM wake up is enabled, the XL710 checks all incoming packets passing L2 filters (BCST, MCST, UCST) for Magic Packets. See Section 5.4.2.1.1 for a definition of Magic Packets. BCST and UCST (according to PFPM_SAL/H that are loaded from NVM and properly assigned by FW 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 needs to be set to 1b.

Once XL710 receives a matching Magic packet, and the *PFPM_APM.APME* bit is set to 1b it executes the flow described in Section 5.4.5.3.

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 enable disabling of system Wake-up when *PMCSR.PME_En* is cleared, Software driver should clear the *PFPM_APM.APME* bit after power-up or PCIe reset.

5.4.2 Wake-Up Filters

The XL710 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.

5.4.2.1 Wake Up Filter Types

The following packet types are detected per PF by the XL710 as a possible Wake up.

5.4.2.1.1 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 may 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 XL710 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 FW to the relevant filters PRTPM_SAL/H.

On power up the FW 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 [15:0] (PF_NUM) to be reported if a magic packet matches this filter

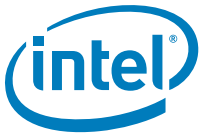
The FW uses the PFGEN_PORTNUM registers to map the per PF MAC address to the relevant ports.

The XL710 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 don't 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 that 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, XL710 starts up with the Ethernet MAC Destination address loaded from the NVM. This enables XL710 to accept packets with the matching IEEE address before the Software device driver loads.

Table 5-2. 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 S-TAG		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.



5.4.3 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.

5.4.4 Wake-Up and Manageability

A packet that passes pre-defined WoL Filter should be checked also against a match in pre-defined manageability filter if the device is in S5 (PRST# is asserted) and the GLPM_WUMC.NOTCO bit is cleared.

When conditions above are met (S5 and GLPM_WUMC.NOTCO bit is cleared) and there is a match in one of the pre-defined manageability filters, the packet will not wake the system if the filter is management only (defined by PRT_MNG_MNGONLY).

EMP has the ability to initiate a wake up event by setting the relevant GLPM_WUMC.MNG_WU_PF bit to initiate a wake up event targeted at a specific PF.

Packets are not inspected for the WoL filter when there is a function in D0 state and the NOTCO bit is cleared (like an MC is present).

5.4.5 Wake-Up Flows

5.4.5.1 Wake-Up Enable Flow

A WoL register set exists in each PF. Each set consists of the following Wake-up filters and registers:

1. *PFPM_WUC* - Wake Up Control Register.
2. *PFPM_WUS* - Wake Up Status Register.

On Power-up, values loaded from the NVM into the following per PF registers define enablement of APM wake-up functionality:

1. The NVM *APM Enable* bit is loaded to the *PFPM_APM.APME* register bit to enable:
   a. Detection of a Magic packet destined to the PF.
   b. Setting of the *PFPM_WUS.MAG* bit on detection of a magic packet destined to the PF.
2. The *EN_APM_D0* bit is loaded from the NVM into the *PFPM_WUC.EN_APM_D0* register bit to enable generating a wake event even when the PF is not in a D3 state and APM is enabled when and a Magic packet is sent to the PF.

5.4.5.2 Wake-Up Disable Flow

Once the XL710 wakes the system following transition to D0 the Software driver may 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.

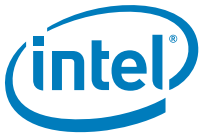
5.4.5.3 Wake-up Packet Flow

Once wake up is enabled, the XL710 monitors incoming packets, if a packet passes at least one of the enabled wake-up filters, the XL710:

- Sets the *PME_Status* bit in the PFPM_WUS and PCI *PMCSR* register.
- If the *PME_En* bit in the *PMCSR* configuration register is set, asserts PE_WAKE_N and/or sends a PM_PME PCIe message.
- Sets one or more of the bits in the *PFPM_WUS* register. The XL710 sets more than one bit if a packet matches more than one filter.

Note: If enabled, a link state change wake-up also causes similar results. Sets the *PFPM_WUS.PME_Status* and the *PMCSR.PME_Status* bit, asserts the PE_WAKE_N signal, and/or sends a PM_PME PCIe message and sets the relevant bit in the *PFPM_WUS* register.

The PE_WAKE_N signal remains asserted, PM_PME Messages are periodically sent to the host, and the XL710 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.



6.0 Non-volatile memory map

6.1 NVM organization

6.1.1 Hierarchical NVM map

Table 6-1 lists in a hierarchical order of all the modules and sub-modules that are defined in the XL710's NVM map. They are referred to separately as NVM sections. All details concerning these sections are found in Section 6.2.

Note that the root section named Init Module (NVM header) is not listed in Table 6-1. Details about this section can be found in Section 6.1.2.

Table 6-1. Hierarchical NVM map

Address of Pointer in Parent Section	1st Level Section	2nd Level Section	3rd Level Section	Description
0x06	RO PCIR Registers Auto-load			Contains RO parameters that configure the PCIe transaction layer (note that it is currently empty).
0x0A	RO PCIe LCB			Contains RO parameters that configure the PCIe link layer (LCB unit)
0x37	Software Alternate MAC Address			Original factory default MAC addresses defined for LAN.
0x28	Permanent SAN MAC Address			MAC addresses used for SAN.
0x16	PBA Block			The PBA block contains the complete PBA number including the dash and the first digit of the 3-digit suffix.
0x30	PXE Setup Options			Setup options for the PXE driver that is defined per PCIe function.
0x31	PXE Configuration Customization Options			Configuration customization options for the PXE driver that is defined per PCIe function.

Table 6-1. Hierarchical NVM map

Address of Pointer in Parent Section	1st Level Section	2nd Level Section	3rd Level Section	Description
0x2F	VPD Module			Vital Product Data (VPD) loaded by an OEM. It contains RO and RW information about a NIC/LOM.
0x17	Boot Configuration Block			Contains the required setup used for boot operations.
0x08	PCIR Registers Auto-load			Default setup to registers and internal memories that load on PCIR events.
0x38	POR Registers Auto-load			Default setup to registers that load on POR events.
0x3C	CORER Registers Auto-load			Default setup to registers and internal memories that load on CORER events.
0x3B	GLOBR Registers Auto-load			Default setup to registers that load on GLOBR events.
0x4A	Core Mem Config			Read margin settings to the device's core memories. RO module.
0x48	EMP SR Settings			This section contains the modes of operation of the EMP that must be stored in the shadow RAM.
	0x05	SR PF Allocations		This section contains the allocation of resources per PF that must be stored in shadow RAM.
	0x0F	MNG MAC Addresses		This section contains the Pass Through LAN Ethernet MAC addresses, 4 addresses per port.
	0x10	PF MAC Addresses		This section contains the PF Ethernet MAC addresses, one address per PF.
0x07	Auto Generated Pointers			Pointers to type 1/2 words used by EMP and software. The module must be mapped to shadow RAM word address 0x7D80.
0x3E	PCIe ALT Auto-load			Modified factory default settings to registers that load on PCI reset events. The module must be mapped to shadow RAM word address 0x7E00. Any change performed to the module contents in shadow RAM is not dumped into the Flash memory.
0x03	PCIe Analog			Configures the analog section of the PCIe PHY. Module is authenticated.
0x04	PHY Analog			Configure the analog section of the SerDes PHY. Module is authenticated.

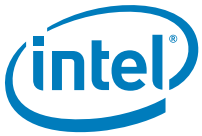


Table 6-1. Hierarchical NVM map

Address of Pointer in Parent Section	1st Level Section	2nd Level Section	3rd Level Section	Description
0x09	EMP Global			This section contains two sub-sections: 1. Vendor Specific Settings: Settings for external PHY or modules. This part has a proprietary format in order to enable advanced PHY setting. 2. List of Qualified Modules: Parameter list of up to 16 modules. Per-module list holds OUI, revision and version numbers. These settings are loaded into the external devices via the MDIO interface.
0x0E	Manageability Module Header			This section contains parameters related to the manageability functionality such as connection type and others. It also points to sub-sections configuring the filters and the sideband interfaces.
	0x03	Pass Through Control Words Structure 0		This section contains the initial setting of the pass-through filters for a port. This section is used only in SMBus legacy pass-through mode. This section is repeated per port.
	0x04	Pass Through Control Words Structure 1		
	0x05	Pass Through Control Words Structure 2		
	0x06	Pass Through Control Words Structure 3		
	0x08	Flexible TCO Filter Configuration Structure		This section contains the setting of the manageability flex filters for all ports.
	0x07	Sideband Configuration Structure		This section describes the setting of the different pass-through interfaces (SMBus, NC-SI and MCTP).
	0x0B	OEM Section		This section is the header of the OEM specific data identifying the OEM for which this data is defined.
		0x02	Partition Information Structure	
		0x01	Extended Capabilities Structure	
		0x00	Inventory Parameters Structure	
0x0F	EMP Settings			This section contains the modes of operation of the EMP.
	0x05	PF Allocations		This section contains the allocation of resources per PF and the PF MAC address.

Table 6-1. Hierarchical NVM map

Address of Pointer in Parent Section	1st Level Section	2nd Level Section	3rd Level Section	Description
	0x07	PHY Capability Data Structure 0		The PHY capabilities data structure contains all the parameters to control the internal and external PHY operation.For example, interface type and mode, EEE parameters, etc. Data structure is repeated per port.
	0x08	PHY Capability Data Structure 1		
	0x09	PHY Capability Data Structure 2		
	0x0A	PHY Capability Data Structure 3		
	0x0B	External-to-Internal PHY Mapping 0		This section provides the internal PHY mode that is selected for each external PHY/module counter part. The XL710 provides multiple PHY interface for connecting different types of external PHYs and optical or copper modules. End users can plug in different types of modules into the SFP+ or QSFP+ connectors and choose from several connectivity options when using external PHYs. For example a 10GBASE-T PHY could use a XAUI or KR connection to connect to the XL710. Firmware has to read the PHY or module ID and choose the appropriate interface to operate with that external module. Further, when working with an external 10GBASE-T PHY, auto-negotiation might result in several speed modes.For example, when a 10GBASE-T PHY can auto-negotiate to 1 GbE. In this case, firmware needs to reduce the internal link to SGMII. This section is repeated per port.
	0x0C	External-to-Internal PHY Mapping 1		
	0x0D	External-to-Internal PHY Mapping 2		
	0x0E	External-to-Internal PHY Mapping 3		
	0x0F	LLDP Configuration		Default settings to the embedded LLDP agent.
	0x14	LLDP OEM TLVs		A set of fixed-structure LLDP TLVs for EMP use.
0x46	2nd Free Provisioning Area			Free area used as the new bank when updating 8 KB long modules that are mapped outside the shadow RAM.
0x05	Expansion/Option ROM (OROM)			It contains pre-boot code and settings read by BIOS. Pre-boot code is authenticated by BIOS during initialization before being executed by an EMP on update.
0x0B	EMP Image			It contains the EMP processor code.
0x40	1st Free Provisioning Area			Free area used as the new bank when updating 1160 KB long modules.

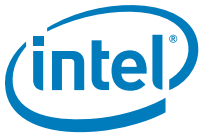


Table 6-1. Hierarchical NVM map

Address of Pointer in Parent Section	1st Level Section	2nd Level Section	3rd Level Section	Description
0x42	Reserved 4th Free Provisioning Area			Free area to be used as the new bank when updating future 64 KB long modules mapped outside the shadow RAM.
0x44	Reserved 3rd Free Provisioning Area			Free area to be used as the new bank when updating future 16 KB long modules mapped outside the shadow RAM.
0x4B	FCoE Scratch Pad Area			Scratch pad area used by FCoE drivers to store for persistent storage of FIP discovery data as well as other potential future debug and/or diagnostic purposes. It uses 4 KB per port.

6.1.2 NVM header

Table 6-2 lists the format of the NVM header section. It includes words with a specific content and pointers to the first level of NVM modules.See section 6.2 for more details. There is no section 6.2.

- **Bold** items are pointers to modules.
- The Auth / RO column indicates the following:
  - Auth - The pointed module is authenticated and the pointer is read-only
  - RO - The pointed module and/or the pointer are read-only, or the word is read-only
  - No - The word or the module is read/write

  A read-only item can be written only when either in the blank Flash programming mode or through the flow described in Section 3.3.6.
- 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.
- Maximum Provisioned Size indicates the following:
  - Modules mapped in shadow RAM - How much has been accounted for the module in the shadow RAM, though the module can exceed this limit as long as the entire shadow RAM contents is not exceeding its size.
  - Modules mapped outside shadow RAM - This is the maximum size the module can take because it has to fit within the associated free provisioning area.
  - The maximum size provisioned for the NVM header itself is 256 words.

Table 6-2. NVM header map

Word Address	Pointed Module Name / Word Name	Auth / RO	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM	Typical Size	Maximum Provisioned Size
0x00	NVM Control Word 1	RO	HWEMP	POR				
0x01	RO Commands Version	RO	EMP	EMPR				
0x02	Reserved			[1]				
0x03	PCIe Analog	Auth	HW	POR	No	No	6 KB	8 KB
0x04	PHY Analog	Auth	HW	POR	No	No	6 KB	8 KB
0x05	Expansion/Option ROM	Auth	BIOS/HW	PCIR[2]	No	No	496 KB	1160 KB
0x06	RO PCIR Registers Auto-load	RO	HWEMP	PCIR	Yes	Yes	64 bytes	128 bytes
0x07	Auto Generated Pointers	No	EMP/SW	POR	Yes	Yes	16 bytes	64 bytes
0x08	PCIR Registers Auto-load	No	HWEMP	PCIR	Yes	Yes	512 bytes	1024 bytes
0x09	EMP Global	RO[3]	EMP	GLOBR	No	No	512 bytes	8 KB
0x0A	RO PCIe LCB	RO	HW	PERST	Yes	Yes	768 bytes	1024 bytes
0x0B	EMP Image	Auth	EMP	EMPR	No	No	816 KB	1160 KB
0x0C	Reserved							
0x0D	Reserved							
0x0E	Manageability	RO[4]	EMP	EMPR	No	No	1 KB	8 KB
0x0F	EMP Settings	RO[4]	EMP	CORER[4]	No	No	1 KB	8 KB
0x10	SW Compatibility Word 1	No	SW/BIOS	POR[2]				
0x11	SW Compatibility Word 2	No	SW/BIOS	POR[2]				
0x12	SW Compatibility Word 3	No	SW/BIOS	POR[2]				
0x13	SW Compatibility Word 4	No	SW/BIOS	POR[2]				
0x14	SW Compatibility Word 5	No	SW/BIOS	POR[2]				
0x15	PBA Flag	No	SW/BIOS	POR[2]				
0x16	PBA Block	No	SW/BIOS	POR[5]	No	Yes	12 bytes	12 bytes
0x17	Boot Configuration	No	BIOS	POR[7]	No	Yes	3072 bytes	3072 bytes
0x18	NVM Image Revision	No	SW/BIOS/ EMP	POR				
0x19	SW Reserved Word 2	No	SW/BIOS	POR[2]				
0x1A	SW Reserved Word 3	No	SW/BIOS	POR[2]				
0x1B	SW Reserved Word 4	No	SW/BIOS	POR[2]				
0x1C	SW Reserved Word 5	No	SW/BIOS	POR[2]				
0x1D	SW Reserved Word 6	No	SW/BIOS	POR[2]				
0x1E	SW Reserved Word 7	No	SW/BIOS	POR[2]				
0x1F	SW Reserved Word 8	No	SW/BIOS	POR[2]				
0x20	SW Reserved Word 9	No	SW/BIOS	POR[2]				
0x21	SW Reserved Word 10	No	SW/BIOS	POR[2]				
0x22	SW Reserved Word 11	No	SW/BIOS	POR[2]				
0x23	SW Reserved Word 12	No	SW/BIOS	POR[2]				
0x24	SW Reserved Word 13	No	SW/BIOS	POR[2]				

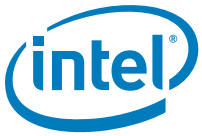


Table 6-2. NVM header map

Word Address	Pointed Module Name / Word Name	Auth / RO	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM	Typical Size	Maximum Provisioned Size
0x25	SW Reserved Word 14	No	SW/BIOS	POR[2]				
0x26	SW Reserved Word 15	No	SW/BIOS	POR[2]				
0x27	SW Reserved Word 16 -	No	SW/BIOS	POR				
0x28	SW Reserved Word 17 - Permanent SAN MAC Address	No	SW/BIOS/ EMP	CORER	No	Yes	98 bytes	128 bytes
0x29	SW Reserved Word 18	No	SW/BIOS	POR[2]				
0x2A	SW Reserved Word 19	No	SW/BIOS	POR[2]				
0x2B	SW Reserved Word 20	No	SW/BIOS	POR[2]				
0x2C	SW Reserved Word 21	No	SW/BIOS	POR[2]				
0x2D	SW Reserved Word 22	No	SW/BIOS	POR[2]				
0x2E	SW Reserved Word 23	No	SW/BIOS	POR[2]				
0x2F	VPD Area	RO[6]	EMP/VPD SW	POR[7]	No	Yes	256 bytes	1024 bytes
0x30	PXE Setup Options	No	BIOS	POR[7]	No	Yes	34 bytes	64 bytes
0x31	PXE Configuration Customization Options	No	BIOS	POR[7]	No	Yes	34 bytes	64 bytes
0x32	PXE Version	No	BIOS	POR[2]				
0x33	IBA Capabilities	No	BIOS	POR[2]				
0x34	SW Reserved Word 24	No	BIOS	POR[2]				
0x35	SW Reserved Word 25	No	BIOS	POR[2]				
0x36	iSCSI Option ROM Version	No	BIOS	POR[2]				
0x37	Software Alternate MAC Address	No	SW/BIOS/ BMC	POR[7]	No	Yes	218 bytes	256 bytes
0x38	POR Registers Auto-load	No	HW	POR	Yes	Yes	1536 bytes	2048 bytes
0x39	Reserved							
0x3A	Reserved for EMPR Registers Auto-load	RO	HW	POR	Yes	Yes	0 bytes	32 v
0x3B	GLOBR Registers Auto-load	No	HW	GLOBR	Yes	Yes	800 bytes	1024 bytes
0x3C	CORER Registers Auto-load	No	HW	CORER	Yes	Yes	41300 bytes	45056 bytes
0x3D	Reserved							
0x3E	PCIe ALT Auto-load	RO[7]	HW/EMP	PCIR[2]	Yes	Yes	512 bytes	1KB
0x3F	SW Checksum	No	SW					
0x40	1st Free Provisioning Area	RO[4]	EMP	EMPR[2,3]	No	No	1160 KB	1160 KB
0x41	1st Free Provisioning Area Size	RO	EMP	EMPR				
0x42	Reserved 4th Free Provisioning Area	RO[4]	EMP	EMPR[2,3]	No	No	64 KB	64 KB
0x43	Reserved 4th Free Provisioning Area Size	RO	EMP	EMPR				
0x44	Reserved 3rd Free Provisioning Area	RO[4]	EMP	EMPR[2,3]	No	No	16 KB	16 KB
0x45	Reserved 3rd Free Provisioning Area Size	RO	EMP	EMPR				

Table 6-2. NVM header map

Word Address	Pointed Module Name / Word Name	Auth / RO	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM	Typical Size	Maximum Provisioned Size
0x46	2nd Free Provisioning Area	RO[4]	EMP	EMPR[2,3]	No	No	8 KB	8 KB
0x47	2nd Free Provisioning Area Size	RO	EMP	EMPR				
0x48	EMP SR Settings	No	EMP	EMPR[8]		Yes	82 bytes	128 bytes
0x49	Reserved	No	EMP	EMPR				
0x4A	Core Mem Config	RO	HW	POR	Yes	Yes	1 KB	1 KB
0x4B	FCoE Scratch Pad Area	RW	SW	POR[3]	No	No	16 KB	16 KB
0x4C	FCoE Scratch Pad Area Size	RW	SW	POR				
0x4D - 0xFF	Spare NVM Header Words	No	HW	POR[2]				

1. The word/module is loaded only into the shadow RAM.
2. Only the pointer is loaded not the module's contents.
3. RO pointer, while the pointed area is RW.
4. Some items load on CORER, others on EMPR.
5. Pointer and module contents are loaded only into shadow RAM.
6. VPD pointer is a RO pointer, regardless to its own value and to the value of the GLPCI_CAPCTRL.VPD_EN bit. If VPD Write Enable bit is cleared in NVM, VPD area cannot be modified via the NVM Update admin command, but only via the VPD register set in the PF config space.
7. The module must be mapped to the shadow RAM word address 0x7E00. Any change performed to the module contents in shadow RAM is not dumped into Flash memory. Software cannot directly modify the module contents via NVM update commands, but only indirectly via the Alternate Structure admin commands.
8. Pointer and module contents.

6.1.3 Structure of hardware modules

An NVM module read by 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 XL710. In Table 6-2, these modules are referred as CSR format = No modules.
- Flexible functionality module — The structure defines the device address or addresses into which the NVM words are loaded. Therefore, the module might be used for different purposes based on its contents. In Table 6-2, these modules are referred as CSR format = Yes modules.

The remainder of this section describes the structure of the contents of a flexible functionality module (excluding the module header described in Section 6.1.5). The general structure of the module is shown in Figure 6-1.

A CSR format module can be made of a mix of repeating Type 1, 2, 3 and 4 segments. Each segment is described by its own header (the shaded fields in Figure 6-1).

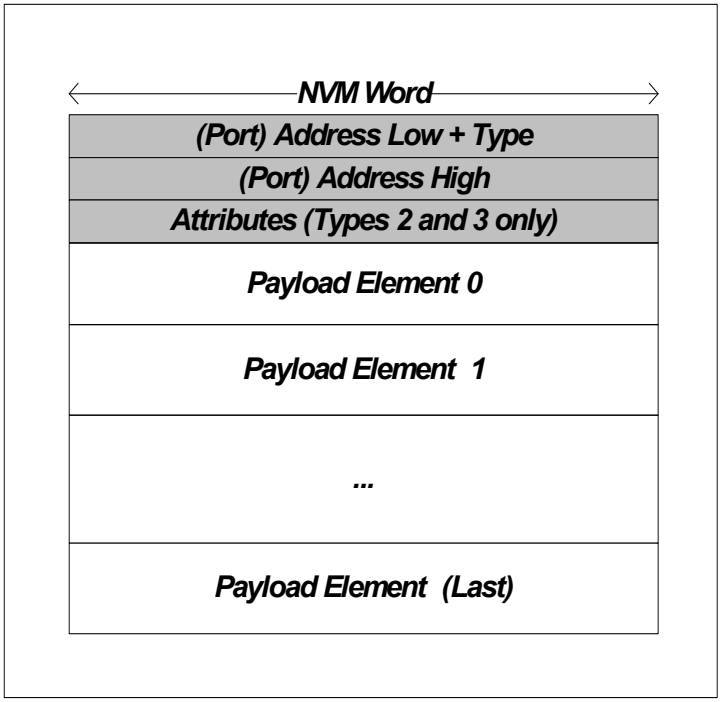


Figure 6-1. Structure of a flexible functionality module

Description of the fields in Figure 6-1:

- **(Port) Address** — A 28-bit value that defines the starting address to which data Dwords are loaded. Serves as either a direct address to load into (Types 1 & 2) or as an indirect address referred to as a port address (Type 3 & 4). A port is made of a 32-bit address register followed by data registers of total size width. Address low is a 12-bit field mapped to word[15:4].
- **Type** — A 4-bit command field mapped to word[3:0] bits that defines the type of flexible module. Descriptions are as follows:
  - — Type 1 (0001b) — Loads a single 32-bit data segment
  - — Type 2 (0010b) — Load a set of 32-bit data Dwords into consecutive addresses
  - — Type 3 (0011b) — Loads a set of 32-bit data Dwords through an address port
  - — Type 4 (0100b) — Loads, through an address port, a sequence of data blocks into arbitrary addresses
- **Attributes** — A 16-bit optional word that defines attributes of the module
  - — **Width** — A 3-bit field mapped to word[2:0] bits that describes the width (in 32-bit Dwords) of a data element
    - 000b = Data is 32-bit wide
    - 001b = Data is 64-bit wide
    - 010b = Data is 128-bit wide
    - 011b = Data is 256-bit wide

  - Else = Reserved
  - **Skip** — A 2-bit field mapped to word[4:3] bits that describes the number of address bytes to be skipped for getting the address of the next payload element (Type 2) or port register involved (Type 3 or 4).
    - 00b = Skip 4 bytes
    - 01b = Skip 8 x 4 bytes = 32 bytes
    - 10b = Skip 32 x 4 bytes = 128 bytes
    - 11b = Reserved
  - **Length** — An 11-bit field mapped to word[15:5] bits that contains the number of data elements, each of Width bits
- **Payload Elements** — A sequence of address or data elements to be loaded into the XL710. 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.

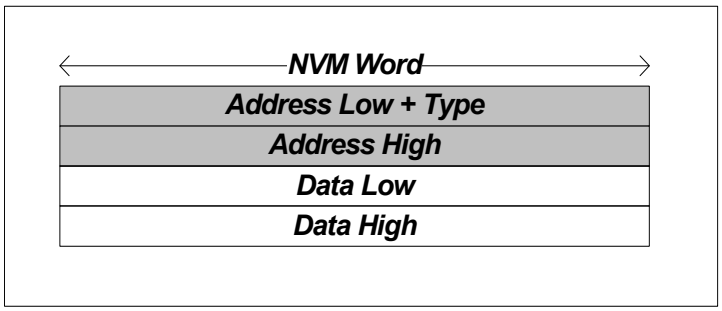


Figure 6-2. Structure of a type 1 module

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

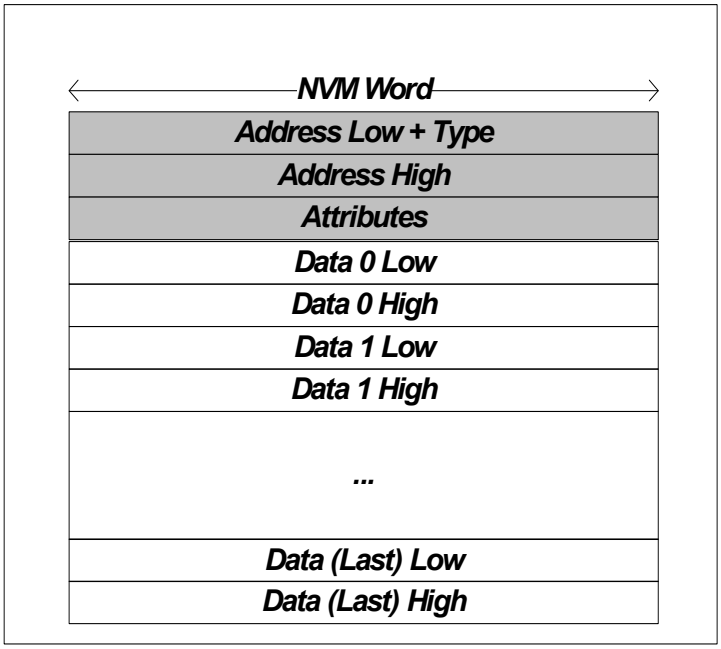


Figure 6-3. Structure of a type 2 module

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 incremented (by the XL710) with each write of a new data element from the list. Consecutive data elements are written into the port.
- The *Data Elements* field are made of length elements, each of the width DWords. Each port load cycle handles one data element

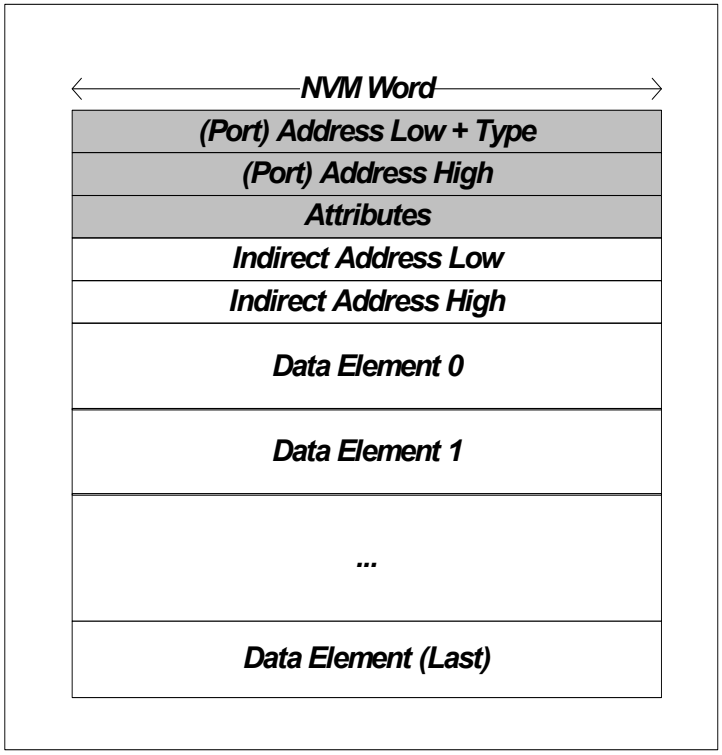


Figure 6-4. Structure of a type 3 Module

The following figures describe two cases of using a Type 3 module:

- Figure 6-5 shows a case of loading 32-bit values into a Global (GL) port
- Figure 6-6 shows a case of loading 64-bit values into a PRT port

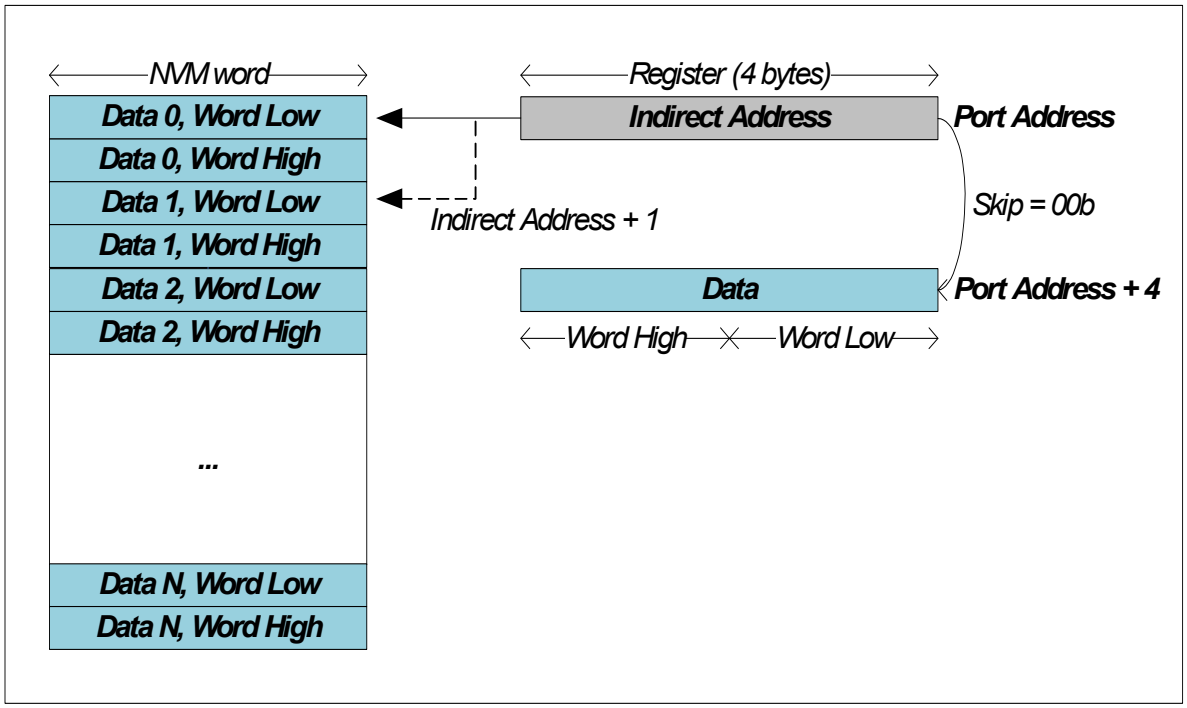


Figure 6-5. Access through a type 3 module of 32-bits wide data elements

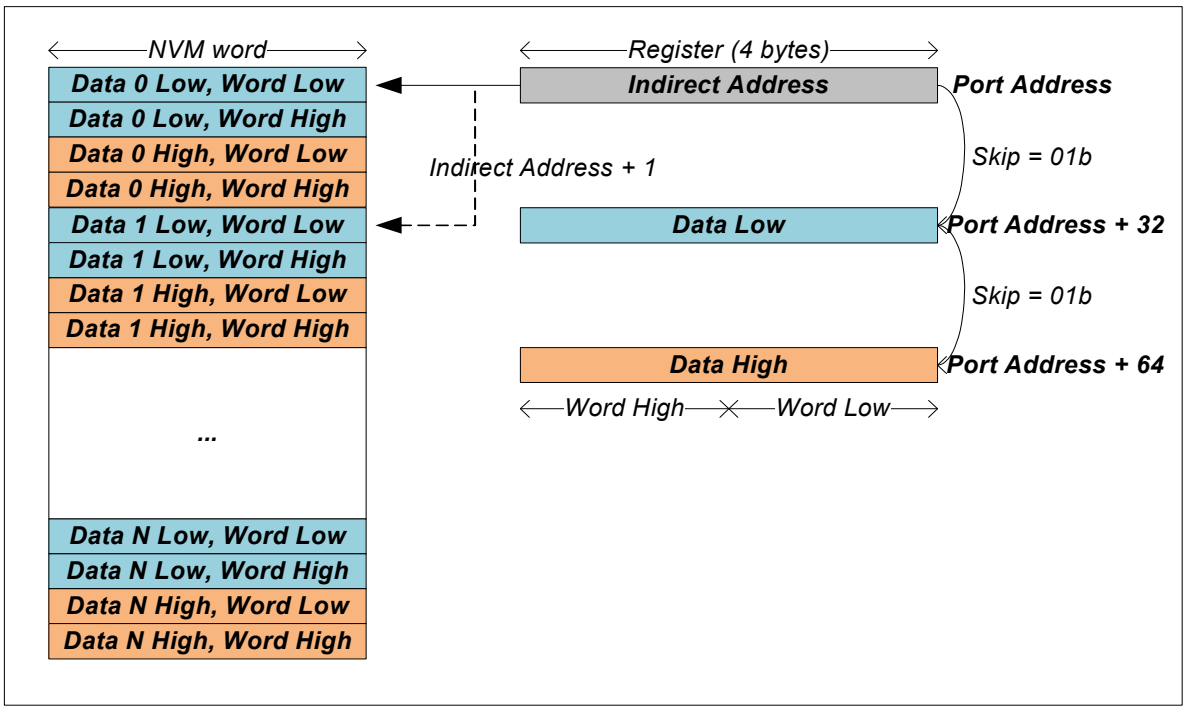


Figure 6-6. Access through a type 3 module of 64-bits wide per port data elements

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 grey 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).

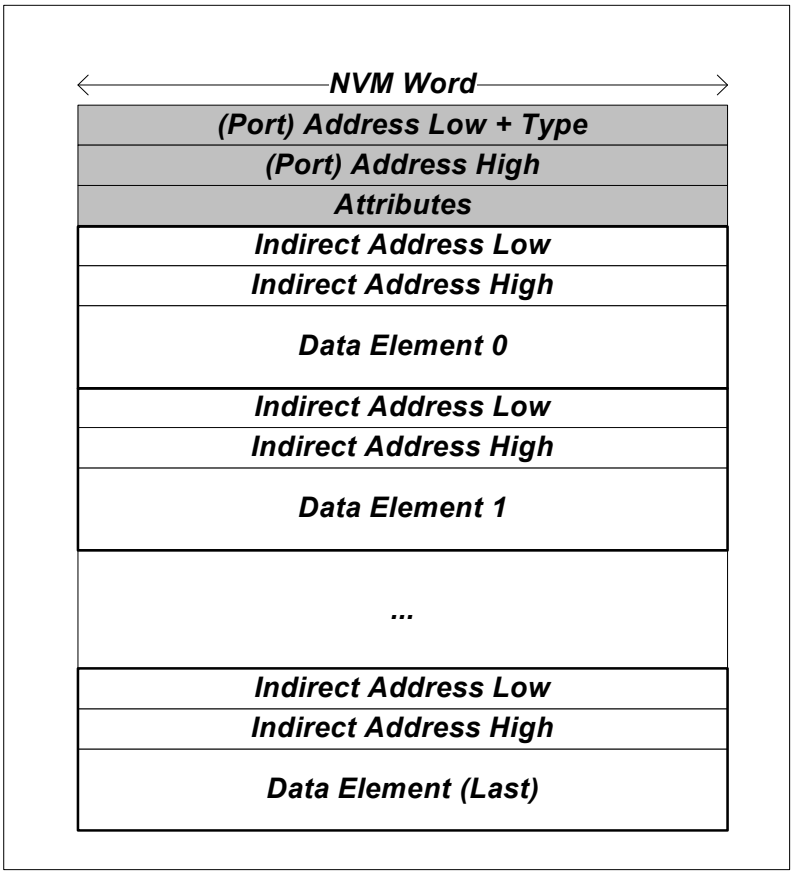


Figure 6-7. Structure of a type 4 module

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

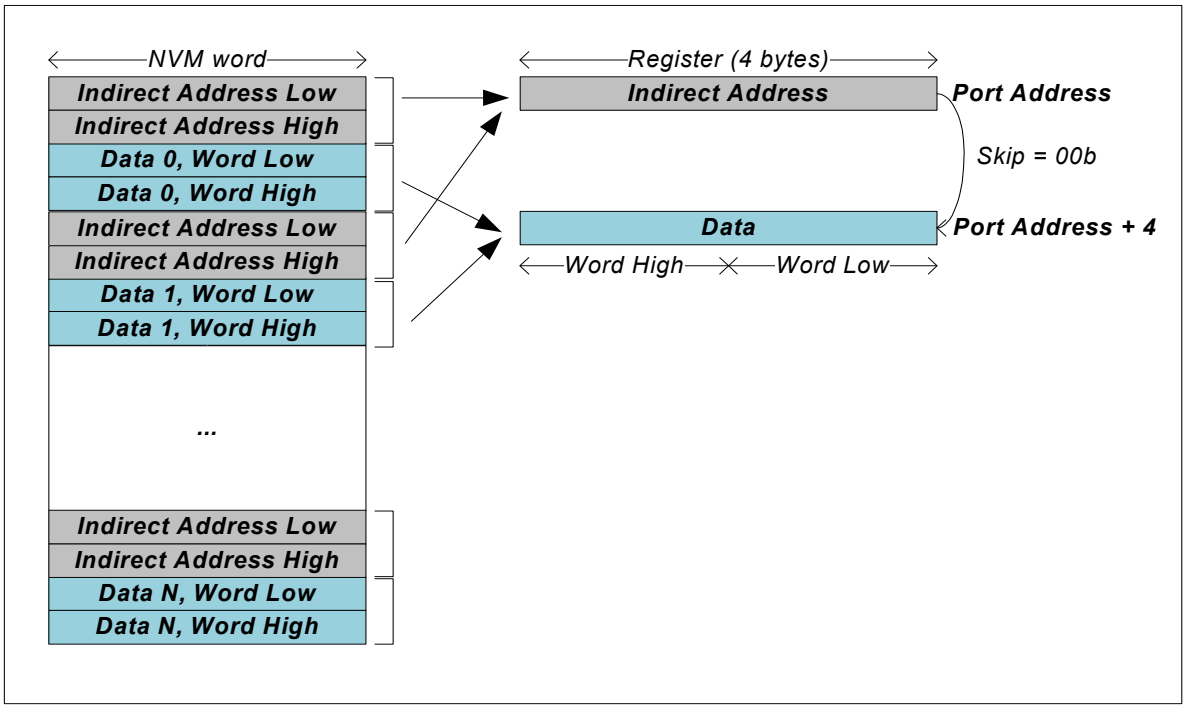


Figure 6-8. Access through a type 4 module of 32-bits wide data elements

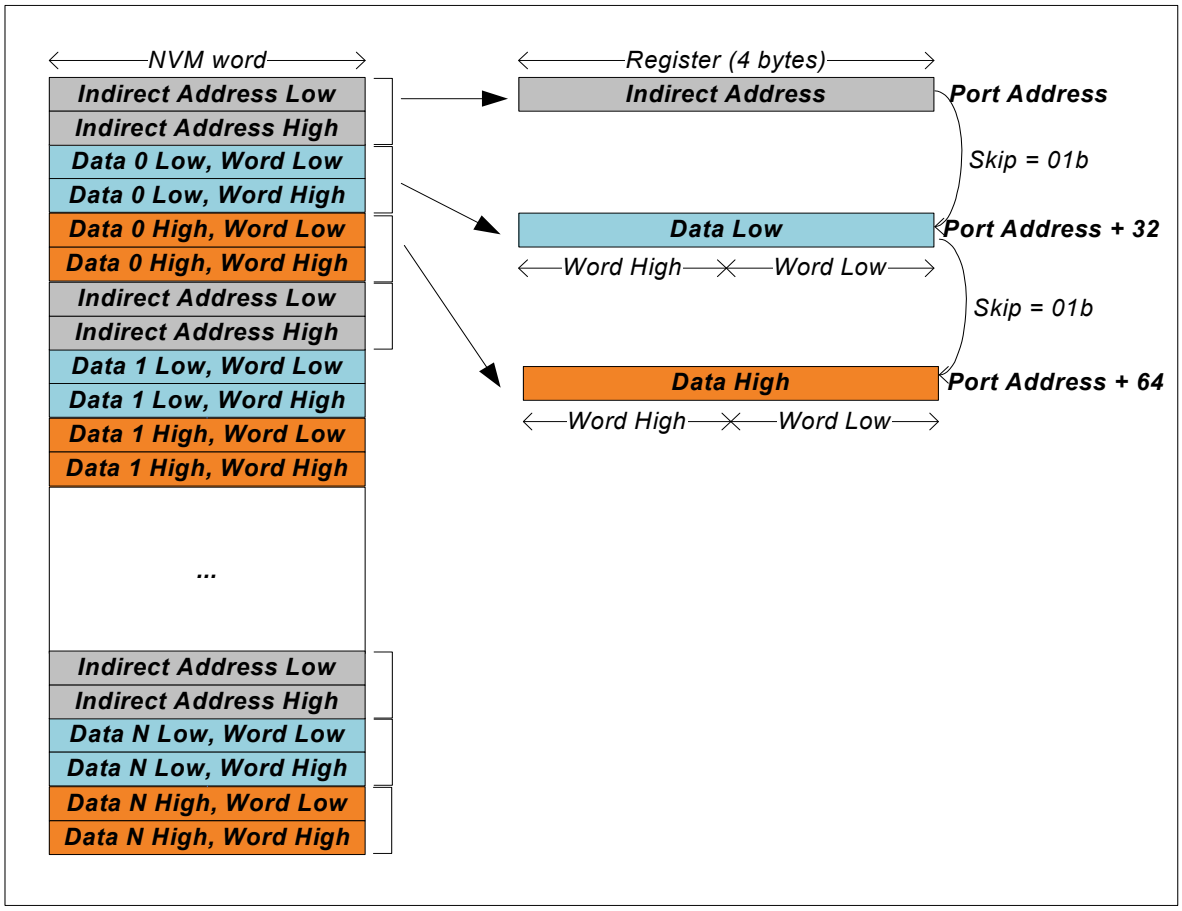


Figure 6-9. Access through a type 4 module of 64-bits wide per port data elements

6.1.3.5 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. Until the list of Type 1/2 items stabilizes, each invocation of the tool might lead to a different mapping offset in the destination module.

EMP code and software tools cannot accommodate these variations in mapping offset, and hence, they need a fixed method for accessing some predefined Type 1/2 items. The auto-generated pointers module is automatically defined by the auto map generation tool for this purpose. It is pointed to by NVM word 0x07. It lists the mapping address in NVM of the pre-defined Type 1/2 items via a 2-word structure per item.

The word address in shadow RAM of an item is given by the sum of the contents of its two associated words: pointer + offset. If the item is relative to an array of registers and/or to a register with a scope different than global, the offset is given for the first data word of the Type 1/2 array, for array instance [0] and scope instance [0].

In the auto-generated pointers module, the location of the 2-word structures relative to an item is made invariant along the project's life. The alias name of the Type 1/2 register appears in the names of the two words in the structure.

Following is the list of auto-generated pointers, listed in the order by which they appear in the module:

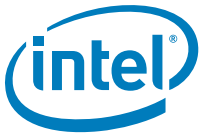
1. PFPM_APM
2. PRTPM_GC
3. GLGEN_STAT
4. GLPCI_SERL
5. GLPCI_SERH
6. PRTGL_SAL
7. PRTGL_SAH
8. GLPCI_CAPSUP
9. PRTDCB_MFLCN
10. PRTDCB_FCCFG
11. PFGEN_PORTNUM
12. PFPCI_FUNC2
13. PFPCI_CLASS
14. Reserved
15. PF_VT_PFALLOC
16. GLGEN_PCIFCNCNT
17. GLPCI_REVID
18. PFPCI_DEVID
19. GLPCI_SUBVENID
20. PFPCI_SUBSYSID
21. GLPCI_VENDORID
22. GLPCI_CNF2

6.1.4 NVM integrity checks by software

This section describes the NVM integrity fields inserted in the XL710 NVM map to provide a basic detection of a faulty Flash part.

A software checksum check is performed by the PF driver just after completion of its initialization sequence. It covers only modules 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 XL710.

Modules mapped outside the shadow RAM include a CRC8 field. EMP is responsible to check the CRC8 of the modules it loads/updates from/to Flash. It also provides software tools with the ability to check the integrity of these modules upon request because the PF driver does not include this check in its initialization sequence.



6.1.4.1 Software checksum

The *Software Checksum* field covers the entire 64 KB shadow RAM contents (reserved words included) with the exception of the VPD area and to the PCIe ALT auto-Load module, which are skipped.

The *Software Checksum* word is located at word 0x3F. Its value 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, it must update 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 must update the *Software Checksum* field accordingly.

6.1.4.2 CRC8

The PCIe and PHY analog modules, and the EMP image have a separate CRC8 field embedded in their header. Refer to the Module Format Version + CRC8 field in Table 6-5 for its computing rules and coverage.

The CRC polynomial used is: X^8+X^2+X+1.

Similarly, the following modules include CRC8 fields in their header and in the headers of their sub-modules:

- EMP global module
- Manageability module
- EMP settings module

EMP is responsible to check CRC8 validity of these modules (and their included sub-modules) on every EMPR. If CRC8 is invalid after two tries (every try takes 260 ms for a 800 KB long EMP image), EMP must report the error in the GL_MNG_FWSM.EXT_ERR_IND and CRC_ERROR_MODULE register fields by posting to the MC an AEN # 0x82 - NVM error (enable bit = 18) over NC-SI and by using the Status Data Byte 2[5] in the SMBus alert.

If the CRC8 error occurred on the EMP code, ROM-EMP must report the error only to the host by setting a bit in a register and must not run the code loaded from RAM. It must also open the hardware NVM security to enable fixing the issue via software tools.

If the CRC8 error occurred on the manageability module, NC-SI/SMBus interfaces are disabled.

If the CRC8 error occurred on the EMP global module then the settings are not loaded to the PHY.

Each time software tools are updating one of these modules, they must update its CRC8 field accordingly.

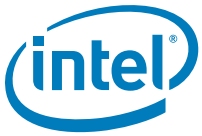
EMP must check the concerned CRC8 fields before committing an NVM Update command that was explicitly addressed for one of these modules. If a CRC8 field read from the free provisioning area of the Flash is not valid, the command completes with the EIO (Flash defect status).

6.1.4.3 NVM integrity summary

Table 6-3. NVM integrity summary table

NVM Module	Integrity Check	When[1]	By	Number of Tries	Action on Error
Shadow RAM (excluding VPD area)	Checksum	Driver Init	PF driver	2	Disable Rx/Tx paths for the host (but not for BMC) and report the failure to user/admin.
VPD area	No	N/A	N/A	N/A	N/A
PCIe Analog	CRC8	EMPR	EMP	2	Report the failure in GL_MNG_FWSM.EXT_ERR_IND. Post to the BMC an AEN # 0x82 - NVM error (enable bit = 18) over NC-SI and then use the Status Data Byte 2[5] in the SMBus alert.
PHY Analog	CRC8	EMPR	EMP	2	Report the failure in GL_MNG_FWSM.EXT_ERR_IND. Post to the BMC an AEN # 0x82 - NVM error (enable bit = 18) over NC-SI and then use the Status Data Byte 2[5] in the SMBus alert.
EMP Global	CRC8	EMPR	EMP	2	Report the failure in GL_MNG_FWSM.EXT_ERR_IND. Post to the BMC an AEN # 0x82 - NVM error (enable bit = 18) over NC-SI and then use the Status Data Byte 2[5] in the SMBus alert. Do not load the data into the PHY.
Manageability	CRC8	EMPR	EMP	2	Report the failure in GL_MNG_FWSM.EXT_ERR_IND. Disable NC-SI/SMBus.
EMP Settings	CRC8	EMPR	EMP	2	Report the failure in GL_MNG_FWSM.EXT_ERR_IND. Post to the BMC an AEN # 0x82 - NVM error (enable bit = 18) over NC-SI and then use the Status Data Byte 2[5] in the SMbus alert.
Option ROM	Yes	Boot Time	BIOS		
EMP Image	CRC8	EMPR	ROM-EMP	2	Report the failure in GL_MNG_FWSM.EXT_ERR_IND. Do not run the RAM code. Open hardware NVM security by clearing the FLA.LOCKED bit. Check CRC8 before committing an EMP image update. In case of an CRC8 error, report the error in the AQ completion and reject the update.

1. Besides the integrity check performed by EMP before committing NVM Update commands.



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 6-4. NVM header of modules mapped to shadow rAM

Number of Words	Field or Segment Name	Description and Comments
1	Module Length	Length of the module contents expressed in words (*Module Length* field excluded). It must be set to N. Modules are size limited to the size of the shadow RAM (64 KB).
N	Word 1	
	Word 2	
	...	
	Word N	

6.1.5.2 Header/trailer of authenticated NVM modules

This section concerns the following modules:

1. EMP Image (pointed by NVM word 0x0B).
2. PCIe Analog (pointed by NVM word 0x03).
3. PHY Analog (pointed by NVM word 0x04).
4. Option ROM (pointed by NVM word 0x05).

For the last three modules, the first 330 words listed in Table 6-5 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 6-5.

Since authenticated modules are by definition, they cannot be modified *in the fields because* no holes are present in such modules.

In Table 6-5, fields colored in cyan are protected by the authentication signature.

Table 6-5. Header of authenticated NVM modules

Number of Words	Field or Segment Name	Description and Comments
64	CSS Header	Refer to Section 3.3.9.2.
128	RSA Public Key	Refer to Section 3.3.9. This field is skipped due to SHA256 Hash computing.
2	RSA Exponent	Refer to Section 3.3.9. This field is skipped due to SHA256 Hash computing.

Table 6-5. Header of authenticated NVM modules (Continued)

Number of Words	Field or Segment Name	Description and Comments
128	Encrypted SHA256 Hash	Refer to Section 3.3.9. This field is skipped due to SHA256 Hash computing.
1	XL710 Blank NVM Device ID	A unique Intel-provided device ID that identifies XL710 controller among other Intel controllers. It must be set to 0x154B in the XL710.
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 580 K words (1160 KB) for an EMP image module and to 4 K 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 must always be set to 0b in the OROM module. Bits 14:8 = Module format version. Set to 0x02touse the currently-defined format. Bits 7:0 = CRC8 value computed over the entire area allocated to the module (1160 KB for EMP image), starting from CSS header (excluding the fields not covered by the RSA authentication signature) and including all the remaining bytes of the area (excluding this word that is skipped for the sake of CRC8 computing.
1	Code Revision	Bits 15:8 = Major revision number. Bits 7:0 = Minor revision number. It is the image revision number (not necessarily referring to any embedded code).
1	Reserved Spare Word	Must be zeroed.
2 or 1	Parent Module Length	Length of the parent module contents expressed in words, module header and *Parent Module Length* field excluded. It excludes all the descendant modules. Modules read by firmware are NOT size limited to 128 KB. • For modules parsed by firmware (EMP image), this length field is two words long and it covers for the firmware code, its padding words, and for the EMP image also the last 4 KB sector reserved for the RO commands section. It must be set to 0x0006DEB4. • For modules parsed by hardware (PCIe analog and PHY analog modules), this length field is one word long and it does not include padded words.
N	Parent Word 1	Firmware module content formatting is specific to each module.
	Parent Word 2	
	...	
	Parent Word N	

6.1.6 Trailer of the EMP image module

In Table 6-6, fields colored in cyan are protected by the authentication signature.

The last 4 KB sector of the EMP image has the following format:

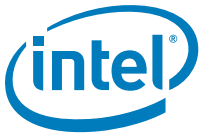


Table 6-6. RO commands section format

Number of Words	Field or Segment Name	Description and Comments
1	RO Commands Version	Default is 0xFFFF, which means the section is empty and the remaining words are discarded.
1	XL710 Blank NVM Device ID	A unique Intel-provided device ID that identifies XL710 controller among other Intel controllers. It must be set to 0x154B in the XL710.
1	Minimum EMP Code Revision	Minimum EMP code revision number required for being able to parse the RO commands section. It must be lower or equal to the code revision number read from the EMP image header listed in Table 6-5.
1	RO Commands Length	Length in words (N), starting from next word.
N	RO Commands word 1	Format of the RO Commands is described in Section 6.1.6.1.
	RO Commands word 2	
	...	
	RO Commands word N	

6.1.6.1 Format of the RO commands

The RO command words can contain the following two structures:

1. Shadow RAM Word Write Command (2 words)
2. CSR Write Command (4 words)

Each command starts with a type field.

Table 6-7 describes the different commands types:

Table 6-7. RO commands types

Type	Description
xxx1b	Word auto load.
0010b	CSR auto load.
Other	Invalid type, parsing is stopped here.

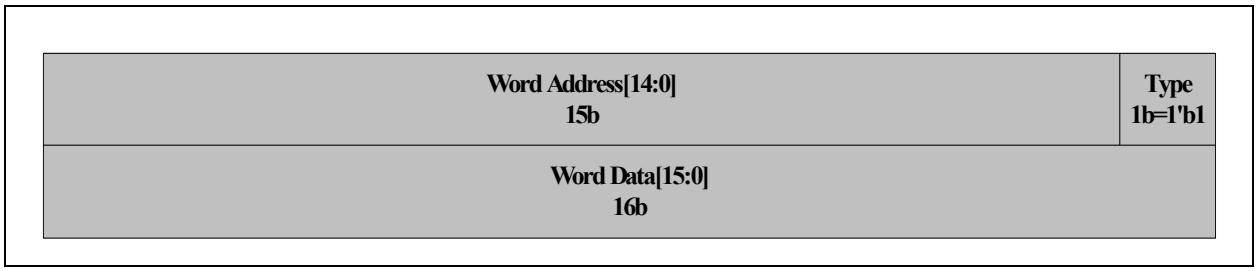


Figure 6-10. Shadow RAM word write command

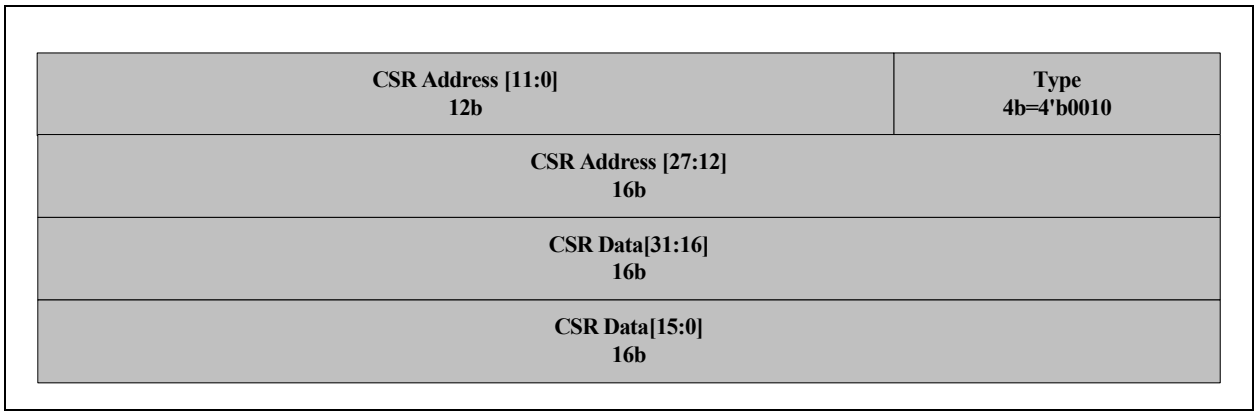
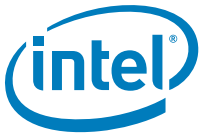


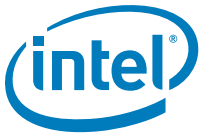
Figure 6-11. CSR write command



6.2 NVM general summary table

Word Address	Used By	Word Name	Reference
0x0000	HW	NVM Control Word 1	299
0x0001	FW	RO Commands Version	299
0x0003	HW	PCIe Analog Module Pointer	299
0x0004	HW	PHY Analog Module Pointer	300
0x0005	HW	OROM Pointer	300
0x0006	HW	RO PCIR Registers Auto-Load Module Pointer	301
0x0007	FW	Auto Generated Pointers Pointer	301
0x0008	HW	PCIR Registers AutoLoad Module Pointer	302
0x0009	FW	EMP Global Module Pointer	302
0x000A	HW	RO PCIe LCB Module Pointer	302
0x000B	FW	EMP Image Pointer	303
0x000D	HW	CSR Protected List Pointer	303
0x000E	FW	Manageability Module Pointer	303
0x000F	FW	EMP Settings Module Pointer	304
0x0010	SW	SW Compatibility Word 1	304
0x0011	SW	SW Compatibility Word 2	304
0x0012	SW	SW Compatibility Word 3	305
0x0013	SW	SW Compatibility Word 4	305
0x0014	SW	SW Compatibility Word 5	305
0x0015	SW	PBA Flags	306
0x0016	SW	PBA Block Pointer	306
0x0017	SW	Boot Configuration Start Address	306
0x0018	SW	Software Reserved Word 1 - Dev Starter Version	306
0x0019	SW	Software Reserved Word 2	307
0x001A	SW	Software Reserved Word 3	308
0x001B	SW	Software Reserved Word 4	308
0x001C	SW	Software Reserved Word 5	308
0x001D	SW	Software Reserved Word 6	308
0x001E	SW	Software Reserved Word 7	308
0x001F	SW	Software Reserved Word 8	308
0x0020	SW	Software Reserved Word 9	309
0x0021	SW	Software Reserved Word 10	309
0x0022	SW	Software Reserved Word 11	309
0x0023	SW	Software Reserved Word 12	309
0x0024	SW	Software Reserved Word 13	309
0x0025	SW	Software Reserved Word 14	309
0x0026	SW	Software Reserved Word 15	310

Word Address	Used By	Word Name	Reference
0x0027	SW	Software Reserved Word 16	310
0x0028	SW	Software Reserved Word 17 - Permanent SAN MAC address ptr	310
0x0029	SW	Software Reserved Word 18 - Map Version	310
0x002A	SW	Software Reserved Word 19 - NVM_Image_Version	311
0x002B	SW	Software Reserved Word 20	311
0x002C	SW	Software Reserved Word 21 - FCoE Offload	311
0x002D	SW	Software Reserved Word 22 - EETRACK ID 1	311
0x002E	SW	Software Reserved Word 23 - EETRACK ID 2	312
0x002F	SW	VPD Module Pointer	312
0x0030	SW	PXE Setup Options Pointer	312
0x0031	SW	PXE Configuration Customization Options Pointer	313
0x0032	SW	PXE Version	313
0x0033	SW	IBA Capabilities	314
0x0034	SW	Software Reserved Word 24 - Original EETRACK ID 1	315
0x0035	SW	Software Reserved Word 25 - Original EETRACK ID 2	315
0x0036	SW	iSCSI Option ROM Version	315
0x0037	SW	Reserved	316
0x0038	HW	POR Registers Auto-Load Module Pointer	316
0x003A	HW	EMPR Registers Auto-Load Pointer	316
0x003B	HW	GLOBR Registers Auto-Load Pointer	317
0x003C	HW	CORER Registers Auto-Load Pointer	317
0x003F	SW	Software Checksum	318
0x0040	FW	1st Free Provisioning Area Pointer	318
0x0041	FW	1st Free Provisioning Area Size	319
0x0042	FW	Reserved for 4th Free Provisioning Area Pointer	319
0x0043	FW	Reserved for 4th Free Provisioning Area Size	320
0x0044	FW	Reserved for 3rd Free Provisioning Area Pointer	320
0x0045	FW	Reserved for 3rd Free Provisioning Area Size	321
0x0046	FW	2nd Free Provisioning Area Pointer	321
0x0047	FW	2nd Free Provisioning Area Size	322
0x0048	HW	EMP SR Settings Pointer	322
0x0049	HW	Reserved	322
0x004A	HW	Core Mem Config Pointer	322
0x004B	FW	FCoE Scratch Pad Area Pointer	323
0x004C	FW	FCoE Scratch Pad Area Size	323
0x004D + 1*n, n=0...180	HW	Spare NVM Header Words	323



6.2.1 Init module section summary table

This is the NVM header module that contains pointers to all other first level sections. It also includes words that are relative to the entire NVM map.

6.2.1.1 NVM control word 1 - 0x0000

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved
7:6	NVM Validity	0x1	The *Signature* field indicates to the XL710 that there is a valid NVM present. If the Signature field is not 01b, the other bits in this word are ignored, no further NVM read is performed, and the default values are used for the configuration space IDs. Valid values are: 0x0 = NVM not present 0. 0x1 = NVM present. 0x2 = NVM not present 2. 0x3 = NVM not present 3.
5	Reserved	0x0	Reserved.

6.2.1.2 RO commands version - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	RO Commands Version	0xFFFF	Contains the version of the RO commands section mapped to the last 4 KB sector of the EMP image.

6.2.1.3 PCIe analog module pointer - 0x0003

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	PCIe Analog Configuration Module Pointer	0x0	

The field PCIe analog configuration module pointer points to the PCIe analog section. For more details about the PCIe analog inner structure, see page 506.

6.2.1.4 PHY analog module pointer - 0x0004

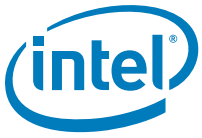
Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	PHY Analog Module Pointer	0x0	

The field PHY analog module pointer points to the PHY analog section. For more details about the PHY analog inner structure, see page 516.

6.2.1.5 OROM pointer - 0x0005

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Only the 4 KB sector unit is supported for this pointer. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	PCIe Expansion/ Option ROM Pointer	0x7FFF	

The field PCIe expansion/option ROM pointer points to the OROM section. For more details about the OROM inner structure, see page 620.



6.2.1.6 RO PCIR registers auto-load module pointer - 0x0006

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	PCIR Registers Auto-Load Module Pointer	0x7FFF	

The field PCIR registers auto-load module pointer points to the RO PCIR registers auto-load module section. For more details about the RO PCIR registers auto-load module inner structure, see page 323.

6.2.1.7 Auto generated pointers pointer - 0x0007

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	Auto Generated Pointers Module Pointer	0x7FFF	

The field auto generated pointers module pointer points to the auto generated pointers module section. For more details about the auto generated pointers module inner structure, see page 497.

6.2.1.8 PCIR registers auto-load module pointer - 0x0008

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	PCIe TL Shared Module Pointer	0x7FFF	

The field PCIe TL shared module pointer points to the PCIR registers auto-load module section. For more details about the PCIR registers auto-load module inner structure, see page 368.

6.2.1.9 EMP global module pointer - 0x0009

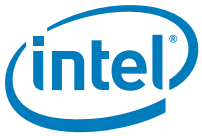
Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	EMP Global Module Pointer	0x7FFF	

The field EMP global module pointer points to the EMP global module section. For more details about the EMP global module inner structure, see page 527.

6.2.1.10 RO PCIe LCB module pointer - 0x000A

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	RO PCIe LCB Module Pointer	0x7FFF	

The field RO PCIe LCB module pointer points to the RO PCIe LCB module section. For more details about the RO PCIe LCB module inner structure, see page 324.



6.2.1.11 EMP image pointer - 0x000B

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	EMP Image Pointer	0x7FFF	

The field EMP image pointer points to the EMP image section. For more details about the EMP image inner structure, see page 638.

6.2.1.12 CSR protected list pointer - 0x000D

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	CSR Protected List Pointer	0x7FFF	

The field CSR protected list pointer points to the CSR protected list section. For more details about the CSR protected list inner structure, see page 325.

6.2.1.13 Manageability module pointer - 0x000E

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	Manageability Configuration Module Pointer	0x7FFF	

The field manageability configuration module pointer points to the manageability module header section. For more details about the manageability module header inner structure, see page 532.

6.2.1.14 EMP settings module pointer - 0x000F

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	EMP Module Pointer	0x7FFF	

The field EMP module pointer points to the EMP settings module header section. For for details about the EMP settings module header inner structure, see page 556.

6.2.1.15 SW compatibility word 1 - 0x0010

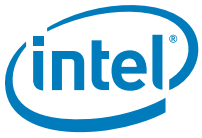
Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11	LOM	0x0	Indicates whether the NVM attached to the XL710 contains a dedicated module for an option ROM. This is used by an option ROM update applications. 0b = NIC (attached Flash contains a module for an option ROM). 1b = LOM (attached Flash has no module for an option ROM).
10	Server	0x1	Legacy, not currently used. 1b = Server. 0b = Client.
9	Reserved	0x0	Reserved.
8	OEM/Retail	0x0	Legacy, not currently used. 1b = OEM. 0b = Retail.
7:0	Reserved	0x00	Reserved.

6.2.1.16 SW compatibility word 2 - 0x0011

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

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



6.2.1.17 SW compatibility word 3 - 0x0012

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

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

6.2.1.18 SW compatibility word 4 - 0x0013

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

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

6.2.1.19 SW compatibility word 5 - 0x0014

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

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

6.2.1.20 PBA flags - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	PBA Flags	0xFAFA	A flag value of 0xFAFA indicates that the PBA is stored in a separate PBA block.

6.2.1.21 PBA block pointer - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	PBA Block Pointer		

6.2.1.22 Boot configuration start address - 0x0017

The address of the iSCSI Boot configuration module. This is a word pointer along with the block length embedded in the module.

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	iSCSI Boot Configuration Start Address	0x7FFF	This module is 1504 bytes long and must be mapped in the first valid 4 KB sector of the Flash.

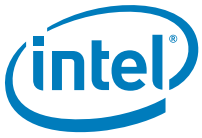
The field iSCSI boot configuration start address points to the boot configuration block section. For more details about the boot configuration block inner structure, see page 341.

6.2.1.23 Software reserved word 1 - dev starter version - 0x0018

The Dev_Starter map version used to produce this image.

This word must be filled in manually,

Bits	Field Name	Default NVM Value	Description
15:12	Major		NVM major version.
11:8	Decimal Point	0x0	Decimal point, used by automatic NVM reading tools. Must always be set to 0x0.



Bits	Field Name	Default NVM Value	Description
7:0	Minor		NVM minor version.

6.2.1.24 Software reserved word 2 - 0x0019

Bits	Field Name	Default NVM Value	Description
15:4	Reserved	0xFFFF	Reserved.
3	WoL Control Port 3	0x1	Wake on LAN (WoL) Feature For Port 3. 1b = Disabled or not supported. 0b = Supported and enabled. Valid values are: 0x0 = Enabled Supported and enabled. 0x1 = Disabled or not supported.
2	WoL Control Port 2	0x1	WoL Feature For Port 2. 1b = Disabled or not supported. 0b = Supported and enabled. Valid values are: 0x0 = Enabled Supported and enabled. 0x1 = Disabled or not supported.
1	WoL Control Port 1	0x1	WoL Feature For Port 1. 1b = Disabled or not supported. 0b = Supported and enabled. Valid values are: 0x0 = Enabled Supported and enabled. 0x1 = Disabled or not supported.
0	WoL Control Port 0	0x1	WoL Feature For Port 0. 1b = Disabled or not supported. 0b = Supported and enabled. Valid values are: 0x0 = Enabled Supported and enabled. 0x1 = Disabled or not supported.

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

6.2.1.25 Software reserved word 3 - 0x001A

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

6.2.1.26 Software reserved word 4 - 0x001B

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

6.2.1.27 Software reserved word 5 - 0x001C

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

6.2.1.28 Software reserved word 6 - 0x001D

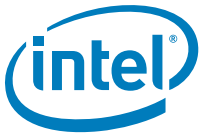
Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.1.29 Software reserved word 7 - 0x001E

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

6.2.1.30 Software reserved word 8 - 0x001F

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



6.2.1.31 Software reserved word 9 - 0x0020

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

6.2.1.32 Software reserved word 10 - 0x0021

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

6.2.1.33 Software reserved word 11 - 0x0022

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

6.2.1.34 Software reserved word 12 - 0x0023

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

6.2.1.35 Software reserved word 13 - 0x0024

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

6.2.1.36 Software reserved word 14 - 0x0025

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

6.2.1.37 Software reserved word 15 - 0x0026

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

6.2.1.38 Software reserved word 16 - 0x0027

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

6.2.1.39 Software reserved word 17 - permanent SAN MAC address ptr - 0x0028

Word 0x28 points to the permanent SAN MAC address block used for FCoE (SPMA and FPMA) and DCB.

One MAC address per enabled PF.

This is a per-device, per-enabled PF value allocated by manufacturing for FCoE boot NVM images.

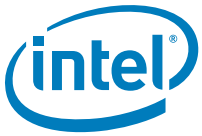
Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	SAN MAC Address Pointer	0x7FFF	Pointer to the permanent SAN MAC address block. An all one's value in this field indicates an invalid pointer.

The field SAN MAC address pointer points to the permanent SAN MAC address section. For more details about the permanent SAN MAC address inner structure, see page 333.

6.2.1.40 Software reserved word 18 - Map Version - 0x0029

Automatically generated by the tool.

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



6.2.1.41 Software reserved word 19 - NVM_Image_Version - 0x002A

Automatically generated by the tool.

Bits	Field Name	Default NVM Value	Description
15:0	NVM Image Version		

6.2.1.42 Software reserved word 20 - 0x002B

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

6.2.1.43 Software reserved word 21 - FCoE offload - 0x002C

This word is for platform/NIC/LOM specific settings.

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x3FFF	Reserved.
1	FCoE Offload	0x1	This bit indicates to software if the XL710 supports FCoE offload. 0b = FCoE offload enabled. 1b = FCoE offload disabled. Valid values are: 0x0 = FCoE offload enable. 0x1 = FCoE offload disable.
0	Reserved	0x1	Reserved.

6.2.1.44 Software reserved word 22 - EETRACK ID 1 - 0x002D

This word is for the first word of the eTrack_ID number written by the EEPROM manager tool.

Bits	Field Name	Default NVM Value	Description
15:0	eTrack_ID Word 1	0xFFFF	The EEPROM manager tool writes a unique 32-bit eTrack_ID number in two sequential NVM words. The eTrack_ID is written when the EEPROM manager tool creates an image on the Intel network. The eTrack_ID DB tracks NVM images back to a specific SCM build.

6.2.1.45 Software reserved word 23 - EETRACK ID 2 - 0x002E

This word is for the second word of the eTrack_ID number written by the EEPROM manager tool.

Bits	Field Name	Default NVM Value	Description
15:0	eTrack_ID Word 2		

6.2.1.46 VPD module pointer - 0x002F

Word pointer to the VPD module along with the block length embedded in the module.

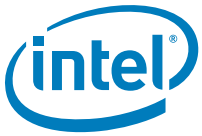
Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	VPD Pointer	0x7FFF	VPD Pointer. 0x7FFF is the default unless a VPD relative section is specified. The VPD section size is usually 64 words and is initialized to 0 or 0x7FFF. During run time, this module is accessible through the VPD capability in the PCI configuration space. This module must be mapped in the first valid 4 KB sector of the Flash.

The field VPD pointer points to the VPD module section. For more details about the VPD module inner structure, see page 333.

6.2.1.47 PXE setup options pointer - 0x0030

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	PXE Setup Options Pointer	0x7FFF	

The field PXE setup options pointer points to the PXE setup options section. For for more details about the PXE setup options inner structure, see page 336.



6.2.1.48 PXE configuration customization options pointer - 0x0031

Word 0x31 of the NVM contains settings that can be programmed by an OEM or network administrator to customize the operation of the software. These settings cannot be changed from within the Control-S setup menu. The lower byte contains settings that would typically be configured by a network administrator using an external utility; these settings generally control which setup menu options are changeable. The upper byte is generally settings that would be used by an OEM to control the operation of the agent in a LOM environment, although there is nothing in the agent to prevent their use on a NIC implementation. The default value for this word is 0x4000.

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	PXE Configuration Customization Options Pointer	0x7FFF	

The field PXE configuration customization options pointer points to the PXE configuration customization options section. For more details about the PXE configuration customization options inner structure, see page 339

6.2.1.49 PXE version - 0x0032

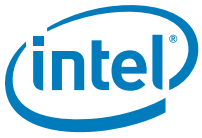
Word 0x32 of the NVM is used to store the version of the boot agent that is stored in the Flash image. When the Boot Agent loads, it can check this value to determine if any first-time configuration needs to be performed. The agent then updates this word with its version. Some diagnostic tools to report the version of the Boot Agent in the Flash also read this word.

Bits	Field Name	Default NVM Value	Description
15:12	Major Version	0x0	PXE Boot Agent Major Version. Default value is 0.
11:8	Minor Version	0x0	PXE Boot Agent Minor Version. Default value is 0.
7:0	Build Number	0x0	PXE Boot Agent Build Number. Default value is 0.

6.2.1.50 IBA capabilities - 0x0033

Word 0x33 of the NVM is used to enumerate the boot technologies that have been programmed into the Flash. This is updated by Flash configuration tools and is not updated or read by IBA.

Bits	Field Name	Default NVM Value	Description
15:14	Signature	0x1	Signature. Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software.
13:6	Reserved	0x000	Reserved. Must be 0x0.
5	FCOE	0x1	FCoE boot code is present if set to 1b. Valid values are: 0x0 = Not present. 0x1 = Present.
4	iSCSI Boot	0x0	iSCSI is present if set to 0x1. Valid values are: 0x0 = Not present. 0x1 = Present.
3	efi ebd Driver	0x0	EFI UNDI driver is present if set to 0x1. Valid values are: 0x0 = Not present. 0x1 = Present.
2	RPL	0x0	RPL module is present if set to 0x1. Reserved bit for devices. Valid values are: 0x0 = Not present. 0x1 = Present.
1	PXE/UNDI Driver	0x1	PXE UNDI driver is present if set to 0x1. Valid values are: 0x0 = Not present. 0x1 = Present.
0	PXE Base Code	0x1	PXE Base Code is present if set to 0x1. Valid values are: 0x0 = Not present. 0x1 = Present.



6.2.1.51 Software reserved word 24 - original EETRACK ID 1 - 0x0034

Bits	Field Name	Default NVM Value	Description
15:0	Original EETRACK ID 1	0x0000	

6.2.1.52 Software reserved word 25 - original EETRACK ID 2 - 0x0035

Bits	Field Name	Default NVM Value	Description
15:0	Original EETRACK ID 2	0x0000	

6.2.1.53 iSCSI option ROM version - 0x0036

Word 0x36 of the NVM is used to store the version of iSCSI Option ROM updated. The value must be above 0x2000 and the value below (word 0x1FFF = 16 KB NVM size) is reserved for future expansion for a pointer to combo option ROM component version structure. iSCSIUtl, FLAUtil, DMiX update iSCSI Option ROM version if the value is above 0x2000, 0x0000, or 0xFFFF. The pointer (0x0040 - 0x1FFF) should be kept and not be overwritten.

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

6.2.1.54 Reserved - 0x0037

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

6.2.1.55 POR registers auto-load module pointer - 0x0038

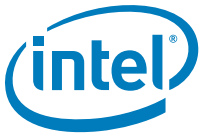
Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	POR Registers Auto-Load Module Pointer	0x7FFF	

The field POR registers auto-load module pointer points to the POR registers auto-load module section. For more details about the POR registers auto-load module inner structure, see page 381.

6.2.1.56 EMPR registers auto-load pointer - 0x003A

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	EMPR Registers Auto-Load Module Pointer	0x7FFF	

The field EMPR registers auto-load module pointer points to the EMPR auto-load section. For more details about the EMPR auto-load inner structure, see page 331.



6.2.1.57 GLOBR registers auto-load pointer - 0x003B

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	GLOBR Registers Auto-Load Module Pointer	0x7FFF	

The field GLOBR registers auto-load module pointer points to the GLOBR registers auto-load module section. For more details about the GLOBR registers auto-load module inner structure, see page 451.

6.2.1.58 CORER registers auto-load pointer - 0x003C

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	CORER Registers Auto-Load Module Pointer	0x7FFF	

The field CORER registers auto-load module pointer points to the CORER registers auto-load module section. For more details about the CORER registers auto-load module inner structure, see page 396.

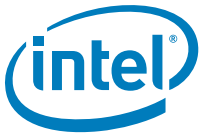
6.2.1.59 Software checksum - 0x003F

Bits	Field Name	Default NVM Value	Description
15:0	Checksum		The software checksum field covers the entire 64 KB shadow RAM contents (reserved words included) except the VPD area and the PCIe ALT auto-load module, which are skipped. 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 used strictly by software. Hardware does not calculate nor check its content but rather checks the NVM validity field in the NVM Control Word 1.

6.2.1.60 1st free provisioning area pointer - 0x0040

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	1st Free Provisioning Area Pointer	0x7FFF	

The field 1st free provisioning area pointer points to 1st free provisioning area section. For more details about the 1st free provisioning area inner structure, see page 647.



6.2.1.61 1st free provisioning area size - 0x0041

Bits	Field Name	Default NVM Value	Description
15:10	Reserved	0x00	
9:0	1st Free Provisioning Area Size	0x0DC	Size expressed in 4 KB sector units.

6.2.1.62 Reserved for 4th free provisioning area pointer - 0x0042

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	2nd Free Provisioning Area Pointer	0x7FFF	

The field 2nd free provisioning area pointer points to the 4th free provisioning area section. For more details about the 4th free provisioning area inner structure, see page 648.

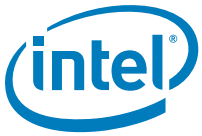
6.2.1.63 Reserved for 4th free provisioning area size - 0x0043

Bits	Field Name	Default NVM Value	Description
15:10	Reserved	0x00	
9:0	2nd Free Provisioning Area Size	0x010	Size expressed in 4 KB sector units.

6.2.1.64 Reserved for 3rd free provisioning area pointer - 0x0044

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	3rd Free Provisioning Area Pointer	0x7FFF	

The field 3rd free provisioning area pointer points to the 3rd free provisioning area section. For more details about the 3rd free provisioning area inner structure, see page 648.



6.2.1.65 Reserved for 3rd free provisioning area size - 0x0045

Bits	Field Name	Default NVM Value	Description
15:10	Reserved	0x00	Reserved.
9:0	3rd Free Provisioning Area Size	0x004	Size expressed in 4 KB sector units.

6.2.1.66 2nd free provisioning area pointer - 0x0046

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	4st Free Provisioning Area Pointer	0x7FFF	

The field 2nd free provisioning area pointer points to the 2nd free provisioning area section. For more details about the 2nd free provisioning area inner structure, see page 620.

6.2.1.67 2nd free provisioning area size - 0x0047

Bits	Field Name	Default NVM Value	Description
15:10	Reserved	0x00	Reserved.
9:0	4st Free Provisioning Area Size	0x002	Size expressed in 4 KB sector units.

6.2.1.68 EMP SR settings pointer - 0x0048

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	EMP SR Settings Pointer	0x7FFF	

The field EMP SR settings pointer points to the EMP SR settings module header section. For more details about the EMP SR settings module header inner structure, see page 492.

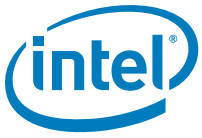
6.2.1.69 Reserved - 0x0049

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

6.2.1.70 Core mem config pointer - 0x004A

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	Core Mem Config Pointer	0x7FFF	

The field core mem config pointer points to the core mem config section. For more detaisl about the core mem config inner structure, see page 332.



6.2.1.71 FCoE scratch pad area pointer - 0x004B

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type. 0b = Word units. 1b = 4 KB sector units. Valid values are: 0x0 = Word units. 0x1 = 4 KB sector units.
14:0	FCoE Scratch Pad Area Pointer	0x7FFF	

The field FCoE scratch pad area pointer points to the FCoE scratch pad area section. For more details about the FCoE scratch pad area inner structure, see page 648.

6.2.1.72 FCoE scratch pad area size - 0x004C

Bits	Field Name	Default NVM Value	Description
15:10	Reserved	0x00	Reserved.
9:0	FCoE Scratch Pad Area Size	0x100	Size expressed in 4 KB sector units.

6.2.1.73 Spare NVM header words[n] (0x004D + 1*n, n=0...180)

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

6.2.2 RO PCIR registers auto-load module section summary table

Contains RO parameters that configure the PCIe transaction layer.

Word Offset	Description	Reference
0x0000	Module Length	Section 6.2.2.1
0x0001	LO PCIR Data	Section 6.2.2.2

6.2.2.1 Module length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		

6.2.2.2 LO PCIR data - 0x0001

Raw data module length: variable.

This word must be disabled at the image level.

6.2.3 RO PCIe LCB module section summary table

Contains RO parameters that configure the PCIe link layer (LCB unit).

Word Offset	Description	Reference
0x0000	Module Length	Section 6.2.3.1
0x0001	RO PCIe LCB Data	Section 6.2.3.2

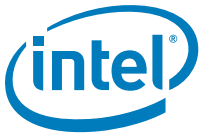
6.2.3.1 Module length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		

6.2.3.2 RO PCIe LCB data - 0x0001

Raw data module length: variable.

This word must be disabled at the image level.



6.2.4 CSR protected list section summary table

Defines the list of the protected CSRs and their default settings. These registers are made RO to the host as they contain settings that are critical for the host to Flash access when in blank Flash programming mode.

Word Offset	Description	Reference
0x0000	Module Length	page 326
0x0001	Reserved	page 326
0x0002 - 0006	NVM contents for GLPCI_LCBADD	page 326
0x0007 - 0008	NVM contents for GLPCI_LCBDATA	page 326
0x0009 - 000D	NVM contents for PRTMAC_PHY_ANA_ADD	page 327
0x000E - 000F	NVM contents for PRTMAC_PHY_ANA_DATA	page 327
0x0010 - 0014	NVM contents for MEM_INIT_GATE_AL_STR	page 327
0x0015 - 0016	NVM contents for MEM_INIT_GATE_AL_DONE	page 328
0x0017 - 001A	NVM contents for GLGEN_STAT	page 328
0x001B - 001E	NVM contents for GLNVM_ALTIMERS	page 329
0x001F - 0023	NVM contents for GLPCI_ANA_ADD	page 329
0x0024 - 0025	NVM contents for GLPCI_ANA_DATA	page 330
0x0026 - 0027	NVM contents for GLPHY_ANA_ADD	page 330
0x0028 - 0029	NVM contents for GLPHY_ANA_DATA	page 330
0x002A - 002D	NVM contents for GLPCI_LBARCTRL	page 330
0x002E - 00A8	NVM contents for GLNVM_PROTCSR	page 331

6.2.4.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 bytes unit - 1. First section -> Word: CSR Protected List -> Module Length. Last section -> Word: CSR Protected List -> Starting Address Low at GLNVM_PROTCSR[0].

6.2.4.2 Reserved - 0x0001

This word must be disabled at the image level.

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

6.2.4.3 GLPCI_LCBADD - (0x0002 - 0x0006)

6.2.4.3.1 Starting address high at GLPCI_LCBADD - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_LCBADD		

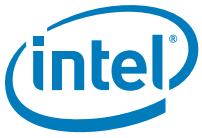
6.2.4.3.2 Data low of GLPCI_LCBADD - 0x0005

6.2.4.3.3 Data high of GLPCI_LCBADD - 0x0006

6.2.4.4 GLPCI_LCBDATA - (0x0007 - 0x0008)

6.2.4.4.1 Data low of GLPCI_LCBDATA - 0x0007

6.2.4.4.2 Data high of GLPCI_LCBDATA - 0x0008



6.2.4.5 PRTMAC_PHY_ANA_ADD - (0x0009 - 0x000D)

6.2.4.5.1 Starting address low at PRTMAC_PHY_ANA_ADD - 0x0009

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_PHY_ANA_ADD	0xA4038	
3:0	Type	0x2	

6.2.4.5.2 Starting address high at PRTMAC_PHY_ANA_ADD - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_PHY_ANA_ADD		

6.2.4.5.3 Attributes at PRTMAC_PHY_ANA_ADD - 0x000B

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.4.6 PRTMAC_PHY_ANA_DATA - (0x000E - 000F)

6.2.4.6.1 Data high of PRTMAC_PHY_ANA_DATA - 0x000F

6.2.4.7 MEM_INIT_GATE_AL_STR - (0x0010 - 0x0014)

6.2.4.7.1 Starting address low at MEM_INIT_GATE_AL_STR - 0x0010

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of MEM_INIT_GATE_AL_STR	0xB6000	

Bits	Field Name	Default NVM Value	Description
3:0	Type	0x2	

6.2.4.7.2 Starting address high at MEM_INIT_GATE_AL_STR - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of MEM_INIT_GATE_AL_STR		

6.2.4.7.3 Attributes at MEM_INIT_GATE_AL_STR - 0x0012

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.4.7.4 Data low of MEM_INIT_GATE_AL_STR - 0x0013

6.2.4.7.5 Data high of MEM_INIT_GATE_AL_STR - 0x0014

6.2.4.8 MEM_INIT_GATE_AL_DONE - (0x0015 - 0x0016)

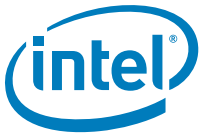
6.2.4.8.1 Data low of MEM_INIT_GATE_AL_DONE - 0x0015

6.2.4.8.2 Data high of MEM_INIT_GATE_AL_DONE - 0x0016

6.2.4.9 GLGEN_STAT - (0x0017 - 0x001A)

6.2.4.9.1 Address low at GLGEN_STAT - 0x0017

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLGEN_STAT	0xB612C	
3:0	Type	0x1	



6.2.4.9.2 Address high at GLGEN_STAT - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_STAT		

6.2.4.9.3 Data low of GLGEN_STAT - 0x0019

6.2.4.9.4 Data high of GLGEN_STAT - 0x001A

6.2.4.10 GLNVM_ALTIMERS - (0x001B - 001E)

6.2.4.10.1 Address high at GLNVM_ALTIMERS - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLNVM_ALTIMERS		

6.2.4.10.2 Data low of GLNVM_ALTIMERS - 0x001D

6.2.4.10.3 Data high of GLNVM_ALTIMERS - 0x001E

6.2.4.11 GLPCI_ANA_ADD - (0x001F - 0x0023)

6.2.4.11.1 Starting address low at GLPCI_ANA_ADD - 0x001F

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_ANA_ADD	0xBA000	
3:0	Type	0x2	

6.2.4.11.2 Starting address high at GLPCI_ANA_ADD - 0x0020

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_ANA_ADD		

6.2.4.11.3 Attributes at GLPCI_ANA_ADD - 0x0021

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.4.12 GLPCI_ANA_DATA - (0x0024 - 0x0025)

6.2.4.12.1 Data high of GLPCI_ANA_DATA - 0x0025

6.2.4.13 GLPHY_ANA_ADD - (0x0026 - 0x0027)

6.2.4.14 GLPHY_ANA_DATA - (0x0028 - 0x0029)

6.2.4.14.1 Data high of GLPHY_ANA_DATA - 0x0029

6.2.4.15 GLPCI_LBARCTRL - (0x002A - 002D)

6.2.4.15.1 Address low at GLPCI_LBARCTRL - 0x002A

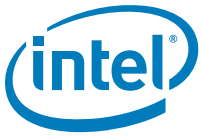
Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_LBARCTRL	0xBE484	
3:0	Type	0x1	

6.2.4.15.2 Address high at GLPCI_LBARCTRL - 0x002B

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_LBARCTRL		

6.2.4.15.3 Data low of GLPCI_LBARCTRL - 0x002C

6.2.4.15.4 Data high of GLPCI_LBARCTRL - 0x002D



6.2.4.16 GLNVM_PROTCSR - (0x002E - 0x00A8)

6.2.4.16.1 Starting address low at GLNVM_PROTCSR[n] (0x002E)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLNVM_PROTCSR	0xB6010	
3:0	Type	0x2	

6.2.4.16.2 Starting address high at GLNVM_PROTCSR[n] (0x002F)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLNVM_PROTCSR		

6.2.4.16.3 Attributes at GLNVM_PROTCSR[n] (0x0030)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x3C	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.4.16.4 Data low of GLNVM_PROTCSR[n] (0x0031 + 2*n, n=0...59)

6.2.4.16.5 Data high of GLNVM_PROTCSR[n] (0x0032 + 2*n, n=0...59)

6.2.5 EMPR auto-load section summary table

Read margin settings to the XL710's core memories that are loaded by hardware auto-load only on the first EMPR event that occurs after POR.

The contents of this module is checked against the CSR protected list.

Word Offset	Description	Reference
0x0000	Module Length	332
0x0001	EMPR Autoload Data	332

6.2.5.1 Module length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First section -> Word: EMPR Autoload -> Module Length Last section -> Word: EMPR Autoload -> EMPR Autoload Data

6.2.5.2 EMPR auto-load Data - 0x0001

Raw data module length: variable.

Contains patches.

6.2.6 Core mem config section summary table

Read margin settings to the XL710's core memories that are loaded by hardware auto-load on every CORER event.

The contents of this module is checked against the CSR protected list.

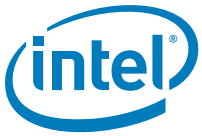
Word Offset	Description	Reference
0x0000	Module Length	332
0x0001	Core Mem Config Data	page 332

6.2.6.1 Module length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First section -> Word: Core Mem Config -> Module Length Last section -> Word: Core Mem Config -> Core Mem Config Data

6.2.6.2 Core mem config data - 0x0001

Raw data module length: variable.



6.2.7 VPD module section summary table

VPD loaded by OEM. It contains RO and RW information about the NIC/LOM.

Word Offset	Description	Reference
0x0000	[New Word]	333

6.2.7.1 [New Word] - 0x0000

Raw data module length: 512 words.

6.2.8 Spare SR words section summary table

Spare words in SR to separate between RO and RW words.

Word Offset	Description	Reference
0x0000 + 1*n, n=0...4095	Reserved	324

6.2.8.1 Module length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		

6.2.9 Permanent SAN MAC address section summary table

MAC addresses to be used for SAN.

Word Offset	Description	Reference
0x0000	Module Length	334
0x0001 + 3*n, n=0...15	SAN MAC Address Word 0 for PF	334
0x0002 + 3*n, n=0...15	SAN MAC Address Word 1 for PF	334
0x0003 + 3*n, n=0...15	SAN MAC Address Word 2 for PF	334

6.2.9.1 Module length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		

6.2.9.2 SAN MAC address word 0 for PF[n] (0x0001 + 3*n, n=0...15)

PF index n is for PF 0 up to PF 15.

Bits	Field Name	Default NVM Value	Description
15:0	SAN MAC Address Word 0 for PF[n]	0xFFFF	

6.2.9.3 SAN MAC address word 1 for PF[n] (0x0002 + 3*n, n=0...15)

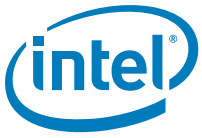
PF index n is for PF 0 up to PF 15.

Bits	Field Name	Default NVM Value	Description
15:0	SAN MAC Address Word 1 for PF[n]	0xFFFF	

6.2.9.4 SAN MAC address word 2 for PF[n] (0x0003 + 3*n, n=0...15)

PF index n is for PF 0 up to PF 15.

Bits	Field Name	Default NVM Value	Description
15:0	SAN MAC Address Word 2 for PF[n]	0xFFFF	



6.2.10 PBA block section summary table

The PBA block contains the complete PBA number including the dash and the first digit of the 3-digit suffix.

Word Offset	Description	Reference
0x0000	PBA Section Length	336
0x0001	Word1	336
0x0002	Word2	336
0x0003	Word3	336
0x0004	Word4	336
0x0005	Word5	336

6.2.10.1 PBA section length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	PBA Section Length Field	0x6	Length in words of the PBA block.

6.2.10.2 Word1 - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Word1 Field		PBA number stored in hexadecimal ASCII values.

6.2.10.3 Word2 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Word2 Field		PBA number stored in hexadecimal ASCII values.

6.2.10.4 Word3 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Word3 Field		PBA number stored in hexadecimal ASCII values.

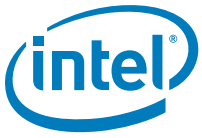
6.2.10.5 Word4 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Word4 Field		PBA number stored in hexadecimal ASCII values.

6.2.10.6 Word5 - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Word5 Field		PBA number stored in hexadecimal ASCII values.

6.2.11 PXE setup options section summary table



Setup options for the PXE driver defined per PCIe function.

Word Offset	Description	Reference
0x0000	Section Length	337
0x0001 + 1*n, n=0...15	Setup Options PCI Function	337

6.2.11.1 Section length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

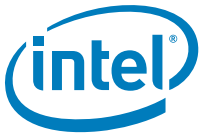
Bits	Field Name	Default NVM Value	Description
15:0	Section Length		

6.2.11.2 Setup options PCI Function[n] (0x0001 + 1*n, n=0...15)

The main setup options for PF n are stored in this word. These options are those that can be changed by the user using the Control-S setup menu.

Bits	Field Name	Default NVM Value	Description
15:13	Reserved	0x00	Reserved. Must be 0x0.
12:10	FSD	0x0	Bits 12-10 control forcing speed and duplex during driver operation. Valid values are: 000b = Auto-negotiate. 001b = 10 Mb/s half duplex. 010b = 100 Mb/s half duplex. 011b = Not valid (treated as 000b). 100b = 10 Mb/s full duplex. 101b = 100 Mb/s full duplex. 111b = 1000 Mb/s full duplex. Only applicable for copper-based adapters. Not applicable to 10 GbE. Default value is 000b. Valid values are: 0x0 = Auto-negotiate. 0x1 = 10 Mb/s half duplex. 0x2 = 100 Mb/s half duplex. 0x4 = 10 Mbps full duplex. 0x5 = 100 Mb/s full duplex. 0x7 = 1000 Mb/s full duplex.
9	Reserved	0x0	Reserved to legacy OS wake-up support. (For 82559-based adapters only), which is not supported in the XL710.
8	DSM	0x1	Display Setup Message. If the bit is set to 1b, the Press Control-S message is displayed after the title message. Default value is 1b.

Bits	Field Name	Default NVM Value	Description
7:6	PT	0x0	Prompt Time. These bits control how long the Press Control-S setup prompt message is displayed, if enabled by DIM. 00b = 2 seconds (default). 01b = 3 seconds. 10b = 5 seconds. 11b = 0 seconds. **Note:** The Ctrl-S message is not displayed if 0 seconds prompt time is selected. Valid values are: 0x0 = 2 seconds. 0x1 = 3 seconds. 0x2 = 5 seconds. 0x3 = 0 seconds.
5	iSCSI Boot disabled	0x0	When this bit is set and the 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. This bit must be changed at factory level and not be altered by any end-customer tools. **Note:** For regular NICs and LOM designs this bit should be always cleared in the NVM image; otherwise, iSCSI setup is not accessible for the user. Valid values are: 0x0 = Enabled. 0x1 = Disabled.
4:3	DBS	0x0	Default Boot Selection. These bits select which device is the default boot device. These bits are only used if the agent detects that the BIOS does not support boot order selection or if the *Mode* field of word 0x31 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. Valid values are: 0x0 = Network then local, 0x1 = Local then network. 0x2 = Network only. 0x3 = Local only.
2:0	PS	0x0	Protocol Select. These bits select the active boot protocol. 00b = PXE (default value). 01b = RPL (only if RPL is in the Flash). 10b = iSCSI Boot primary port (only if iSCSI boot is using this adapter). 11b = iSCSI Boot secondary port (only if iSCSI boot is using this adapter). Only the default value of 00b should be initially programmed into the adapter; other values should only be set by configuration utilities. Valid values are: 0x0 = PXE (default value). 0x1 = Boot disabled. 0x2 = iSCSI primary. 0x3 = iSCSI secondary. 0x4 = FCoE. 0x7: 0x5 = Reserved.



6.2.12 PXE configuration customization options section summary table

Configuration customization options for the PXE driver. It is defined per PCIe function.

Word Offset	Description	Reference
0x0000	Section Length	339
0x0001 + 1*n, n=0...15	Configuration Customization Options PCI Function	339

6.2.12.1 Section length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

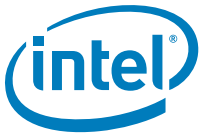
Bits	Field Name	Default NVM Value	Description
15:0	Section Length		

6.2.12.2 Configuration customization options PCI Function[n] (0x0001 + 1*n, n=0...15)

It contains settings that can be programmed by an OEM or network administrator to customize the operation of the software. These settings cannot be changed from within the Control- S setup menu. The lower byte contains settings that would typically be configured by a network administrator using an external utility; these settings generally control which setup menu options are changeable. The upper byte is generally settings that would be used by an OEM to control the operation of the agent in a LOM environment, although there is nothing in the agent to prevent their use on a NIC implementation. The default value for this word is 0x4000.

Bits	Field Name	Default NVM Value	Description
15:14	Signature	0x1	Signature. Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software.
13:12	Reserved	0x0	Reserved. Must be 0x0.
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. Default value is 0b. Valid values are: 0x0 = Disable. 0x1 = Enable.

Bits	Field Name	Default NVM Value	Description
10:8	Operating mode	0x0	Selects the agent's boot order setup mode. This field changes the agent default behavior in order to make it compatible with systems that do not completely support the BBS and PnP Expansion ROM standards. Valid values and their meanings are: 000b = Normal behavior. The agent attempts to detect BBS and PnP Expansion ROM support as it normally does. 001b = Force legacy mode. The agent does not attempt to detect BBS or PnP Expansion ROM supports in the BIOS and assumes the BIOS is not compliant. The user can change the BIOS boot order in the Setup Menu. 010b = Force BBS mode. The agent assumes the BIOS is BBS compliant, even though it might not be detected as such by the agent?s detection code. The user CANNOT change the BIOS boot order in the Setup Menu. 011b = Force PnP Int18 mode. The agent assumes the BIOS allows boot order setup for PnP Expansion ROMs and hooks interrupt 0x18 (to inform the BIOS that the agent is a bootable device) in addition to registering as a BBS IPL device. The user CANNOT change the BIOS boot order in the Setup Menu. 100b = Force PnP Int19 mode. The agent assumes the BIOS allows boot order setup for PnP Expansion ROMs and hook interrupt 0x19 (to inform the BIOS that the agent is a bootable device) in addition to registering as a BBS IPL device. The user CANNOT change the BIOS boot order in the Setup Menu. 101b = Reserved for future use. If specified, is treated as a value of 000b. 110b = Reserved for future use. If specified, is treated as a value of 000b. 111b = Reserved for future use. If specified, is treated as a value of 000b. Valid values are: 0x0 = Normal mode. 0x1 = Force legacy. 0x2 = Force BBS. 0x3 = Force PnP Int18. 0x4 = Force PnP Int19.
7:6	Reserved	0x0	Reserved. Must be 0x0.
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. Default value is 0b. Valid values are: 0x0 = Enable Flash update. 0x1 = Disable Flash update.
4	Disable Legacy OS Wakeup Menu	0b	Disable Legacy Wake-up Support. If this bit is set to 1b, the user is not allowed to change the legacy OS wake-up support menu option. Default value is 0b. Valid values are: 0x0 = Enable legacy wake-up support. 0x1 = Disable legacy wake-up support.
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. Default value is 0b. Valid values are: 0x0 = Enable. 0x1 = Disable.
2	Disable Protocol Selection Menu	0b	Disable Protocol Select. If set to 1b, the user is not allowed to change the boot protocol. Default value is 0b. Valid values are: 0x0 = Enable. 0x1 = Disable.



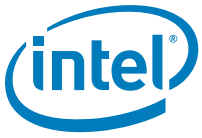
Bits	Field Name	Default NVM Value	Description
1	Disable Title Message Display	0b	Disable Title Message. If this bit is set to 1b, the title message displaying the version of the boot agent is suppressed; the Control-S message is also suppressed. This is for OEMs who do not want the boot agent to display any messages at system boot. Default value is 0b. Valid values are: 0x0 = Enable. 0x1 = 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 might only be changed via an external program. Default value is 0b. Valid values are: 0x0 = Enable. 0x1 = Disable.

6.2.13 Boot configuration block section summary table

Contains the required setup to be used for the boot operations.

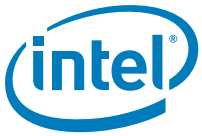
Word Offset	Description	Reference
0x0000	Boot Signature	346
0x0001	Block Size	346
0x0002	Structure Version	346
0x0003	iSCSI Initiator Name	346
0x0083 + 1*n, n=0...16	Reserved	347
0x0094	iSCSI Flags	347
0x0095 + 1*n, n=0...1	iSCSI Initiator IP	347
0x0097 + 1*n, n=0...1	Subnet Mask	347
0x0099 + 1*n, n=0...1	Gateway IP	348
0x009B	iSCSI Boot LUN	348
0x009C + 1*n, n=0...1	iSCSI Target IP	348
0x009E	iSCSI Target Port	349
0x009F	iSCSI Target Name	349
0x011F	CHAP Password	349
0x0128	CHAP User Name	349
0x0168	VLAN ID	349
0x0169	Mutual CHAP Password	349

Word Offset	Description	Reference
0x0172	FCoE Flags	350
0x0173 + 1*n, n=0...2	Reserved	350
0x0176	Target Worldwide Port Name (WWPN)	350
0x017A	Boot LUN	351
0x017B	VLAN ID	351
0x017C	Target Boot Order	351
0x017D	Reserved	351
0x017E	Target Worldwide Port Name (WWPN)	351
0x0182	Boot LUN	352
0x0183	VLAN ID	352
0x0184	Target Boot Order	352
0x0185	Reserved	352
0x0186	Target Worldwide Port Name (WWPN)	352
0x018A	Boot LUN	353
0x018B	VLAN ID	353
0x018C	Target Boot Order	353
0x018D	Reserved	353
0x018E	Target Worldwide Port Name (WWPN)	353
0x0192	Boot LUN	354
0x0193	VLAN ID	354
0x0194	Target Boot Order	354
0x0195 + 1*n, n=0...44	Reserved	354
0x01C2	iSCSI Flags	354
0x01C3 + 1*n, n=0...1	iSCSI Initiator IP	354
0x01C5 + 1*n, n=0...1	Subnet Mask	355
0x01C7 + 1*n, n=0...1	Gateway IP	355
0x01C9	iSCSI Boot LUN	355
0x01CA + 1*n, n=0...1	iSCSI Target IP	355
0x01CC	iSCSI Target Port	355
0x01CD	iSCSI Target Name	355
0x024D	CHAP Password	356
0x0256	CHAP User Name	356
0x0296	VLAN ID	356
0x0297	Mutual CHAP Password	356
0x02A0	FCoE Flags	356



Word Offset	Description	Reference
0x02A1 + 1*n, n=0...2	Reserved	356
0x02A4	Target Worldwide Port Name (WWPN)	356
0x02A8	Boot LUN	357
0x02A9	VLAN ID	357
0x02AA	Target Boot Order	357
0x02AB	Reserved	357
0x02AC	Target Worldwide Port Name (WWPN)	357
0x02B0	Boot LUN	357
0x02B1	VLAN ID	357
0x02B2	Target Boot Order	357
0x02B3	Reserved	358
0x02B4	Target Worldwide Port Name (WWPN)	358
0x02B8	Boot LUN	358
0x02B9	VLAN ID	358
0x02BA	Target Boot Order	358
0x02BB	Reserved	358
0x02BC	Target Worldwide Port Name (WWPN)	358
0x02C0	Boot LUN	358
0x02C1	VLAN ID	359
0x02C2	Target Boot Order	359
0x02C3 + 1*n, n=0...44	Reserved	359
0x02F0	iSCSI Flags	359
0x02F1 + 1*n, n=0...1	iSCSI Initiator IP	359
0x02F3 + 1*n, n=0...1	Subnet Mask	359
0x02F5 + 1*n, n=0...1	Gateway IP	359
0x02F7	iSCSI Boot LUN	360
0x02F8 + 1*n, n=0...1	iSCSI Target IP	360
0x02FA	iSCSI Target Port	360
0x02FB	iSCSI Target Name	360
0x037B	CHAP Password	360
0x0384	CHAP User Name	360
0x03C4	VLAN ID	361
0x03C5	Mutual CHAP Password	361
0x03CE	FCoE Flags	361
0x03CF + 1*n, n=0...2	Reserved	361

Word Offset	Description	Reference
0x03D2	Target Worldwide Port Name (WWPN)	361
0x03D6	Boot LUN	361
0x03D7	VLAN ID	361
0x03D8	Target Boot Order	361
0x03D9	Reserved	362
0x03DA	Target Worldwide Port Name (WWPN)	362
0x03DE	Boot LUN	362
0x03DF	VLAN ID	362
0x03E0	Target Boot Order	362
0x03E1	Reserved	362
0x03E2	Target Worldwide Port Name (WWPN)	362
0x03E6	Boot LUN	362
0x03E7	VLAN ID	363
0x03E8	Target Boot Order	363
0x03E9	Reserved	363
0x03EA	Target Worldwide Port Name (WWPN)	363
0x03EE	Boot LUN	363
0x03EF	VLAN ID	363
0x03F0	Target Boot Order	363
0x03F1 + 1*n, n=0...44	Reserved	363
0x041E	iSCSI Flags	363
0x041F + 1*n, n=0...1	iSCSI Initiator IP	364
0x0421 + 1*n, n=0...1	Subnet Mask	364
0x0423 + 1*n, n=0...1	Gateway IP	364
0x0425	iSCSI Boot LUN	364
0x0426 + 1*n, n=0...1	iSCSI Target IP	364
0x0428	iSCSI Target Port	365
0x0429	iSCSI Target Name	365
0x04A9	CHAP Password	365
0x04B2	CHAP User Name	365
0x04F2	VLAN ID	365
0x04F3	Mutual CHAP Password	365
0x04FC	FCoE Flags	366
0x04FD + 1*n, n=0...2	Reserved	366
0x0500	Target Worldwide Port Name (WWPN)	366



Word Offset	Description	Reference
0x0504	Boot LUN	366
0x0505	VLAN ID	366
0x0506	Target Boot Order	366
0x0507	Reserved	366
0x0508	Target Worldwide Port Name (WWPN)	366
0x050C	Boot LUN	367
0x050D	VLAN ID	367
0x050E	Target Boot Order	367
0x050F	Reserved	367
0x0510	Target Worldwide Port Name (WWPN)	367
0x0514	Boot LUN	367
0x0515	VLAN ID	367
0x0516	Target Boot Order	367
0x0517	Reserved	367
0x0518	Target Worldwide Port Name (WWPN)	368
0x051C	Boot LUN	368
0x051D	VLAN ID	368
0x051E	Target Boot Order	368
0x051F + 1*n, n=0...44	Reserved	368

6.2.13.1 Boot signature - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Boot Signature	0x5369	Boot Signature: 'i', 'S' (0x3569)

6.2.13.2 Block size - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Block Size		Length in: 1 Byte unit. First section -> Word: Boot Configuration Block -> Boot Signature. Last section -> Word: Boot Configuration Block -> Reserved.

6.2.13.3 Structure version - 0x0002

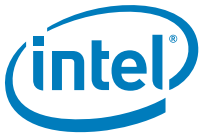
Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x00	
7:0	Structure Version	0x01	Version of this structure. Should be set to 1b.

6.2.13.4 iSCSI initiator name - 0x0003

Raw data module length: 128 words.

iSCSI initiator name.

This field is optional and built by manual input, DHCP host name, or with a MAC address.



6.2.13.5 Reserved[n] (0x0083 + 1*n, n=0...16)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0000	Reserved for future use, should be set to 0x0.

6.2.13.6 iSCSI flags - 0x0094

Bits	Field Name	Default NVM Value	Description
15:10	ARP Timeout	0x0	Timeout value for each try.
9:8	ARP Retries	0x0	Retry value.
7:6	Reserved	0x0	Reserved.
5:4	Ctrl-D Entry	0x0	Valid values are: 0x0 = Enabled. 0x3 = Disabled skip Ctrl-D entry.
3:2	Authentication Type	0x0	Valid values are: 0x0 = None. 0x1 = One way CHAP. 0x2 = Mutual CHAP.
1	Enable DHCP for iSCSI Target	0x1	Enable DHCP for getting iSCSI target information. Valid values are: 0x0 = Disabled Use static target configuration. 0x1 = Enabled Use DHCP to get target information by the option 17 root path.
0	Enable DHCP	0x1	Valid values are: 0x0 = Disabled. Use static configurations from this structure. 0x1 = Enabled. Overrides configurations retrieved from DHCP.

6.2.13.7 iSCSI initiator IP[n] (0x0095 + 1*n, n=0...1)

Initiator DHCP flag.

Not set = This field should contain the initiator IP address.

Set = This field is ignored.

Bits	Field Name	Default NVM Value	Description
15:0	iSCSI Initiator IP	0x0000	

6.2.13.8 Subnet Mask[n] (0x0097 + 1*n, n=0...1)

Initiator DHCP flag.

Not set = This field should contain the subnet mask.

Set = This field is ignored.

Bits	Field Name	Default NVM Value	Description
15:0	iSCSI Initiator IP	0x0000	

6.2.13.9 Gateway IP[n] (0x0099 + 1*n, n=0...1)

Initiator DHCP flag.

Not set = This field should contain the gateway IP address.

Set = This field is ignored.

Bits	Field Name	Default NVM Value	Description
15:0	iSCSI Initiator IP	0x0000	

6.2.13.10 iSCSI boot LUN - 0x009B

Target DHCP flag.

Not set = iSCSI target LUN number should be specified.

Set = This field is ignored.

Bits	Field Name	Default NVM Value	Description
15:0	iSCSI Boot LUN	0x0000	

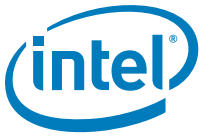
6.2.13.11 iSCSI target IP[n] (0x009C + 1*n, n=0...1)

Target DHCP flag.

Not set = IP address of iSCSI target.

Set = This field is ignored.

Bits	Field Name	Default NVM Value	Description
15:0	iSCSI Initiator IP	0x0000	



6.2.13.12 iSCSI target port - 0x009E

Bits	Field Name	Default NVM Value	Description
15:0	iSCSI Target Port	0x0CBC	Target DHCP Flag. Not set = TCP port used by iSCSI target. Default is 3260. Set = This field is ignored.

6.2.13.13 iSCSI target name - 0x009F

Raw data module length: 128 words.

Target DHCP flag.

Not set = iSCSI target name should be specified.

Set = This field is ignored.

6.2.13.14 CHAP password - 0x011F

Raw data module length: 9 words.

The minimum CHAP secret must be 12 octets and the maximum CHAP secret size is 16. The last 2 bytes are null alignment padding.

6.2.13.15 CHAP user name - 0x0128

Raw data module length: 64 words.

The user name must be non-null value and the maximum size of user name allowed is 127 characters.

6.2.13.16 VLAN ID - 0x0168

Bits	Field Name	Default NVM Value	Description
15:0	VLAN ID	0x0000	Reserved area since the function is disabled due to Microsoft restrictions. The VLAN ID includes the tag in iSCSI boot frames. A valid VLAN ID is between 1 and 4094. Zero means no VLAN tag support.

6.2.13.17 Mutual CHAP password - 0x0169

Raw data module length: 9 words.

The minimum mutual CHAP secret must be 12 octets and the maximum mutual CHAP secret size is 16. The last 2 bytes are null alignment padding.

6.2.13.18 FCoE flags - 0x0172

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	Reserved.
1	Disable FCoE Ctrl-D Menu	0x0	Valid values are: 0x0 = Enabled. FCoE Ctrl-D menu is enabled and the user can configure FCoE ports. 0x1 = Disabled. FCoE Ctrl-D menu is disabled and the user cannot configure FCoE ports.
0	Reserved	0x0	Reserved.

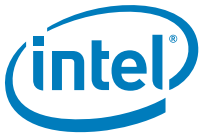
6.2.13.19 Reserved[n] (0x0173 + 1*n, n=0...2)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0000	Reserved for future use, should be set to 0x0.

6.2.13.20 Target worldwide port name (WWPN) - 0x0176

Raw data module length: 4 words.

Byte string of target WWPN.



6.2.13.21 Boot LUN - 0x017A

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

6.2.13.22 VLAN ID - 0x017B

Bits	Field Name	Default NVM Value	Description
15:0	VLAN ID	0x0000	VLAN ID for the Port. Default is 0x0.

6.2.13.23 Target boot order - 0x017C

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x00	Reserved for future use, should be set to 0x0.
7:0	Target Boot Order	0x00	Target Boot Order. Valid range is 0-4, with 0 meaning no boot order.

6.2.13.24 Reserved - 0x017D

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

6.2.13.25 Target worldwide port name (WWPN) - 0x017E

Raw data module length: 4 words.

Byte string of target WWPN.

6.2.13.26 Boot LUN - 0x0182

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

6.2.13.27 VLAN ID - 0x0183

Bits	Field Name	Default NVM Value	Description
15:0	VLAN ID	0x0000	VLAN ID for the Port. Default is 0x0.

6.2.13.28 Target boot order - 0x0184

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x00	Reserved for future use, should be set to 0x0.
7:0	Target Boot Order	0x00	Target Boot Order. Valid range is 0-4, with 0 meaning no boot order.

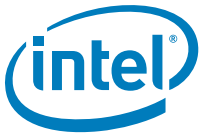
6.2.13.29 Reserved - 0x0185

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

6.2.13.30 Target worldwide port name (WWPN) - 0x0186

Raw data module length: 4 words.

Byte string of target WWPN.



6.2.13.31 Boot LUN - 0x018A

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

6.2.13.32 VLAN ID - 0x018B

Bits	Field Name	Default NVM Value	Description
15:0	VLAN ID	0x0000	VLAN ID for the Port. Default is 0x0.

6.2.13.33 Target boot order - 0x018C

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x00	Reserved for future use, should be set to 0x0.
7:0	Target Boot Order	0x00	Target Boot Order. Valid range is 0-4, with 0 meaning no boot order.

6.2.13.34 Reserved - 0x018D

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

6.2.13.35 Target worldwide port name (WWPN) - 0x018E

Raw data module length: 4 words.

Byte string of target WWPN.

6.2.13.36 Boot LUN - 0x0192

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

6.2.13.37 VLAN ID - 0x0193

Bits	Field Name	Default NVM Value	Description
15:0	VLAN ID	0x0000	VLAN ID for the Port. Default is 0x0.

6.2.13.38 Target boot order - 0x0194

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x00	Reserved for future use, should be set to 0x0.
7:0	Target Boot Order	0x00	Target Boot Order. Valid range is 0-4, with 0 meaning no boot order.

6.2.13.39 Reserved[n] (0x0195 + 1*n, n=0...44)

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

6.2.13.40 iSCSI flags - 0x01C2

For more details about the inner structure, see page 347.

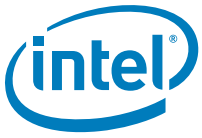
6.2.13.41 iSCSI initiator IP[n] (0x01C3 + 1*n, n=0...1)

Initiator DHCP flag.

Not set = This field should contain the initiator IP address.

Set = This field is ignored.

For more details about the inner structure, see 347.



6.2.13.42 Subnet Mask[n] (0x01C5 + 1*n, n=0...1)

Initiator DHCP flag

Not set = This field should contain the subnet mask.

Set = This field is ignored.

For more details about the inner structure, see 347.

6.2.13.43 Gateway IP[n] (0x01C7 + 1*n, n=0...1)

Initiator DHCP flag.

Not set = This field should contain the gateway IP address.

Set = This field is ignored.

For more details about the inner structure, see page 348.

6.2.13.44 iSCSI boot LUN - 0x01C9

Target DHCP flag.

Not set = iSCSI target LUN number should be specified.

Set = This field is ignored.

For more details about the inner structure, see page 348.

6.2.13.45 iSCSI target IP[n] (0x01CA + 1*n, n=0...1)

Target DHCP flag;

Not set = IP address of iSCSI target.

Set = This field is ignored.

For more details about the inner structure, see page 348.

6.2.13.46 iSCSI target port - 0x01CC

For more details about the inner structure, see page 349.

6.2.13.47 iSCSI target name - 0x01CD

Raw data module length: 128 words.

Target DHCP flag.

Not set = iSCSI target name should be specified.

Set = This field is ignored.

For more details about the inner structure, see page 349.

6.2.13.48 CHAP password - 0x024D

Raw data module length: 9 words.

The minimum CHAP secret must be 12 octets and the maximum CHAP secret size is 16. The last 2 bytes are null alignment padding.

For more details about the inner structure, see page 349.

6.2.13.49 CHAP user name - 0x0256

Raw data module length: 64 words.

The user name must be non-null value and maximum size of user name allowed is 127 characters.

For more details about the inner structure, see page 349.

6.2.13.50 VLAN ID - 0x0296

For more details about the inner structure, see page 349.

6.2.13.51 Mutual CHAP password - 0x0297

Raw data module length: 9 words.

The minimum mutual CHAP secret must be 12 octets and the maximum mutual CHAP secret size is 16. The last 2 bytes are null alignment padding.

For more details about the inner structure, see page 349.

6.2.13.52 FCoE Flags - 0x02A0

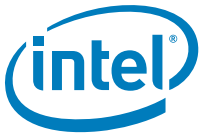
For more details about the inner structure, see page 350.

6.2.13.53 Reserved[n] (0x02A1 + 1*n, n=0...2)

For more details about the inner structure, see page 350.

6.2.13.54 Target worldwide port name (WWPN) - 0x02A4

Raw data module length: 4 words



Byte string of target WWPN.

For more details about the inner structure, see page 350.

6.2.13.55 Boot LUN - 0x02A8

For more details about the inner structure, see page 351.

6.2.13.56 VLAN ID - 0x02A9

For more details about the inner structure, see page 351.

6.2.13.57 Target boot order - 0x02AA

For more details about the inner structure, see page 351.

6.2.13.58 Reserved - 0x02AB

For more details about the inner structure, see page 351.

6.2.13.59 Target worldwide port name (WWPN) - 0x02AC

Raw data module length: 4 words.

Byte string of target WWPN.

For more details about the inner structure, see page 351.

6.2.13.60 Boot LUN - 0x02B0

For more details about the inner structure, see page 352.

6.2.13.61 VLAN ID - 0x02B1

For more details about the inner structure, see page 352.

6.2.13.62 Target boot order - 0x02B2

For more details about the inner structure, see page 352.

6.2.13.63 Reserved - 0x02B3

For more details about the inner structure, see page 352.

6.2.13.64 Target worldwide port name (WWPN) - 0x02B4

Raw data module length: 4 words.

Byte string of target WWPN.

For more details about the inner structure, see page 352.

6.2.13.65 Boot LUN - 0x02B8

For more details about the inner structure, see page 353.

6.2.13.66 VLAN ID - 0x02B9

For more details about the inner structure, see page 353.

6.2.13.67 Target boot order - 0x02BA

For more details about the inner structure, see page 353.

6.2.13.68 Reserved - 0x02BB

For more details about the inner structure, see page 353.

6.2.13.69 Target worldwide port name (WWPN) - 0x02BC

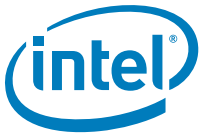
Raw data module length: 4 words.

Byte string of target WWPN.

For more details about the inner structure, see page 353.

6.2.13.70 Boot LUN - 0x02C0

For more details about the inner structure, see page 354.



6.2.13.71 VLAN ID - 0x02C1

For more details about the inner structure, see page 354.

6.2.13.72 Target boot order - 0x02C2

For more details about the inner structure, see page 354.

6.2.13.73 Reserved[n] (0x02C3 + 1*n, n=0...44)

For more details about the inner structure, see page 354.

6.2.13.74 iSCSI flags - 0x02F0

For more details about the inner structure, see page 347.

6.2.13.75 iSCSI initiator IP[n] (0x02F1 + 1*n, n=0...1)

Initiator DHCP flag.

Not set = This field should contain the initiator IP address.

Set = This field is ignored.

For more details about the inner structure, see page 347.

6.2.13.76 Subnet Mask[n] (0x02F3 + 1*n, n=0...1)

Initiator DHCP flag.

Not set = This field should contain the subnet mask.

Set = This field is ignored.

For more details about the inner structure, see page 347.

6.2.13.77 Gateway IP[n] (0x02F5 + 1*n, n=0...1)

Initiator DHCP flag.

Not set = This field should contain the gateway IP address.

Set = This field is ignored.

For more details about the inner structure, see page 348.

6.2.13.78 iSCSI boot LUN - 0x02F7

Target DHCP flag.

Not set = iSCSI target LUN number should be specified.

Set = This field is ignored.

For more details about the inner structure, see page 348.

6.2.13.79 iSCSI target IP[n] (0x02F8 + 1*n, n=0...1)

Target DHCP flag.

Not set = IP address of iSCSI target.

Set = This field is ignored.

For more details about the inner structure, see page 348.

6.2.13.80 iSCSI target port - 0x02FA

For more details about the inner structure, see page 349.

6.2.13.81 iSCSI target name - 0x02FB

Raw data module length: 128 words.

Target DHCP flag.

Not set = iSCSI target name should be specified.

Set = This field is ignored.

For more details about the inner structure, see page 349.

6.2.13.82 CHAP password - 0x037B

Raw data module length: 9 words.

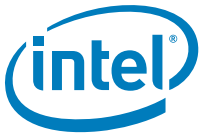
The minimum CHAP secret must be 12 octets and the maximum CHAP secret size is 16. The last 2 bytes are null alignment padding.

For more details about the inner structure, see page 349.

6.2.13.83 CHAP user name - 0x0384

Raw data module length: 64 words.

The user name must be a non-null value and the maximum size of user name allowed is 127 characters.



For more details about the inner structure, see page 349.

6.2.13.84 VLAN ID - 0x03C4

For more details about the inner structure, see page 349.

6.2.13.85 Mutual CHAP pPassword - 0x03C5

Raw data module length: 9 words.

The minimum mutual CHAP secret must be 12 octets and the maximum mutual CHAP secret size is 16. The last 2 bytes are null alignment padding.

For more details about the inner structure, see page 349.

6.2.13.86 FCoE flags - 0x03CE

For more details about the inner structure, see page 350.

6.2.13.87 Reserved[n] (0x03CF + 1*n, n=0...2)

For more details about the inner structure, see page 350.

6.2.13.88 Target worldwide port name (WWPN) - 0x03D2

Raw data module length: 4 words.

Byte string of target WWPN.

For more details about the inner structure, see page 350.

6.2.13.89 Boot LUN - 0x03D6

For more details about the inner structure, see page 351.

6.2.13.90 VLAN ID - 0x03D7

For more details about the inner structure, see page 351.

6.2.13.91 Target boot order - 0x03D8

For more details about the inner structure, see page 351.

6.2.13.92 Reserved - 0x03D9

For more details about the inner structure, see page 351.

6.2.13.93 Target worldwide port name (WWPN) - 0x03DA

Raw data module length: 4 words.

Byte string of target WWPN.

For more details about the inner structure, see page 351.

6.2.13.94 Boot LUN - 0x03DE

For more details about the inner structure, see page 352.

6.2.13.95 VLAN ID - 0x03DF

For more details about the inner structure, see page 352.

6.2.13.96 Target boot order - 0x03E0

For more details about the inner structure, see page 352.

6.2.13.97 Reserved - 0x03E1

For more details about the inner structure, see page 352.

6.2.13.98 Target worldwide port name (WWPN) - 0x03E2

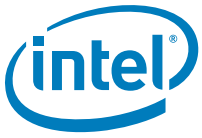
Raw data module length: 4 words

Byte string of target WWPN.

For more details about the inner structure, see page 352.

6.2.13.99 Boot LUN - 0x03E6

For more details about the inner structure, see page 353.



6.2.13.100 VLAN ID - 0x03E7

For more details about the inner structure, see page 353.

6.2.13.101 Target Boot Order - 0x03E8

For more details about the inner structure, see page 353.

6.2.13.102 Reserved - 0x03E9

For more details about the inner structure, see page 353.

6.2.13.103 Target worldwide port name (WWPN) - 0x03EA

Raw data module length: 4 words.

Byte string of target WWPN.

For more details about the inner structure, see page 353.

6.2.13.104 Boot LUN - 0x03EE

For more details about the inner structure, see page 354.

6.2.13.105 VLAN ID - 0x03EF

For more details about the inner structure, see page 354.

6.2.13.106 Target boot order - 0x03F0

For more details about the inner structure, see page 354.

6.2.13.107 Reserved[n] (0x03F1 + 1*n, n=0...44)

For more details about the inner structure, see page 354.

6.2.13.108 iSCSI flags - 0x041E

For more details about the inner structure, see page 347.

6.2.13.109 iSCSI initiator IP[n] (0x041F + 1*n, n=0...1)

Initiator DHCP flag.

Not set = This field should contain the initiator IP address.

Set = This field is ignored.

For more details about the inner structure, see page 347.

6.2.13.110 Subnet Mask[n] (0x0421 + 1*n, n=0...1)

Initiator DHCP flag

not set = This field should contain the subnet mask.

Set = This field is ignored.

For more details about the inner structure, see page 347.

6.2.13.111 Gateway IP[n] (0x0423 + 1*n, n=0...1)

Initiator DHCP flag.

not set = This field should contain the gateway IP address.

Set = This field is ignored.

For more details about the inner structure, see page 348.

6.2.13.112 iSCSI boot lUN - 0x0425

Target DHCP flag.

Not set = iSCSI target LUN number should be specified.

Set = This field is ignored.

For more details about the inner structure, see page 348.

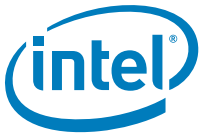
6.2.13.113 iSCSI target IP[n] (0x0426 + 1*n, n=0...1)

Target DHCP flag.

Not set = IP address of iSCSI target.

Set = This field is ignored.

For more details about the inner structure, see page 348.



6.2.13.114 iSCSI target port - 0x0428

For more details about the inner structure, see page 349

6.2.13.115 iSCSI target name - 0x0429

Raw data module length: 128 words

Target DHCP flag

Not set = iSCSI target name should be specified.

Set = This field is ignored.

For more details about the inner structure, see page 349

6.2.13.116 CHAP password - 0x04A9

Raw data module length: 9 words.

The minimum CHAP secret must be 12 octets and the maximum CHAP secret size is 16. The last 2 bytes are null alignment padding.

For more details about the inner structure, see page 349.

6.2.13.117 CHAP user name - 0x04B2

Raw data module length: 64 words.

The user name must be a non-null value and the maximum size of user name allowed is 127 characters.

For more details about the inner structure, see page 349.

6.2.13.118 VLAN ID - 0x04F2

For more details about the inner structure, see page 349.

6.2.13.119 Mutual CHAP password - 0x04F3

Raw data module length: 9 words.

The minimum mutual CHAP secret must be 12 octets and the maximum mutual CHAP secret size is 16. The last 2 bytes are null alignment padding.

For more details about the inner structure, see page 349.

6.2.13.120 FCoE flags - 0x04FC

For more details about the inner structure, see page 350.

6.2.13.121 Reserved[n] (0x04FD + 1*n, n=0...2)

For more details about the inner structure, see page 350.

6.2.13.122 Target worldwide port name (WWPN) - 0x0500

Raw data module length: 4 words.

Byte string of target WWPN.

For more details about the inner structure, see page 350.

6.2.13.123 Boot LUN - 0x0504

For more details about the inner structure, see page 351.

6.2.13.124 VLAN ID - 0x0505

For more details about the inner structure, see page 351.

6.2.13.125 Target Boot Order - 0x0506

For more details about the inner structure, see page 351.

6.2.13.126 Reserved - 0x0507

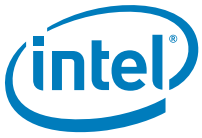
For more details about the inner structure, see page 351.

6.2.13.127 Target worldwide port name (WWPN) - 0x0508

Raw data module length: 4 words.

Byte string of target WWPN.

For more details about the inner structure, see page 351.



6.2.13.128 Boot LUN - 0x050C

For more details about the inner structure, see page 352.

6.2.13.129 VLAN ID - 0x050D

For more details about the inner structure, see page 352.

6.2.13.130 Target boot order - 0x050E

For more details about the inner structure, see page 352.

6.2.13.131 Reserved - 0x050F

For more details about the inner structure, see page 352.

6.2.13.132 Target worldwide port name (WWPN) - 0x0510

Raw data module length: 4 words.

Byte string of target WWPN.

For more details about the inner structure, see page 352.

6.2.13.133 Boot LUN - 0x0514

For more details about the inner structure, see page 353

6.2.13.134 VLAN ID - 0x0515

For more details about the inner structure, see page 353.

6.2.13.135 Target Boot Order - 0x0516

For more details about the inner structure, see page 353.

6.2.13.136 Reserved - 0x0517

For more details about the inner structure, see page 353.

6.2.13.137 Target worldwide port name (WWPN) - 0x0518

Raw data module length: 4 words.

Byte string of target WWPN.

For more details about the inner structure, see page 353.

6.2.13.138 Boot LUN - 0x051C

For more details about the inner structure, see page 354.

6.2.13.139 VLAN ID - 0x051D

For more details about the inner structure, see page 354.

6.2.13.140 Target boot order - 0x051E

For more details about the inner structure, see page 354.

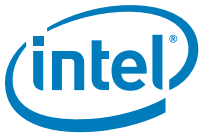
6.2.13.141 Reserved[n] (0x051F + 1*n, n=0...44)

For more details about the inner structure, see page 354.

6.2.14 PCIR registers auto-load module section summary table

Default setup to registers and internal memories that load on PCIR events.

Word Offset	Description	Reference
0x0000	Module Length	370
0x0001 - 0023	NVM contents for PFPCI_CNF	370
0x0024 - 0046	NVM contents for PFPCI_DEVID	371
0x0047 - 0069	NVM contents for PFPCI_FUNC2	372
0x006A - 008C	NVM contents for PF_VT_PFALLOC_PCIE	373
0x008D - 00AF	NVM contents for PFPCI_CLASS	374



Word Offset	Description	Reference
0x00B0 - 00B3	NVM contents for GLTPH_CTRL	375
0x00B4 - 00B8	NVM contents for GLPCI_SUBVENID	375
0x00B9 - 00BA	NVM contents for GLPCI_PWRDATA	376
0x00BB - 00BC	NVM contents for GLPCI_CNF2	376
0x00BD - 00BE	NVM contents for GLPCI_SERL	376
0x00BF - 00C0	NVM contents for GLPCI_SERH	376
0x00C1 - 00C5	NVM contents for GLPCI_CAPCTRL	376
0x00C6 - 00C7	NVM contents for GLPCI_CAPSUP	377
0x00C8 - 00C9	NVM contents for GLPCI_LINKCAP	377
0x00CA - 00CB	NVM contents for GLPCI_PMSUP	377
0x00CC - 00CD	NVM contents for GLPCI_REVID	378
0x00CE - 00CF	NVM contents for GLPCI_VFSUP	378
0x00D0 - 00D3	NVM contents for GL_PCI_DBGCTL	378
0x00D4 - 00D7	NVM contents for GLPCI_PCIERR	378
0x00D8 - 00DB	NVM contents for GLPCI_VENDORID	379
0x00DC - 00DF	NVM contents for GLGEN_PCIFCNCNT	379

6.2.14.1 Module length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 bytes unit - 1. First section -> Word: PCIR Registers Auto-load Module -> Module Length. Last section -> Word: PCIR Registers Auto-load Module -> Address Low at GLGEN_PCIFCNCNT.

6.2.14.2 PFPCI_CNF - (0x0001 - 0x0023)

6.2.14.2.1 Starting address low at PFPCI_CNF[PF] (0x0001 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPCI_CNF, for PF[0]	0xBE000	
3:0	Type	0x2	

6.2.14.2.2 Starting address high at PFPCI_CNF[PF] (0x0002 + 2*PF, PF=0)

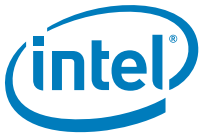
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPCI_CNF, for PF[0]		

6.2.14.2.3 Attributes at PFPCI_CNF[PF] (0x0003 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.14.2.4 Data low of PFPCI_CNF[PF] (0x0004 + 2*PF, PF=0...15)

6.2.14.2.5 Data high of PFPCI_CNF[PF] (0x0005 + 2*PF, PF=0...15)



6.2.14.3 PFPCI_DEVID - (0x0024 - 0x0046)

6.2.14.3.1 Starting address low at PFPCI_DEVID[PF] (0x0024 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPCI_DEVID, for PF[0]	0xBE080	
3:0	Type	0x2	

6.2.14.3.2 Starting address high at PFPCI_DEVID[PF] (0x0025 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPCI_DEVID, for PF[0]		

6.2.14.3.3 Attributes at PFPCI_DEVID[PF] (0x0026 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.14.3.4 Data low of PFPCI_DEVID[PF] (0x0027 + 2*PF, PF=0...15)

6.2.14.3.5 Data high of PFPCI_DEVID[PF] (0x0028 + 2*PF, PF=0...15)

6.2.14.4 PFPCI_FUNC2 - (0x0047 - 0x0069)

6.2.14.4.1 Starting address low at PFPCI_FUNC2[PF] (0x0047 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPCI_FUNC2, for PF[0]	0xBE180	
3:0	Type	0x2	

6.2.14.4.2 Starting address high at PFPCI_FUNC2[PF] (0x0048 + 2*PF, PF=0)

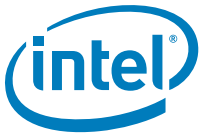
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPCI_FUNC2, for PF[0]		

6.2.14.4.3 Attributes at PFPCI_FUNC2[PF] (0x0049 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.14.4.4 Data low of PFPCI_FUNC2[PF] (0x004A + 2*PF, PF=0...15)

6.2.14.4.5 Data high of PFPCI_FUNC2[PF] (0x004B + 2*PF, PF=0...15)



6.2.14.5 PF_VT_PFALLOC_PCIE - (0x006A - 0x008C)

6.2.14.5.1 Starting address low at PF_VT_PFALLOC_PCIE[PF] (0x006A + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PF_VT_PFALLOC_PCIE, for PF[0]	0xBE380	
3:0	Type	0x2	

6.2.14.5.2 Starting address high at PF_VT_PFALLOC_PCIE[PF] (0x006B + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PF_VT_PFALLOC_PCIE, for PF[0]		

6.2.14.5.3 Attributes at PF_VT_PFALLOC_PCIE[PF] (0x006C + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.14.5.4 Data low of PF_VT_PFALLOC_PCIE[PF] (0x006D + 2*PF, PF=0…15)

6.2.14.5.5 Data high of PF_VT_PFALLOC_PCIE[PF] (0x006E + 2*PF, PF=0…15)

6.2.14.6 PFPCI_CLASS - (0x008D - 0x00AF)

6.2.14.6.1 Starting address low at PFPCI_CLASS[PF] (0x008D + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPCI_CLASS, for PF[0]	0xBE400	
3:0	Type	0x2	

6.2.14.6.2 Starting address high at PFPCI_CLASS[PF] (0x008E + 2*PF, PF=0)

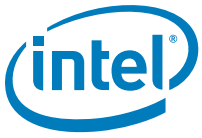
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPCI_CLASS, for PF[0]		

6.2.14.6.3 Attributes at PFPCI_CLASS[PF] (0x008F + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.14.6.4 Data low of PFPCI_CLASS[PF] (0x0090 + 2*PF, PF=0...15)

6.2.14.6.5 Data high of PFPCI_CLASS[PF] (0x0091 + 2*PF, PF=0...15)



6.2.14.7 GLTPH_CTRL - (0x00B0 - 0x00B3)

6.2.14.7.1 Address high at GLTPH_CTRL - 0x00B1

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLTPH_CTRL		

6.2.14.7.2 Data low of GLTPH_CTRL - 0x00B2

6.2.14.7.3 Data high of GLTPH_CTRL - 0x00B3

6.2.14.8 GLPCI_SUBVENID - (0x00B4 - 0x00B8)

6.2.14.8.1 Starting address low at GLPCI_SUBVENID - 0x00B4

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_SUBVENID	0xBE48C	
3:0	Type	0x2	

6.2.14.8.2 Starting address high at GLPCI_SUBVENID - 0x00B5

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_SUBVENID		

6.2.14.8.3 Attributes at GLPCI_SUBVENID - 0x00B6

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x5	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.14.8.4 Data low of GLPCI_SUBVENID - 0x00B7

6.2.14.8.5 Data high of GLPCI_SUBVENID - 0x00B8

6.2.14.9 GLPCI_PWRDATA - (0x00B9 - 0x00BA)

6.2.14.9.1 Data low of GLPCI_PWRDATA - 0x00B9

6.2.14.9.2 Data high of GLPCI_PWRDATA - 0x00BA

6.2.14.10 GLPCI_CNF2 - (0x00BB - 0x00BC)

6.2.14.10.1 Data low of GLPCI_CNF2 - 0x00BB

6.2.14.10.2 Data high of GLPCI_CNF2 - 0x00BC

6.2.14.11 GLPCI_SERL - (0x00BD - 0x00BE)

6.2.14.11.1 Data low of GLPCI_SERL - 0x00BD

6.2.14.11.2 Data high of GLPCI_SERL - 0x00BE

6.2.14.12 GLPCI_SERH - (0x00BF - 0x00C0)

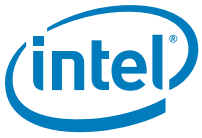
6.2.14.12.1 Data low of GLPCI_SERH - 0x00BF

6.2.14.12.2 Data high of GLPCI_SERH - 0x00C0

6.2.14.13 GLPCI_CAPCTRL - (0x00C1 - 0x00C5)

6.2.14.13.1 Starting address low at GLPCI_CAPCTRL - 0x00C1

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_CAPCTRL	0xBE4A4	
3:0	Type	0x2	



6.2.14.13.2 Starting address high at GLPCI_CAPCTRL - 0x00C2

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_CAPCTRL		

6.2.14.13.3 Attributes at GLPCI_CAPCTRL - 0x00C3

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x6	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.14.13.4 Data low of GLPCI_CAPCTRL - 0x00C4

6.2.14.13.5 Data high of GLPCI_CAPCTRL - 0x00C5

6.2.14.14 GLPCI_CAPSUP - (0x00C6 - 0x00C7)

6.2.14.14.1 Data low of GLPCI_CAPSUP - 0x00C6

6.2.14.14.2 Data high of GLPCI_CAPSUP - 0x00C7

6.2.14.15 GLPCI_LINKCAP - (0x00C8 - 0x00C9)

6.2.14.15.1 Data low of GLPCI_LINKCAP - 0x00C8

6.2.14.15.2 Data high of GLPCI_LINKCAP - 0x00C9

6.2.14.16 GLPCI_PMSUP - (0x00CA - 0x00CB)

6.2.14.16.1 Data low of GLPCI_PMSUP - 0x00CA

6.2.14.16.2 Data high of GLPCI_PMSUP - 0x00CB

6.2.14.17 GLPCI_REVID - (0x00CC - 0x00CD)

6.2.14.17.1 Data low of GLPCI_REVID - 0x00CC

6.2.14.17.2 Data high of GLPCI_REVID - 0x00CD

6.2.14.18 GLPCI_VFSUP - (0x00CE - 0x00CF)

6.2.14.18.1 Data low of GLPCI_VFSUP - 0x00CE

6.2.14.18.2 Data high of GLPCI_VFSUP - 0x00CF

6.2.14.19 GL_PCI_DBGCTL - (0x00D0 - 0x00D3)

6.2.14.19.1 Address high at GL_PCI_DBGCTL - 0x00D1

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_PCI_DBGCTL		

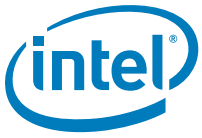
6.2.14.19.2 Data low of GL_PCI_DBGCTL - 0x00D2

6.2.14.19.3 Data high of GL_PCI_DBGCTL - 0x00D3

6.2.14.20 GLPCI_PCIERR - (0x00D4 - 0x00D7)

6.2.14.20.1 Address low at GLPCI_PCIERR - 0x00D4

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_PCIERR	0xBE4FC	
3:0	Type	0x1	



6.2.14.20.2 Address high at GLPCI_PCIERR - 0x00D5

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_PCIERR		

6.2.14.20.3 Data low of GLPCI_PCIERR - 0x00D6

6.2.14.20.4 Data high of GLPCI_PCIERR - 0x00D7

6.2.14.21 GLPCI_VENDORID - (0x00D8 - 0x00DB)

6.2.14.21.1 Address low at GLPCI_VENDORID - 0x00D8

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_VENDORID	0xBE518	
3:0	Type	0x1	

6.2.14.21.2 Address high at GLPCI_VENDORID - 0x00D9

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_VENDORID		

6.2.14.21.3 Data low of GLPCI_VENDORID - 0x00DA

6.2.14.21.4 Data high of GLPCI_VENDORID - 0x00DB

6.2.14.22 GLGEN_PCIFCNCNT - (0x00DC - 0x00DF)

6.2.14.22.1 Address low at GLGEN_PCIFCNCNT - 0x00DC

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLGEN_PCIFCNCNT	0x1C0AB4	
3:0	Type	0x1	

6.2.14.22.2 Address high at GLGEN_PCIFCNCNT - 0x00DD

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_PCIFCNCNT		

6.2.14.22.3 Data low of GLGEN_PCIFCNCNT - 0x00DE

6.2.14.22.4 Data high of GLGEN_PCIFCNCNT - 0x00DF

6.2.15 PCIR registers auto-load module section summary table

Word Offset	Description	Reference
0x0000	LCB Port Address Low	380
0x0001	LCB Port Address High	380
0x0002	LCB Attributes	380
0x0003	LCB Data	380

6.2.15.1 LCB attributes - 0x0002

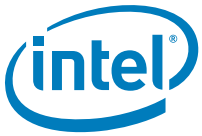
Part of a Type 4 structure to load the PCIe LCB unit.

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x060	
4:3	Skip	0x00	

6.2.15.2 LCB data - 0x0003

Raw data module length: 256 words.

Part of a Type 4 structure to load the PCIe LCB unit.



6.2.16 POR registers auto-load module section summary table

Default setup to registers that load on POR events.

Word Offset	Description	Reference
0x0000	Module Length	382
0x0001 - 0023	NVM contents for PFPM_WUC	382
0x0024 - 0027	NVM contents for GLPM_WUMC	383
0x0028 - 0066	NVM contents for GLGEN_GPIO_CTL	383
0x0067 - 0068	NVM contents for GLGEN_LED_CTL	384
0x0069 - 0073	NVM contents for GLGEN_I2CPARAMS	384
0x0074 - 007E	NVM contents for GLGEN_MDIO_I2C_SEL	385
0x007F - 0086	NVM contents for GLGEN_MDIO_CTRL	386
0x0087 - 008A	NVM contents for GL_MNG_HWARB_CTRL	386
0x008B - 008E	NVM contents for GLNVM_EMPRQ	387
0x008F - 00B1	NVM contents for PFGEN_PORTNUM_CAR	387
0x00B2 - 00D4	NVM contents for PFPM_APM	388
0x00D5 - 00DF	NVM contents for PRTGEN_CNF	389
0x00E0 - 00EA	NVM contents for PRTPM_GC	390
0x00EB - 00F5	NVM contents for PRTGEN_CNF2	391
0x00F6 - 00FA	NVM contents for GLGEN_RSTCTL	392
0x00FB - 00FC	NVM contents for GLGEN_CLKSTAT	392
0x00FD - 0101	NVM contents for GLMNG_WD_ENA	393
0x0102 - 0103	NVM contents for GLPCI_CLKCTL	393
0x0106 - 0107	NVM contents for GLGEN_MISC_CONFIG	393

Word Offset	Description	Reference
0x0108 - 012A	NVM contents for PFPCI_FUNC	394
0x012B - 012E	NVM contents for GLPCI_CNF	395
0x012F - 0132	NVM contents for GLPCI_SPARE1	Section 6.2.16.23

6.2.16.1 Module length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 bytes unit - 1. First Section -> Word: POR Registers Auto-load Module -> Module Length. Last Section -> Word: POR Registers Auto-load Module -> Address Low at GLPCI_SPARE1.

6.2.16.2 PFPM_WUC - (0x0001 - 0023)

6.2.16.2.1 Starting address low at PFPM_WUC[PF] (0x0001 + 2*PF, PF=0)

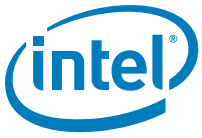
Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPM_WUC, for PF[0]	0x6B200	
3:0	Type	0x2	

6.2.16.2.2 Starting address high at PFPM_WUC[PF] (0x0002 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPM_WUC, for PF[0]		

6.2.16.2.3 Attributes at PFPM_WUC[PF] (0x0003 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	



6.2.16.2.4 Data low of PFPM_WUC[PF] (0x0004 + 2*PF, PF=0...15)

6.2.16.2.5 Data high of PFPM_WUC[PF] (0x0005 + 2*PF, PF=0...15)

6.2.16.3 GLPM_WUMC - (0x0024 - 0x0027)

6.2.16.3.1 Address low at GLPM_WUMC - 0x0024

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPM_WUMC	0x6C800	
3:0	Type	0x1	

6.2.16.3.2 Address high at GLPM_WUMC - 0x0025

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPM_WUMC		

6.2.16.3.3 Data low of GLPM_WUMC - 0x0026

6.2.16.3.4 Data high of GLPM_WUMC - 0x0027

6.2.16.4 GLGEN_GPIO_CTL - (0x0028 - 0x0066)

6.2.16.4.1 Starting address low at GLGEN_GPIO_CTL[n] (0x0028)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLGEN_GPIO_CTL	0x88100	
3:0	Type	0x2	

6.2.16.4.2 Starting address high at GLGEN_GPIO_CTL[n] (0x0029)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_GPIO_CTL		

6.2.16.4.3 Attributes at GLGEN_GPIO_CTL[n] (0x002A)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x1F	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.16.4.4 Data low of GLGEN_GPIO_CTL[n] (0x002B + 2*n, n=0...29)

6.2.16.4.5 Data high of GLGEN_GPIO_CTL[n] (0x002C + 2*n, n=0...29)

6.2.16.5 GLGEN_LED_CTL - (0x0067 - 0x0068)

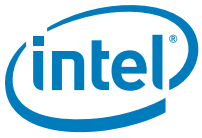
6.2.16.5.1 Data low of GLGEN_LED_CTL - 0x0067

6.2.16.5.2 Data ligh of GLGEN_LED_CTL - 0x0068

6.2.16.6 GLGEN_I2CPARAMS - (0x0069 - 0x0073)

6.2.16.6.1 Starting address low at GLGEN_I2CPARAMS[n] (0x0069)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLGEN_I2CPARAMS	0x881AC	
3:0	Type	0x2	



6.2.16.6.2 Starting address high at GLGEN_I2CPARAMS[n] (0x006A)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_I2CPARAMS		

6.2.16.6.3 Attributes at GLGEN_I2CPARAMS[n] (0x006B)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.16.6.4 Data low of GLGEN_I2CPARAMS[n] (0x006C + 2*n, n=0...3)

6.2.16.6.5 Data high of GLGEN_I2CPARAMS[n] (0x006D + 2*n, n=0...3)

6.2.16.7 GLGEN_MDIO_I2C_SEL - (0x0074 - 0x007E)

6.2.16.7.1 Starting address low at GLGEN_MDIO_I2C_SEL[n] (0x0074)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLGEN_MDIO_I2C_SEL	0x881C0	
3:0	Type	0x2	

6.2.16.7.2 Starting address high at GLGEN_MDIO_I2C_SEL[n] (0x0075)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_MDIO_I2C_SEL		

6.2.16.7.3 Attributes at GLGEN_MDIO_I2C_SEL[n] (0x0076)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.16.7.4 Data low of GLGEN_MDIO_I2C_SEL[n] (0x0077 + 2*n, n=0...3)

6.2.16.7.5 Data high of GLGEN_MDIO_I2C_SEL[n] (0x0078 + 2*n, n=0...3)

6.2.16.8 GLGEN_MDIO_CTRL - (0x007F - 0x0086)

6.2.16.8.1 Data low of GLGEN_MDIO_CTRL[n] (0x007F + 2*n, n=0...3)

6.2.16.8.2 Data high of GLGEN_MDIO_CTRL[n] (0x0080 + 2*n, n=0...3)

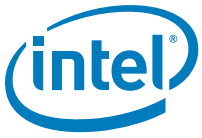
6.2.16.9 GL_MNG_HWARB_CTRL - (0x0087 - 0x008A)

6.2.16.9.1 Address low at GL_MNG_HWARB_CTRL - 0x0087

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_MNG_HWARB_CTRL	0xB6130	
3:0	Type	0x1	

6.2.16.9.2 Address high at GL_MNG_HWARB_CTRL - 0x0088

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_MNG_HWARB_CTRL		



6.2.16.9.3 Data low of GL_MNG_HWARB_CTRL - 0x0089

6.2.16.9.4 Data high of GL_MNG_HWARB_CTRL - 0x008A

6.2.16.10 GLNVM_EMPRQ - (0x008B - 0x008E)

6.2.16.10.1 Address high at GLNVM_EMPRQ - 0x008C

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLNVM_EMPRQ		

6.2.16.10.2 Data low of GLNVM_EMPRQ - 0x008D

6.2.16.10.3 Data high of GLNVM_EMPRQ - 0x008E

6.2.16.11 PFGEN_PORTNUM_CAR - (0x008F - 0x00B1)

6.2.16.11.1 Starting address low at PFGEN_PORTNUM_CAR[PF] (0x008F + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFGEN_PORTNUM_CAR, for PF[0]	0xB8000	
3:0	Type	0x2	

6.2.16.11.2 Starting address high at PFGEN_PORTNUM_CAR[PF] (0x0090 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFGEN_PORTNUM_CAR, for PF[0]		

6.2.16.11.3 Attributes at PFGEN_PORTNUM_CAR[PF] (0x0091 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.16.11.4 Data Low of PFGEN_PORTNUM_CAR[PF] (0x0092 + 2*PF, PF=0...15)

6.2.16.11.5 Data High of PFGEN_PORTNUM_CAR[PF] (0x0093 + 2*PF, PF=0...15)

6.2.16.12 PFPM_APM - (0x00B2 - 0x00D4)

6.2.16.12.1 Starting address low at PFPM_APM[PF] (0x00B2 + 2*PF, PF=0)

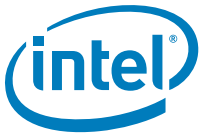
Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPM_APM, for PF[0]	0xB8080	
3:0	Type	0x2	

6.2.16.12.2 Starting address high at PFPM_APM[PF] (0x00B3 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPM_APM, for PF[0]		

6.2.16.12.3 Attributes at PFPM_APM[PF] (0x00B4 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	



6.2.16.12.4 Data low of PFPM_APM[PF] (0x00B5 + 2*PF, PF=0...15)

6.2.16.12.5 Data high of PFPM_APM[PF] (0x00B6 + 2*PF, PF=0...15)

6.2.16.13 PRTGEN_CNF - (0x00D5 - 0x00DF)

6.2.16.13.1 Starting address low at PRTGEN_CNF[PRT] (0x00D5 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTGEN_CNF, for PRT[0]	0xB8120	
3:0	Type	0x2	

6.2.16.13.2 Starting address high at PRTGEN_CNF[PRT] (0x00D6 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTGEN_CNF, for PRT[0]		

6.2.16.13.3 Attributes at PRTGEN_CNF[PRT] (0x00D7 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.16.13.4 Data low of PRTGEN_CNF[PRT] (0x00D8 + 2*PRT, PRT=0...3)

6.2.16.13.5 Data high of PRTGEN_CNF[PRT] (0x00D9 + 2*PRT, PRT=0...3)

6.2.16.14 PRTPM_GC - (0x00E0 - 0x00EA)

6.2.16.14.1 Starting address low at PRTPM_GC[PRT] (0x00E0 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTPM_GC, for PRT[0]	0xB8140	
3:0	Type	0x2	

6.2.16.14.2 Starting address high at PRTPM_GC[PRT] (0x00E1 + 2*PRT, PRT=0)

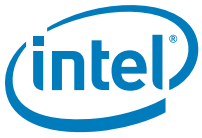
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTPM_GC, for PRT[0]		

6.2.16.14.3 Attributes at PRTPM_GC[PRT] (0x00E2 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.16.14.4 Data low of PRTPM_GC[PRT] (0x00E3 + 2*PRT, PRT=0...3)

6.2.16.14.5 Data high of PRTPM_GC[PRT] (0x00E4 + 2*PRT, PRT=0...3)



6.2.16.15 PRTGEN_CNF2 - (0x00EB - 0x00F5)

6.2.16.15.1 Starting address low at PRTGEN_CNF2[PRT] (0x00EB + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTGEN_CNF2, for PRT[0]	0xB8160	
3:0	Type	0x2	

6.2.16.15.2 Starting address high at PRTGEN_CNF2[PRT] (0x00EC + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTGEN_CNF2, for PRT[0]		

6.2.16.15.3 Attributes at PRTGEN_CNF2[PRT] (0x00ED + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.16.15.4 Data low of PRTGEN_CNF2[PRT] (0x00EE + 2*PRT, PRT=0...3)

6.2.16.15.5 Data high of PRTGEN_CNF2[PRT] (0x00EF + 2*PRT, PRT=0...3)

6.2.16.16 GLGEN_RSTCTL - (0x00F6 - 0x00FA)

6.2.16.16.1 Starting address low at GLGEN_RSTCTL - 0x00F6

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLGEN_RSTCTL	0xB8180	
3:0	Type	0x2	

6.2.16.16.2 Starting address high at GLGEN_RSTCTL - 0x00F7

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_RSTCTL		

6.2.16.16.3 Attributes at GLGEN_RSTCTL - 0x00F8

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

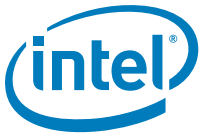
6.2.16.16.4 Data low of GLGEN_RSTCTL - 0x00F9

6.2.16.16.5 Data high of GLGEN_RSTCTL - 0x00FA

6.2.16.17 GLGEN_CLKSTAT - (0x00FB - 0x00FC)

6.2.16.17.1 Data low of GLGEN_CLKSTAT - 0x00FB

6.2.16.17.2 Data high of GLGEN_CLKSTAT - 0x00FC



6.2.16.18 GLMNG_WD_ENA - (0x00FD - 0x0101)

6.2.16.18.1 Starting address low at GLMNG_WD_ENA - 0x00FD

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLMNG_WD_ENA	0xB8198	
3:0	Type	0x2	

6.2.16.18.2 Starting address high at GLMNG_WD_ENA - 0x00FE

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLMNG_WD_ENA		

6.2.16.18.3 Attributes at GLMNG_WD_ENA - 0x00FF

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.16.18.4 Data low of GLMNG_WD_ENA - 0x0100

6.2.16.18.5 Data high of GLMNG_WD_ENA - 0x0101

6.2.16.19 GLPCI_CLKCTL - (0x0102 - 0x0103)

6.2.16.19.1 Data low of GLPCI_CLKCTL - 0x0102

6.2.16.19.2 Data high of GLPCI_CLKCTL - 0x0103

6.2.16.20 GLGEN_MISC_CONFIG - (0x0106 - 0x0107)

6.2.16.20.1 Data low of GLGEN_MISC_CONFIG - 0x0106

6.2.16.20.2 Data high of GLGEN_MISC_CONFIG - 0x0107

6.2.16.21 PFPCI_FUNC - (0x0108 - 0x 012A)

6.2.16.21.1 Starting address low at PFPCI_FUNC[PF] (0x0108 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPCI_FUNC, for PF[0]	0xBE200	
3:0	Type	0x2	

6.2.16.21.2 Starting address high at PFPCI_FUNC[PF] (0x0109 + 2*PF, PF=0)

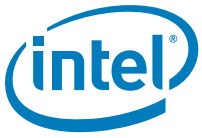
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPCI_FUNC, for PF[0]		

6.2.16.21.3 Attributes at PFPCI_FUNC[PF] (0x010A + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.16.21.4 Data low of PFPCI_FUNC[PF] (0x010B + 2*PF, PF=0...15)

6.2.16.21.5 Data high of PFPCI_FUNC[PF] (0x010C + 2*PF, PF=0...15)



6.2.16.22 GLPCI_CNF - (0x012B - 012E)

6.2.16.22.1 Address low at GLPCI_CNF - 0x012B

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_CNF	0xBE4C0	
3:0	Type	0x1	

6.2.16.22.2 Address high at GLPCI_CNF - 0x012C

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_CNF		

6.2.16.22.3 Data low of GLPCI_CNF - 0x012D

6.2.16.22.4 Data high of GLPCI_CNF - 0x012E

6.2.16.23 GLPCI_SPARE1 - (0x012F - 0x0132)

6.2.16.23.1 Address high at GLPCI_SPARE1 - 0x0130

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_SPARE1		

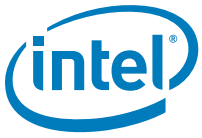
6.2.16.23.2 Data low of GLPCI_SPARE1 - 0x0131

6.2.16.23.3 Data high of GLPCI_SPARE1 - 0x0132

6.2.17 CORER registers auto-load module section summary table

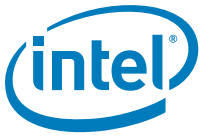
Default setup to registers and internal memories that load on CORER events.

Word Offset	Description	Reference
0x0000	Module Length	401
0x0001 - 0023	NVM contents for PFGEN_PORTMDIO_NUM	401
0x0024 - 0027	NVM contents for GLINT_CTL	402
0x0028 - 002B	NVM contents for GLGEN_PCIFCNCNT_INT	402
0x002C - 0036	NVM contents for PRTDCB_TDPUC	402
0x0037 - 0049	NVM contents for GL_SWT_L2TAGTXIB	403
0x004A - 004B	NVM contents for GLLAN_TSOMSK_F	404
0x004C - 004D	NVM contents for GLLAN_TSOMSK_M	404
0x004E - 004F	NVM contents for GLLAN_TSOMSK_L	404
0x0050 - 0054	NVM contents for GLLAN_TCTL_1	405
0x0055 - 0056	NVM contents for GL_MDCK_TDAT	405
0x0057 - 0069	NVM contents for GL_SWT_L2TAGRXEB	406
0x006A - 006D	NVM contents for GLDCB_RLLPC	406
0x006E - 0078	NVM contents for PRTDCB_GENC	407
0x0079 - 007C	NVM contents for GLDCB_GENC	408
0x007D - 0080	NVM contents for ECC_ENA	408
0x0081 - 008B	NVM contents for PRTDCB_TFLLPC	409
0x008C - 0096	NVM contents for PRTDCB_TDPMC	410
0x0097 - 00A1	NVM contents for PRTDCB_TFPFCC	411
0x00A2 - 00A5	NVM contents for GLDCB_TGENC_TLPM	412



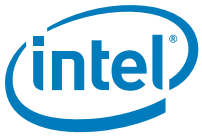
Word Offset	Description	Reference
0x00A6 - 00B0	NVM contents for PRTDCB_TCPMC	412
0x00B1 - 00BB	NVM contents for PRTDCB_TCPFCPC	413
0x00BC - 00C6	NVM contents for PRTDCB_TCPFCTCC	414
0x00C7 - 00CA	NVM contents for GLDCB_TGENC_TUPM	415
0x00CB - 00CE	NVM contents for GLFOC_CACHE_CTL	415
0x00CF - 00D9	NVM contents for PRTRPB_SPS	415
0x00DA - 00DE	NVM contents for GLRPB_PHW	416
0x00DF - 00E0	NVM contents for GLRPB_PLW	417
0x00E1 - 00EB	NVM contents for PRTDCB_TCLLPC	417
0x00EC - 00EF	NVM contents for GLLAN_TCTL_2	418
0x00F0 - 00F3	NVM contents for GLSCD_QUANTA	418
0x00F4 - 00F7	NVM contents for GLSCD_LLPREALTHRESH	419
0x00F8 - 00FC	NVM contents for GLSCD_CREDITSPERQUANTA	419
0x00FD - 00FE	NVM contents for GLSCD_BWLCREDUPDATE	420
0x00FF - 0100	NVM contents for GLSCD_BWLLINESPERARB	420
0x0101 - 0104	NVM contents for GLSCD_LANTCBCMDS	421
0x0105 - 0127	NVM contents for GLHMC_SDPART	421
0x0128 - 012B	NVM contents for GLHMC_PMATCFG	422
0x012C - 0130	NVM contents for GLLAN_TCTL_0	422
0x0131 - 0132	NVM contents for GL_MDCK_TCMD	423
0x0133 - 0137	NVM contents for GLTLAN_MIN_MAX_PKT	423
0x0138 - 0139	NVM contents for GLTLAN_MIN_MAX_MSS	424
0x013A - 013D	NVM contents for GLCM_LANCONFIG	424

Word Offset	Description	Reference
0x013E - 0195	NVM contents for PRTDCB_RETSTCC	424
0x0196 - 01A0	NVM contents for PRTDCB_RPPMC	425
0x01A1 - 01AB	NVM contents for PRTDCB_RETSC	426
0x01AC - 01B0	NVM contents for GLDCB_RSPMC	427
0x01B1 - 01B2	NVM contents for GLDCB_RPRRD0	427
0x01B3 - 01B4	NVM contents for GLDCB_RPRRD1	427
0x01B5 - 01B6	NVM contents for GLDCB_RMPMC	428
0x01B7 - 01BB	NVM contents for GLLAN_RCTL_1	428
0x01BC - 01BD	NVM contents for GLFCOE_RLANCTL	428
0x01BE - 01BF	NVM contents for GL_MDCK_RX	429
0x01C0 - 01E2	NVM contents for GLLAN_PF_RECIPE	429
0x01E3 - 0205	NVM contents for PFLAN_QALLOC	430
0x0206 - 0228	NVM contents for PFGEN_PORTNUM	431
0x0229 - 024B	NVM contents for PF_VT_PFALLOC	432
0x024C - 024F	NVM contents for GLANL_PRE_LY2	433
0x0250 - 0272	NVM contents for GLANL_L2ULP	433
0x0273 - 0274	NVM contents for GLGEN_DUAL40	433
0x0275 - 0284	NVM contents for GL_SWT_L2TAGCTRL	434
0x0285 - 028F	NVM contents for PRT_L2TAGSEN	434
0x0290 - 02B2	NVM contents for PFQF_FDALLOC	435
0x02B3 - 02BD	NVM contents for PRT_MSCCNT	436
0x02BE - 02C8	NVM contents for PRT_SWT_BSCCNT	437
0x02C9 - 02D3	NVM contents for PRT_SWT_BSCTRH	438



Word Offset	Description	Reference
0x02D4 - 02DE	NVM contents for PRT_SWT_SCBI	439
0x02DF - 02E9	NVM contents for PRT_SWT_SCTC	440
0x02EA - 02F4	NVM contents for PRTQF_CTL_0	441
0x02F5 - 03F7	NVM contents for GLQF_PTYPE	442
0x03F8 - 04FA	NVM contents for GLQF_PTYPE_ENA	442
0x0541 - 0547	NVM contents for GLQF_FDENA	443
0x0548 - 054B	NVM contents for GL_RXA_CFG	444
0x054C - 054F	NVM contents for GL_SWR_DP	444
0x0550 - 0553	NVM contents for GLFCOE_RCTL	445
0x0554 - 0557	NVM contents for GLFCOE_RSOF	445
0x0558 - 055B	NVM contents for GLQF_CTL	446
0x055C - 0566	NVM contents for PRT_SWR_PM_THR	446
0x0567 - 0583	NVM contents for GLQF_HKEY	447
0x0584	DPU_IMEM Port Address Low	448
0x0585	DPU_IMEM Port Address High	448
0x0586	DPU_IMEM Attributes	448
0x0587	DPU_IMEM Indirect Address Low	448
0x0588	DPU_IMEM Indirect Address High	448
0x0589	DPU_IMEM Data	448
0x2589	DPU_RECIPE_ADDRESS Port Address Low	448
0x258A	DPU_RECIPE_ADDRESS Port Address High	448
0x258B	DPU_RECIPE_ADDRESS Attributes	448
0x258C	DPU_RECIPE_ADDRESS Indirect Address Low	448
0x258D	DPU_RECIPE_ADDRESS Indirect Address High	448
0x258E	DPU_RECIPE_ADDRESS Data	448
0x278E	DPU_RECIPE_CAM Port Address Low	448
0x278F	DPU_RECIPE_CAM Port Address High	448
0x2790	DPU_RECIPE_CAM Attributes	448
0x2791	DPU_RECIPE_CAM Indirect Address Low	449
0x2792	DPU_RECIPE_CAM Indirect Address High	449

Word Offset	Description	Reference
0x2793	DPU_RECIPE_CAM Data	449
0x2B93	DPU_RECIPE_MASK Port Address Low	449
0x2B94	DPU_RECIPE_MASK Port Address High	449
0x2B95	DPU_RECIPE_MASK Attributes	449
0x2B96	DPU_RECIPE_MASK Indirect Address Low	449
0x2B97	DPU_RECIPE_MASK Indirect Address High	449
0x2B98	DPU_RECIPE_MASK Data	449
0x2BB8	ANA_IMEM Port Address Low	449
0x2BB9	ANA_IMEM Port Address High	449
0x2BBA	ANA_IMEM Attributes	449
0x2BBB	ANA_IMEM Indirect Address Low	449
0x2BBC	ANA_IMEM Indirect Address High	449
0x2BBD	ANA_IMEM Data	449
0x2C5D	ANA_NH Port Address Low	449
0x2C5E	ANA_NH Port Address High	449
0x2C5F	ANA_NH Attributes	449
0x2C60	ANA_NH Indirect Address Low	450
0x2C61	ANA_NH Indirect Address High	450
0x2C62	ANA_NH Data	450
0x2CB2	ANA_SKIP Port Address Low	450
0x2CB3	ANA_SKIP Port Address High	450
0x2CB4	ANA_SKIP Attributes	450
0x2CB5	ANA_SKIP Indirect Address Low	450
0x2CB6	ANA_SKIP Indirect Address High	450
0x2CB7	ANA_SKIP Data	450
0x2D07	ANA_REPLACE Port Address Low	450
0x2D08	ANA_REPLACE Port Address High	450
0x2D09	ANA_REPLACE Attributes	450
0x2D0A	ANA_REPLACE Indirect Address Low	450
0x2D0B	ANA_REPLACE Indirect Address High	450
0x2D0C	ANA_REPLACE Data	450
0x2D5C	ANA_MERGE Port Address Low	451
0x2D5D	ANA_MERGE Port Address High	451
0x2D5E	ANA_MERGE Attributes	451
0x2D5F	ANA_MERGE Indirect Address Low	451
0x2D60	ANA_MERGE Indirect Address High	451
0x2D61	ANA_MERGE Data	451



6.2.17.1 Module length - 0x0000

Bits	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.

6.2.17.2 PFGEN_PORTMDIO_NUM - (0x0001 - 0x0023)

6.2.17.2.1 Starting address low at PFGEN_PORTMDIO_NUM[PF] (0x0001 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFGEN_PORTMDIO_NUM, for PF[0]	0x3F100	
3:0	Type	0x2	

6.2.17.2.2 Starting address high at PFGEN_PORTMDIO_NUM[PF] (0x0002 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFGEN_PORTMDIO_NUM, for PF[0]		

6.2.17.2.3 Attributes at PFGEN_PORTMDIO_NUM[PF] (0x0003 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.2.4 Data low of PFGEN_PORTMDIO_NUM[PF] (0x0004 + 2*PF, PF=0...15)

6.2.17.2.5 Data high of PFGEN_PORTMDIO_NUM[PF] (0x0005 + 2*PF, PF=0...15)

6.2.17.3 GLINT_CTL - (0x0024 - 0x0027)

6.2.17.3.1 Address high at GLINT_CTL - 0x0025

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLINT_CTL		

6.2.17.3.2 Data low of GLINT_CTL - 0x0026

6.2.17.3.3 Data high of GLINT_CTL - 0x0027

6.2.17.4 GLGEN_PCIFCNCNT_INT - (0x0028 - 0x002B)

6.2.17.4.1 Address high at GLGEN_PCIFCNCNT_INT - 0x0029

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_PCIFCNCNT_INT		

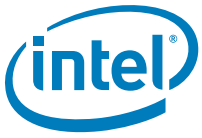
6.2.17.4.2 Data low of GLGEN_PCIFCNCNT_INT - 0x002A

6.2.17.4.3 Data high of GLGEN_PCIFCNCNT_INT - 0x002B

6.2.17.5 PRTDCB_TDPUC - (0x002C - 0x0036)

6.2.17.5.1 Starting address low at PRTDCB_TDPUC[PRT] (0x002C + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_TDPUC, for PRT[0]	0x44100	
3:0	Type	0x2	



6.2.17.5.2 Starting address high at PRTDCB_TDPUC[PRT] (0x002D + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_TDPUC, for PRT[0]		

6.2.17.5.3 Attributes at PRTDCB_TDPUC[PRT] (0x002E + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.5.4 Data low of PRTDCB_TDPUC[PRT] (0x002F + 2*PRT, PRT=0...3)

6.2.17.5.5 Data high of PRTDCB_TDPUC[PRT] (0x0030 + 2*PRT, PRT=0...3)

6.2.17.6 GL_SWT_L2TAGTXIB - (0x0037 - 0x0049)

6.2.17.6.1 Starting address low at GL_SWT_L2TAGTXIB[n] (0x0037)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_SWT_L2TAGTXIB	0x442B8	
3:0	Type	0x2	

6.2.17.6.2 Starting address high at GL_SWT_L2TAGTXIB[n] (0x0038)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_SWT_L2TAGTXIB		

6.2.17.6.3 Attributes at GL_SWT_L2TAGTXIB[n] (0x0039)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0xB	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.6.4 Data low of GL_SWT_L2TAGTXIB[n] (0x003A + 2*n, n=0...7)

6.2.17.6.5 Data high of GL_SWT_L2TAGTXIB[n] (0x003B + 2*n, n=0...7)

6.2.17.7 GLLAN_TSOMSK_F - (0x004A - 0x004B)

6.2.17.7.1 Data low of GLLAN_TSOMSK_F - 0x004A

6.2.17.7.2 Data high of GLLAN_TSOMSK_F - 0x004B

6.2.17.8 GLLAN_TSOMSK_M - (0x004C - 0x004D)

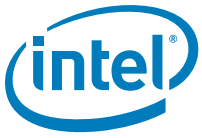
6.2.17.8.1 Data low of GLLAN_TSOMSK_M - 0x004C

6.2.17.8.2 Data high of GLLAN_TSOMSK_M - 0x004D

6.2.17.9 GLLAN_TSOMSK_L - (0x004E - 0x004F)

6.2.17.9.1 Data low of GLLAN_TSOMSK_L - 0x004E

6.2.17.9.2 Data high of GLLAN_TSOMSK_L - 0x004F



6.2.17.10 GLLAN_TCTL_1 - (0x0050 - 0x0054)

6.2.17.10.1 Starting address low at GLLAN_TCTL_1 - 0x0050

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLLAN_TCTL_1	0x442F0	
3:0	Type	0x2	

6.2.17.10.2 Starting address high at GLLAN_TCTL_1 - 0x0051

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLLAN_TCTL_1		

6.2.17.10.3 Attributes at GLLAN_TCTL_1 - 0x0052

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.10.4 Data low of GLLAN_TCTL_1 - 0x0053

6.2.17.10.5 Data high of GLLAN_TCTL_1 - 0x0054

6.2.17.11 GL_MDCK_TDAT - (0x0055 - 0x0056)

6.2.17.11.1 Data low of GL_MDCK_TDAT - 0x0055

6.2.17.11.2 Data high of GL_MDCK_TDAT - 0x0056

6.2.17.12 GL_SWT_L2TAGRXEB - (0x0057 - 0069)

6.2.17.12.1 Starting address low at GL_SWT_L2TAGRXEB[n] (0x0057)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_SWT_L2TAGRXEB	0x51000	
3:0	Type	0x2	

6.2.17.12.2 Starting address high at GL_SWT_L2TAGRXEB[n] (0x0058)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_SWT_L2TAGRXEB		

6.2.17.12.3 Attributes at GL_SWT_L2TAGRXEB[n] (0x0059)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	0x0	
2:0	Width	0x0	

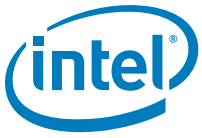
6.2.17.12.4 Data low of GL_SWT_L2TAGRXEB[n] (0x005A + 2*n, n=0...7)

6.2.17.12.5 Data high of GL_SWT_L2TAGRXEB[n] (0x005B + 2*n, n=0...7)

6.2.17.13 GLDCB_RLLPC - (0x006A - 0x006D)

6.2.17.13.1 Address high at GLDCB_RLLPC - 0x006B

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_RLLPC		



6.2.17.13.2 Data low of GLDCB_RLLPC - 0x006C

6.2.17.13.3 Data high of GLDCB_RLLPC - 0x006D

6.2.17.14 PRTDCB_GENC - (0x006E - 0078)

6.2.17.14.1 Starting address low at PRTDCB_GENC[PRT] (0x006E + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_GENC, for PRT[0]	0x83000	
3:0	Type	0x2	

6.2.17.14.2 Starting address high at PRTDCB_GENC[PRT] (0x006F + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_GENC, for PRT[0]		

6.2.17.14.3 Attributes at PRTDCB_GENC[PRT] (0x0070 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.14.4 Data low of PRTDCB_GENC[PRT] (0x0071 + 2*PRT, PRT=0...3)

6.2.17.14.5 Data high of PRTDCB_GENC[PRT] (0x0072 + 2*PRT, PRT=0...3)

6.2.17.15 GLDCB_GENC - (0x0079 - 0x007C)

6.2.17.15.1 Address low at GLDCB_GENC - 0x0079

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLDCB_GENC	0x83044	
3:0	Type	0x1	

6.2.17.15.2 Address high at GLDCB_GENC - 0x007A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_GENC		

6.2.17.15.3 Data low of GLDCB_GENC - 0x007B

6.2.17.15.4 Data high of GLDCB_GENC - 0x007C

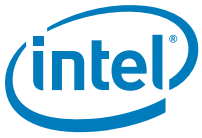
6.2.17.16 ECC_ENA - (0x007D - 0x0080)

6.2.17.16.1 Address high at ECC_ENA - 0x007E

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of ECC_ENA		

6.2.17.16.2 Data low of ECC_ENA - 0x007F

6.2.17.16.3 Data high of ECC_ENA - 0x0080



6.2.17.17 PRTDCB_TFLLPC - (0x0081 - 0x008B)

6.2.17.17.1 Starting address low at PRTDCB_TFLLPC[PRT] (0x0081 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_TFLLPC, for PRT[0]	0x98000	
3:0	Type	0x2	

6.2.17.17.2 Starting address high at PRTDCB_TFLLPC[PRT] (0x0082 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_TFLLPC, for PRT[0]		

6.2.17.17.3 Attributes at PRTDCB_TFLLPC[PRT] (0x0083 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.17.4 Data low of PRTDCB_TFLLPC[PRT] (0x0084 + 2*PRT, PRT=0...3)

6.2.17.17.5 Data high of PRTDCB_TFLLPC[PRT] (0x0085 + 2*PRT, PRT=0...3)

6.2.17.18 PRTDCB_TDPMC - (0x008C - 0x0096)

6.2.17.18.1 Starting address low at PRTDCB_TDPMC[PRT] (0x008C + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_TDPMC, for PRT[0]	0xA0180	
3:0	Type	0x2	

6.2.17.18.2 Starting address high at PRTDCB_TDPMC[PRT] (0x008D + 2*PRT, PRT=0)

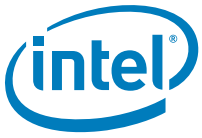
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_TDPMC, for PRT[0]		

6.2.17.18.3 Attributes at PRTDCB_TDPMC[PRT] (0x008E + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.18.4 Data low of PRTDCB_TDPMC[PRT] (0x008F + 2*PRT, PRT=0...3)

6.2.17.18.5 Data high of PRTDCB_TDPMC[PRT] (0x0090 + 2*PRT, PRT=0...3)



6.2.17.19 PRTDCB_TFPFCC - (0x0097 - 0x00A1)

6.2.17.19.1 Starting address low at PRTDCB_TFPFCC[PRT] (0x0097 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_TFPFCC, for PRT[0]	0xA01A0	
3:0	Type	0x2	

6.2.17.19.2 Starting address high at PRTDCB_TFPFCC[PRT] (0x0098 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_TFPFCC, for PRT[0]		

6.2.17.19.3 Attributes at PRTDCB_TFPFCC[PRT] (0x0099 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.19.4 Data low of PRTDCB_TFPFCC[PRT] (0x009A + 2*PRT, PRT=0...3)

6.2.17.19.5 Data high of PRTDCB_TFPFCC[PRT] (0x009B + 2*PRT, PRT=0...3)

6.2.17.20 GLDCB_TGENC_TLPM - (0x00A2 - 0x00A5)

6.2.17.20.1 Address high at GLDCB_TGENC_TLPM - 0x00A3

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_TGENC_TLPM		

6.2.17.20.2 Data low of GLDCB_TGENC_TLPM - 0x00A4

6.2.17.20.3 Data high of GLDCB_TGENC_TLPM - 0x00A5

6.2.17.21 PRTDCB_TCPMC - (0x00A6 - 0x00B0)

6.2.17.21.1 Starting address low at PRTDCB_TCPMC[PRT] (0x00A6 + 2*PRT, PRT=0)

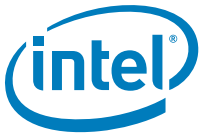
Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_TCPMC, for PRT[0]	0xA21A0	
3:0	Type	0x2	

6.2.17.21.2 Starting address high at PRTDCB_TCPMC[PRT] (0x00A7 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_TCPMC, for PRT[0]		

6.2.17.21.3 Attributes at PRTDCB_TCPMC[PRT] (0x00A8 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	



6.2.17.21.4 Data low of PRTDCB_TCPMC[PRT] (0x00A9 + 2*PRT, PRT=0...3)

6.2.17.21.5 Data high of PRTDCB_TCPMC[PRT] (0x00AA + 2*PRT, PRT=0...3)

6.2.17.22 PRTDCB_TCPFCPC - (0x00B1 - 0x00BB)

6.2.17.22.1 Starting address low at PRTDCB_TCPFCPC[PRT] (0x00B1 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_TCPFCPC, for PRT[0]	0xA21C0	
3:0	Type	0x2	

6.2.17.22.2 Starting address high at PRTDCB_TCPFCPC[PRT] (0x00B2 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_TCPFCPC, for PRT[0]		

6.2.17.22.3 Attributes at PRTDCB_TCPFCPC[PRT] (0x00B3 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.22.4 Data low of PRTDCB_TCPFCPC[PRT] (0x00B4 + 2*PRT, PRT=0...3)

6.2.17.22.5 Data high of PRTDCB_TCPFCPC[PRT] (0x00B5 + 2*PRT, PRT=0...3)

6.2.17.23 PRTDCB_TCPFCTCC - (0x00BC - 0x00C6)

6.2.17.23.1 Starting address low at PRTDCB_TCPFCTCC[PRT] (0x00BC + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_TCPFCTCC, for PRT[0]	0xA21E0	
3:0	Type	0x2	

6.2.17.23.2 Starting address high at PRTDCB_TCPFCTCC[PRT] (0x00BD + 2*PRT, PRT=0)

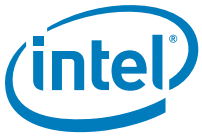
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_TCPFCTCC, for PRT[0]		

6.2.17.23.3 Attributes at PRTDCB_TCPFCTCC[PRT] (0x00BE + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.23.4 Data low of PRTDCB_TCPFCTCC[PRT] (0x00BF + 2*PRT, PRT=0...3)

6.2.17.23.5 Data high of PRTDCB_TCPFCTCC[PRT] (0x00C0 + 2*PRT, PRT=0...3)



6.2.17.24 GLDCB_TGENC_TUPM - (0x00C7 - 0x00CA)

6.2.17.24.1 Address high at GLDCB_TGENC_TUPM - 0x00C8

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_TGENC_TUPM		

6.2.17.24.2 Data low of GLDCB_TGENC_TUPM - 0x00C9

6.2.17.24.3 Data high of GLDCB_TGENC_TUPM - 0x00CA

6.2.17.25 GLFOC_CACHE_CTL - (0x00CB - 0x00CE)

6.2.17.25.1 Address high at GLFOC_CACHE_CTL - 0x00CC

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLFOC_CACHE_CTL		

6.2.17.25.2 Data low of GLFOC_CACHE_CTL - 0x00CD

6.2.17.26 PRTRPB_SPS - (0x00CF - 0x00D9)

6.2.17.26.1 Starting address low at PRTRPB_SPS[PRT] (0x00CF + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTRPB_SPS, for PRT[0]	0xAC7C0	
3:0	Type	0x2	

6.2.17.26.2 Starting address high at PRTRPB_SPS[PRT] (0x00D0 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTRPB_SPS, for PRT[0]		

6.2.17.26.3 Attributes at PRTRPB_SPS[PRT] (0x00D1 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.26.4 Data low of PRTRPB_SPS[PRT] (0x00D2 + 2*PRT, PRT=0...3)

6.2.17.26.5 Data high of PRTRPB_SPS[PRT] (0x00D3 + 2*PRT, PRT=0...3)

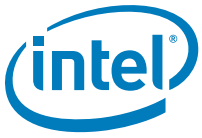
6.2.17.27 GLRPB_PHW - (0x00DA - 0x00DE)

6.2.17.27.1 Starting address low at GLRPB_PHW - 0x00DA

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLRPB_PHW	0xAC844	
3:0	Type	0x2	

6.2.17.27.2 Starting address high at GLRPB_PHW - 0x00DB

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLRPB_PHW		



6.2.17.27.3 Attributes at GLRPB_PHW - 0x00DC

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.27.4 Data low of GLRPB_PHW - 0x00DD

6.2.17.27.5 Data high of GLRPB_PHW - 0x00DE

6.2.17.28 GLRPB_PLW - (0x00DF - 0x00E0)

6.2.17.28.1 Data low of GLRPB_PLW - 0x00DF

6.2.17.28.2 Data high of GLRPB_PLW - 0x00E0

6.2.17.29 PRTDCB_TCLLPC - (0x00E1 - 0x00EB)

6.2.17.29.1 Starting address low at PRTDCB_TCLLPC[PRT] (0x00E1 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_TCLLPC, for PRT[0]	0xAE000	
3:0	Type	0x2	

6.2.17.29.2 Starting address high at PRTDCB_TCLLPC[PRT] (0x00E2 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_TCLLPC, for PRT[0]		

6.2.17.29.3 Attributes at PRTDCB_TCLLPC[PRT] (0x00E3 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.29.4 Data low of PRTDCB_TCLLPC[PRT] (0x00E4 + 2*PRT, PRT=0...3)

6.2.17.29.5 Data high of PRTDCB_TCLLPC[PRT] (0x00E5 + 2*PRT, PRT=0...3)

6.2.17.30 GLLAN_TCTL_2 - (0x00EC - 0x00EF)

6.2.17.30.1 Address high at GLLAN_TCTL_2 - 0x00ED

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLLAN_TCTL_2		

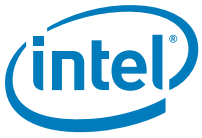
6.2.17.30.2 Data low of GLLAN_TCTL_2 - 0x00EE

6.2.17.30.3 Data high of GLLAN_TCTL_2 - 0x00EF

6.2.17.31 GLSCD_QUANTA - (0x00F0 - 0x00F3)

6.2.17.31.1 Address low at GLSCD_QUANTA - 0x00F0

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLSCD_QUANTA	0xB2080	
3:0	Type	0x1	



6.2.17.31.2 Address high at GLSCD_QUANTA - 0x00F1

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLSCD_QUANTA		

6.2.17.31.3 Data low of GLSCD_QUANTA - 0x00F2

6.2.17.31.4 Data high of GLSCD_QUANTA - 0x00F3

6.2.17.32 GLSCD_LLPREALTHRESH - (0x00F4 - 0x00F7)

6.2.17.32.1 Address high at GLSCD_LLPREALTHRESH - 0x00F5

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLSCD_LLPREALTHRESH		

6.2.17.32.2 Data low of GLSCD_LLPREALTHRESH - 0x00F6

6.2.17.32.3 Data high of GLSCD_LLPREALTHRESH - 0x00F7

6.2.17.33 GLSCD_CREDITSPERQUANTA - (0x00F8 - 0x00FC)

6.2.17.33.1 Starting address low at GLSCD_CREDITSPERQUANTA - 0x00F8

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLSCD_CREDITSPERQUANTA	0xB2144	
3:0	Type	0x2	

6.2.17.33.2 Starting address high at GLSCD_CREDITSPERQUANTA - 0x00F9

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLSCD_CREDITSPERQUANTA		

6.2.17.33.3 Attributes at GLSCD_CREDITSPERQUANTA - 0x00FA

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x3	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.33.4 Data low of GLSCD_CREDITSPERQUANTA - 0x00FB

6.2.17.33.5 Data high of GLSCD_CREDITSPERQUANTA - 0x00FC

6.2.17.34 GLSCD_BWLCREDUPDATE - (0x00FD - 0x00FE)

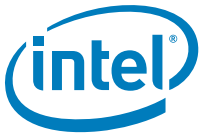
6.2.17.34.1 Data low of GLSCD_BWLCREDUPDATE - 0x00FD

6.2.17.34.2 Data high of GLSCD_BWLCREDUPDATE - 0x00FE

6.2.17.35 GLSCD_BWLLINESPERARB - (0x00FF - 0x0100)

6.2.17.35.1 Data low of GLSCD_BWLLINESPERARB - 0x00FF

6.2.17.35.2 Data high of GLSCD_BWLLINESPERARB - 0x0100



6.2.17.36 GLSCD_LANTCBCMDS - (0x0101 - 0x0104)

6.2.17.36.1 Address high at GLSCD_LANTCBCMDS - 0x0102

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLSCD_LANTCBCMDS		

6.2.17.36.2 Data low of GLSCD_LANTCBCMDS - 0x0103

6.2.17.36.3 Data high of GLSCD_LANTCBCMDS - 0x0104

6.2.17.37 GLHMC_SDPART - (0x0105 - 0x0127)

6.2.17.37.1 Starting address low at GLHMC_SDPART[n] (0x0105)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLHMC_SDPART	0xC0800	
3:0	Type	0x2	

6.2.17.37.2 Starting address high at GLHMC_SDPART[n] (0x0106)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLHMC_SDPART		

6.2.17.37.3 Attributes at GLHMC_SDPART[n] (0x0107)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.37.4 Data low of GLHMC_SDPART[n] (0x0108 + 2*n, n=0...15)

6.2.17.37.5 Data high of GLHMC_SDPART[n] (0x0109 + 2*n, n=0...15)

6.2.17.38 GLHMC_PMATCFG - (0x0128 - 0x012B)

6.2.17.38.1 Address high at GLHMC_PMATCFG - 0x0129

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLHMC_PMATCFG		

6.2.17.38.2 Data low of GLHMC_PMATCFG - 0x012A

6.2.17.38.3 Data high of GLHMC_PMATCFG - 0x012B

6.2.17.39 GLLAN_TCTL_0 - (0x012C - 0x0130)

6.2.17.39.1 Starting address low at GLLAN_TCTL_0 - 0x012C

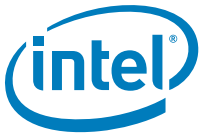
Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLLAN_TCTL_0	0xE6488	
3:0	Type	0x2	

6.2.17.39.2 Starting address high at GLLAN_TCTL_0 - 0x012D

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLLAN_TCTL_0		

6.2.17.39.3 Attributes at GLLAN_TCTL_0 - 0x012E

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	



6.2.17.39.4 Data low of GLLAN_TCTL_0 - 0x012F

6.2.17.39.5 Data high of GLLAN_TCTL_0 - 0x0130

6.2.17.40 GL_MDCK_TCMD - (0x0131 - 0x0132)

6.2.17.40.1 Data low of GL_MDCK_TCMD - 0x0131

6.2.17.40.2 Data high of GL_MDCK_TCMD - 0x0132

6.2.17.41 GLTLAN_MIN_MAX_PKT - (0x0133 - 0x0137)

6.2.17.41.1 Starting address low at GLTLAN_MIN_MAX_PKT - 0x0133

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLTLAN_MIN_MAX_PKT	0xE64D8	
3:0	Type	0x2	

6.2.17.41.2 Starting address high at GLTLAN_MIN_MAX_PKT - 0x0134

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLTLAN_MIN_MAX_PKT		

6.2.17.41.3 Attributes at GLTLAN_MIN_MAX_PKT - 0x0135

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.41.4 Data low of GLTLAN_MIN_MAX_PKT - 0x0136

6.2.17.41.5 Data high of GLTLAN_MIN_MAX_PKT - 0x0137

6.2.17.42 GLTLAN_MIN_MAX_MSS - (0x0138 - 0x0139)

6.2.17.42.1 Data low of GLTLAN_MIN_MAX_MSS - 0x0138

6.2.17.42.2 Data high of GLTLAN_MIN_MAX_MSS - 0x0139

6.2.17.43 GLCM_LANCONFIG - (0x013A - 0x013D)

6.2.17.43.1 Address high at GLCM_LANCONFIG - 0x013B

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLCM_LANCONFIG		

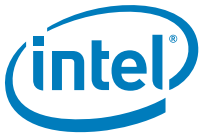
6.2.17.44 PRTDCB_RETSTCC - (0x013E - 0x0195)

6.2.17.44.1 Starting address low at PRTDCB_RETSTCC[n,PRT] (0x013E + 11*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_RETSTCC, for PRT[0]	0x122180	
3:0	Type	0x2	

6.2.17.44.2 Starting address high at PRTDCB_RETSTCC[n,PRT] (0x013F + 11*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_RETSTCC, for PRT[0]		



6.2.17.44.3 Attributes at PRTDCB_RETSTCC[n,PRT] (0x0140 + 11*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.44.4 Data low of PRTDCB_RETSTCC[n,PRT] (0x0141 + 11*n + 2*PRT, n=0...7, PRT=0...3)

6.2.17.44.5 Data high of PRTDCB_RETSTCC[n,PRT] (0x0142 + 11*n + 2*PRT, n=0...7, PRT=0...3)

6.2.17.45 PRTDCB_RPPMC - (0x0196 - 0x01A0)

6.2.17.45.1 Starting address low at PRTDCB_RPPMC[PRT] (0x0196 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_RPPMC, for PRT[0]	0x1223A0	
3:0	Type	0x2	

6.2.17.45.2 Starting address high at PRTDCB_RPPMC[PRT] (0x0197 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_RPPMC, for PRT[0]		

6.2.17.45.3 Attributes at PRTDCB_RPPMC[PRT] (0x0198 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	

Bits	Field Name	Default NVM Value	Description
2:0	Width	0x0	

6.2.17.45.4 Data high of PRTDCB_RPPMC[PRT] (0x019A + 2*PRT, PRT=0...3)

6.2.17.46 PRTDCB_RETSC - (0x01A1 - 0x01AB)

6.2.17.46.1 Starting address low at PRTDCB_RETSC[PRT] (0x01A1 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_RETSC, for PRT[0]	0x1223E0	
3:0	Type	0x2	

6.2.17.46.2 Starting address hHigh at PRTDCB_RETSC[PRT] (0x01A2 + 2*PRT, PRT=0)

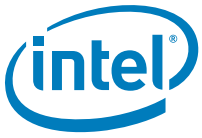
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_RETSC, for PRT[0]		

6.2.17.46.3 Attributes at PRTDCB_RETSC[PRT] (0x01A3 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.46.4 Data low of PRTDCB_RETSC[PRT] (0x01A4 + 2*PRT, PRT=0...3)

6.2.17.46.5 Data high of PRTDCB_RETSC[PRT] (0x01A5 + 2*PRT, PRT=0...3)



6.2.17.47 GLDCB_RSPMC - (0x01AC - 0x01B0)

6.2.17.47.1 Starting address low at GLDCB_RSPMC - 0x01AC

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLDCB_RSPMC	0x122604	
3:0	Type	0x2	

6.2.17.47.2 Starting address high at GLDCB_RSPMC - 0x01AD

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_RSPMC		

6.2.17.47.3 Attributes at GLDCB_RSPMC - 0x01AE

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.47.4 Data low of GLDCB_RSPMC - 0x01AF

6.2.17.47.5 Data high of GLDCB_RSPMC - 0x01B0

6.2.17.48 GLDCB_RPRRD0 - (0x01B1 - 0x01B2)

6.2.17.48.1 Data low of GLDCB_RPRRD0 - 0x01B1

6.2.17.48.2 Data high of GLDCB_RPRRD0 - 0x01B2

6.2.17.49 GLDCB_RPRRD1 - (0x01B3 - 0x01B4)

6.2.17.49.1 Data low of GLDCB_RPRRD1 - 0x01B3

6.2.17.49.2 Data high of GLDCB_RPRRD1 - 0x01B4

6.2.17.50 GLDCB_RMPMC - (0x01B5 - 0x01B6)

6.2.17.50.1 Data low of GLDCB_RMPMC - 0x01B5

6.2.17.50.2 Data high of GLDCB_RMPMC - 0x01B6

6.2.17.51 GLLAN_RCTL_1 - (0x01B7 - 0x01BB)

6.2.17.51.1 Starting address low at GLLAN_RCTL_1 - 0x01B7

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLLAN_RCTL_1	0x12A504	
3:0	Type	0x2	

6.2.17.51.2 Starting address high at GLLAN_RCTL_1 - 0x01B8

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLLAN_RCTL_1		

6.2.17.51.3 Attributes at GLLAN_RCTL_1 - 0x01B9

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x3	
4:3	Skip	0x0	
2:0	Width	0x0	

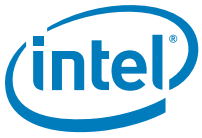
6.2.17.51.4 Data Low of GLLAN_RCTL_1 - 0x01BA

6.2.17.51.5 Data High of GLLAN_RCTL_1 - 0x01BB

6.2.17.52 GLFCOE_RLANCTL - (0x01BC - 0x01BD)

6.2.17.52.1 Data low of GLFCOE_RLANCTL - 0x01BC

6.2.17.52.2 Data high of GLFCOE_RLANCTL - 0x01BD



6.2.17.53 GL_MDCK_RX - (0x01BE - 0x01BF)

6.2.17.53.1 Data low of GL_MDCK_RX - 0x01BE

6.2.17.53.2 Data high of GL_MDCK_RX - 0x01BF

6.2.17.54 GLLAN_PF_RECIPE - (0x01C0 - 01E2)

6.2.17.54.1 Starting address low at GLLAN_PF_RECIPE[n] (0x01C0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLLAN_PF_RECIPE	0x12A5E0	
3:0	Type	0x2	

6.2.17.54.2 Starting address high at GLLAN_PF_RECIPE[n] (0x01C1)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLLAN_PF_RECIPE		

6.2.17.54.3 Attributes at GLLAN_PF_RECIPE[n] (0x01C2)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.54.4 Data low of GLLAN_PF_RECIPE[n] (0x01C3 + 2*n, n=0...15)

6.2.17.54.5 Data high of GLLAN_PF_RECIPE[n] (0x01C4 + 2*n, n=0...15)

6.2.17.55 PFLAN_QALLOC - (0x01E3 - 0x0205)

6.2.17.55.1 Starting address low at PFLAN_QALLOC[PF] (0x01E3 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFLAN_QALLOC, for PF[0]	0x1C0400	
3:0	Type	0x2	

6.2.17.55.2 Starting address high at PFLAN_QALLOC[PF] (0x01E4 + 2*PF, PF=0)

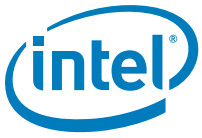
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFLAN_QALLOC, for PF[0]		

6.2.17.55.3 Attributes at PFLAN_QALLOC[PF] (0x01E5 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.55.4 Data low of PFLAN_QALLOC[PF] (0x01E6 + 2*PF, PF=0...15)

6.2.17.55.5 Data high of PFLAN_QALLOC[PF] (0x01E7 + 2*PF, PF=0...15)



6.2.17.56 PFGEN_PORTNUM - (0x0206 - 0x0228)

6.2.17.56.1 Starting address low at PFGEN_PORTNUM[PF] (0x0206 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFGEN_PORTNUM, for PF[0]	0x1C0480	
3:0	Type	0x2	

6.2.17.56.2 Starting address high at PFGEN_PORTNUM[PF] (0x0207 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFGEN_PORTNUM, for PF[0]		

6.2.17.56.3 Attributes at PFGEN_PORTNUM[PF] (0x0208 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.56.4 Data low of PFGEN_PORTNUM[PF] (0x0209 + 2*PF, PF=0...15)

6.2.17.56.5 Data high of PFGEN_PORTNUM[PF] (0x020A + 2*PF, PF=0...15)

6.2.17.57 PF_VT_PFALLOC - (0x0229 - 0x024B)

6.2.17.57.1 Starting address low at PF_VT_PFALLOC[PF] (0x0229 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PF_VT_PFALLOC, for PF[0]	0x1C0500	
3:0	Type	0x2	

6.2.17.57.2 Starting address high at PF_VT_PFALLOC[PF] (0x022A + 2*PF, PF=0)

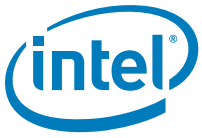
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PF_VT_PFALLOC, for PF[0]		

6.2.17.57.3 Attributes at PF_VT_PFALLOC[PF] (0x022B + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.57.4 Data low of PF_VT_PFALLOC[PF] (0x022C + 2*PF, PF=0...15)

6.2.17.57.5 Data high of PF_VT_PFALLOC[PF] (0x022D + 2*PF, PF=0...15)



6.2.17.58 GLANL_PRE_LY2 - (0x024C - 0x024F)

6.2.17.58.1 Address high at GLANL_PRE_LY2 - 0x024D

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLANL_PRE_LY2		

6.2.17.59 GLANL_L2ULP - (0x0250 - 0x0272)

6.2.17.59.1 Starting address low at GLANL_L2ULP[n] (0x0250)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLANL_L2ULP	0x1C0A2C	
3:0	Type	0x2	

6.2.17.59.2 Starting address high at GLANL_L2ULP[n] (0x0251)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLANL_L2ULP		

6.2.17.59.3 Attributes at GLANL_L2ULP[n] (0x0252)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x19	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.60 GLGEN_DUAL40 - (0x0273 - 0x0274)

6.2.17.60.1 Data low of GLGEN_DUAL40 - 0x0273

6.2.17.60.2 Data high of GLGEN_DUAL40 - 0x0274

6.2.17.61 GL_SWT_L2TAGCTRL - (0x0275 - 0x0284)

6.2.17.61.1 Data low of GL_SWT_L2TAGCTRL[n] (0x0275 + 2*n, n=0...7)

6.2.17.61.2 Data high of GL_SWT_L2TAGCTRL[n] (0x0276 + 2*n, n=0...7)

6.2.17.62 PRT_L2TAGSEN - (0x0285 - 0x028F)

6.2.17.62.1 Starting address low at PRT_L2TAGSEN[PRT] (0x0285 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRT_L2TAGSEN, for PRT[0]	0x1C0B20	
3:0	Type	0x2	

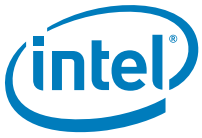
6.2.17.62.2 Starting address high at PRT_L2TAGSEN[PRT] (0x0286 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRT_L2TAGSEN, for PRT[0]		

6.2.17.62.3 Attributes at PRT_L2TAGSEN[PRT] (0x0287 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.62.4 Data low of PRT_L2TAGSEN[PRT] (0x0288 + 2*PRT, PRT=0...3)



6.2.17.62.5 Data high of PRT_L2TAGSEN[PRT] (0x0289 + 2*PRT, PRT=0...3)

6.2.17.63 PFQF_FDALLOC - (0x0290 - 0x02B2)

6.2.17.63.1 Starting address low at PFQF_FDALLOC[PF] (0x0290 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFQF_FDALLOC, for PF[0]	0x246280	
3:0	Type	0x2	

6.2.17.63.2 Starting address high at PFQF_FDALLOC[PF] (0x0291 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFQF_FDALLOC, for PF[0]		

6.2.17.63.3 Attributes at PFQF_FDALLOC[PF] (0x0292 + 2*PF, PF=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.63.4 Data low of PFQF_FDALLOC[PF] (0x0293 + 2*PF, PF=0...15)

6.2.17.63.5 Data high of PFQF_FDALLOC[PF] (0x0294 + 2*PF, PF=0...15)

6.2.17.64 PRT_MSCCNT - (0x02B3 - 0x02BD)

6.2.17.64.1 Starting address low at PRT_MSCCNT[PRT] (0x02B3 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRT_MSCCNT, for PRT[0]	0x256BA0	
3:0	Type	0x2	

6.2.17.64.2 Starting address high at PRT_MSCCNT[PRT] (0x02B4 + 2*PRT, PRT=0)

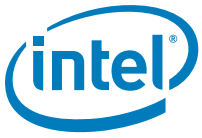
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRT_MSCCNT, for PRT[0]		

6.2.17.64.3 Attributes at PRT_MSCCNT[PRT] (0x02B5 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.64.4 Data low of PRT_MSCCNT[PRT] (0x02B6 + 2*PRT, PRT=0...3)

6.2.17.64.5 Data high of PRT_MSCCNT[PRT] (0x02B7 + 2*PRT, PRT=0...3)



6.2.17.65 PRT_SWT_BSCCNT - (0x02BE - 0x02C8)

6.2.17.65.1 Starting address low at PRT_SWT_BSCCNT[PRT] (0x02BE + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRT_SWT_BSCCNT, for PRT[0]	0x256C60	
3:0	Type	0x2	

6.2.17.65.2 Starting address high at PRT_SWT_BSCCNT[PRT] (0x02BF + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRT_SWT_BSCCNT, for PRT[0]		

6.2.17.65.3 Attributes at PRT_SWT_BSCCNT[PRT] (0x02C0 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.65.4 Data low of PRT_SWT_BSCCNT[PRT] (0x02C1 + 2*PRT, PRT=0...3)

6.2.17.65.5 Data high of PRT_SWT_BSCCNT[PRT] (0x02C2 + 2*PRT, PRT=0...3)

6.2.17.66 PRT_SWT_BSCTRH - (0x02C9 - 0x02D3)

6.2.17.66.1 Starting address low at PRT_SWT_BSCTRH[PRT] (0x02C9 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRT_SWT_BSCTRH, for PRT[0]	0x256CA0	
3:0	Type	0x2	

6.2.17.66.2 Starting address high at PRT_SWT_BSCTRH[PRT] (0x02CA + 2*PRT, PRT=0)

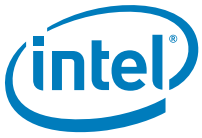
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRT_SWT_BSCTRH, for PRT[0]		

6.2.17.66.3 Attributes at PRT_SWT_BSCTRH[PRT] (0x02CB + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.66.4 Data low of PRT_SWT_BSCTRH[PRT] (0x02CC + 2*PRT, PRT=0...3)

6.2.17.66.5 Data high of PRT_SWT_BSCTRH[PRT] (0x02CD + 2*PRT, PRT=0...3)



6.2.17.67 PRT_SWT_SCBI - (0x02D4 - 0x02DE)

6.2.17.67.1 Starting address low at PRT_SWT_SCBI[PRT] (0x02D4 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRT_SWT_SCBI, for PRT[0]	0x256D60	
3:0	Type	0x2	

6.2.17.67.2 Starting address high at PRT_SWT_SCBI[PRT] (0x02D5 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRT_SWT_SCBI, for PRT[0]		

6.2.17.67.3 Attributes at PRT_SWT_SCBI[PRT] (0x02D6 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.67.4 Data low of PRT_SWT_SCBI[PRT] (0x02D7 + 2*PRT, PRT=0...3)

6.2.17.67.5 Data high of PRT_SWT_SCBI[PRT] (0x02D8 + 2*PRT, PRT=0...3)

6.2.17.68 PRT_SWT_SCTC - (0x02DF - 0x02E9)

6.2.17.68.1 Starting address low at PRT_SWT_SCTC[PRT] (0x02DF + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRT_SWT_SCTC, for PRT[0]	0x256DE0	
3:0	Type	0x2	

6.2.17.68.2 Starting address high at PRT_SWT_SCTC[PRT] (0x02E0 + 2*PRT, PRT=0)

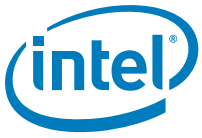
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRT_SWT_SCTC, for PRT[0]		

6.2.17.68.3 Attributes at PRT_SWT_SCTC[PRT] (0x02E1 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.68.4 Data low of PRT_SWT_SCTC[PRT] (0x02E2 + 2*PRT, PRT=0...3)

6.2.17.68.5 Data high of PRT_SWT_SCTC[PRT] (0x02E3 + 2*PRT, PRT=0...3)



6.2.17.69 PRTQF_CTL_0 - (0x02EA - 0x02F4)

6.2.17.69.1 Starting address low at PRTQF_CTL_0[PRT] (0x02EA + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTQF_CTL_0, for PRT[0]	0x256E60	
3:0	Type	0x2	

6.2.17.69.2 Starting address high at PRTQF_CTL_0[PRT] (0x02EB + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTQF_CTL_0, for PRT[0]		

6.2.17.69.3 Attributes at PRTQF_CTL_0[PRT] (0x02EC + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.69.4 Data low of PRTQF_CTL_0[PRT] (0x02ED + 2*PRT, PRT=0...3)

6.2.17.69.5 Data high of PRTQF_CTL_0[PRT] (0x02EE + 2*PRT, PRT=0...3)

6.2.17.70 GLQF_PTYPE - (0x02F5 - 0x03F7)

6.2.17.70.1 Starting address low at GLQF_PTYPE[0,0] (0x02F5)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLQF_PTYPE	0x268200	
3:0	Type	0x2	

6.2.17.70.2 Starting address high at GLQF_PTYPE[0,0] (0x02F6)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLQF_PTYPE		

6.2.17.70.3 Attributes at GLQF_PTYPE[0,0] (0x02F7)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x80	
4:3	Skip	0x0	
2:0	Width	0x0	

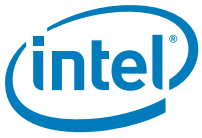
6.2.17.70.4 Data low of GLQF_PTYPE[n,m] (0x02F8 + 2*n +4m, n=0...1, m=0...63)

6.2.17.70.5 Data high of GLQF_PTYPE[n,m] (0x02F9 + 2*n +4m, n=0...1, m=0...63)

6.2.17.71 GLQF_PTYPE_ENA - (0x03F8 - 0x04FA)

6.2.17.71.1 Starting address low at GLQF_PTYPE_ENA[0,0] (0x03F8)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLQF_PTYPE_ENA	0x268600	
3:0	Type	0x2	



6.2.17.71.2 Starting address high at GLQF_PTYPE_ENA[0,0] (0x03F9)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLQF_PTYPE_ENA		

6.2.17.71.3 Attributes at GLQF_PTYPE_ENA[0,0] (0x03FA)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x80	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.71.4 Data low of GLQF_PTYPE_ENA[n,m] (0x03FB + 2*n +4m, n=0...1, m=0...63)

6.2.17.71.5 Data high of GLQF_PTYPE_ENA[n,m] (0x03FC + 2*n +4m, n=0...1, m=0...63)

6.2.17.72 GLQF_FDENA - (0x0541 - 0x0547)

6.2.17.72.1 Starting address low at GLQF_FDENA[n] (0x0541)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLQF_FDENA	0x2698A8	
3:0	Type	0x2	

6.2.17.72.2 Starting address high at GLQF_FDENA[n] (0x0542)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLQF_FDENA		

6.2.17.72.3 Attributes at GLQF_FDENA[n] (0x0543)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.72.4 Data low of GLQF_FDENA[n] (0x0544 + 2*n, n=0...1)

6.2.17.72.5 Data high of GLQF_FDENA[n] (0x0545 + 2*n, n=0...1)

6.2.17.73 GL_RXA_CFG - (0x0548 - 0x054B)

6.2.17.73.1 Address high at GL_RXA_CFG - 0x0549

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_RXA_CFG		

6.2.17.73.2 Data low of GL_RXA_CFG - 0x054A

6.2.17.73.3 Data high of GL_RXA_CFG - 0x054B

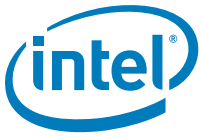
6.2.17.74 GL_SWR_DP - (0x054C - 0x054F)

6.2.17.74.1 Address high at GL_SWR_DP - 0x054D

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_SWR_DP		

6.2.17.74.2 Data low of GL_SWR_DP - 0x054E

6.2.17.74.3 Data high of GL_SWR_DP - 0x054F



6.2.17.75 GLFCOE_RCTL - (0x0550 - 0x0553)

6.2.17.75.1 Address low at GLFCOE_RCTL - 0x0550

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLFCOE_RCTL	0x269B94	
3:0	Type	0x1	

6.2.17.75.2 Address high at GLFCOE_RCTL - 0x0551

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLFCOE_RCTL		

6.2.17.75.3 Data low of GLFCOE_RCTL - 0x0552

6.2.17.75.4 Data high of GLFCOE_RCTL - 0x0553

6.2.17.76 GLFCOE_RSOF - (0x0554 - 0x0557)

6.2.17.76.1 Address high at GLFCOE_RSOF - 0x0555

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLFCOE_RSOF		

6.2.17.76.2 Data low of GLFCOE_RSOF - 0x0556

6.2.17.76.3 Data high of GLFCOE_RSOF - 0x0557

6.2.17.77 GLQF_CTL - (0x0558 - 0x055B)

6.2.17.77.1 Address low at GLQF_CTL - 0x0558

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLQF_CTL	0x269BA4	
3:0	Type	0x1	

6.2.17.77.2 Address high at GLQF_CTL - 0x0559

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLQF_CTL		

6.2.17.77.3 Data low of GLQF_CTL - 0x055A

6.2.17.77.4 Data high of GLQF_CTL - 0x055B

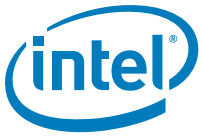
6.2.17.78 PRT_SWR_PM_THR - (0x055C - 0566)

6.2.17.78.1 Starting address low at PRT_SWR_PM_THR[PRT] (0x055C + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRT_SWR_PM_THR, for PRT[0]	0x26CD00	
3:0	Type	0x2	

6.2.17.78.2 Starting address high at PRT_SWR_PM_THR[PRT] (0x055D + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRT_SWR_PM_THR, for PRT[0]		



6.2.17.78.3 Attributes at PRT_SWR_PM_THR[PRT] (0x055E + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.78.4 Data low of PRT_SWR_PM_THR[PRT] (0x055F + 2*PRT, PRT=0...3)

6.2.17.78.5 Data high of PRT_SWR_PM_THR[PRT] (0x0560 + 2*PRT, PRT=0...3)

6.2.17.79 GLQF_HKEY - (0x0567 - 0x0583)

6.2.17.79.1 Starting address low at GLQF_HKEY[n] (0x0567)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLQF_HKEY	0x270140	
3:0	Type	0x2	

6.2.17.79.2 Starting address high at GLQF_HKEY[n] (0x0568)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLQF_HKEY		

6.2.17.79.3 Attributes at GLQF_HKEY[n] (0x0569)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0xD	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.17.79.4 Data lhHigh of GLQF_HKEY[n] (0x056B + 2*n, n=0...12)

6.2.17.80 DPU_IMEM Attributes - 0x0586

Part of a Type 3 structure used to load the DPU_IMEM.

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x400	
4:3	Skip	0x00	
2:0	Width	0x2	

6.2.17.81 DPU_IMEM Data - 0x0589

Raw data module length: 8192 words.

Part of a Type 3 structure used to load the DPU_IMEM.

6.2.17.82 DPU_RECIPE_ADDRESS attributes - 0x258B

Part of a Type 3 structure used to load the DPU_RECIPE_ADDRESS.

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x100	
4:3	Skip	0x00	

6.2.17.83 DPU_RECIPE_ADDRESS data - 0x258E

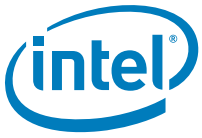
Raw data module length: 512 words.

Part of a Type 3 structure used to load the DPU_RECIPE_ADDRESS.

6.2.17.84 DPU_RECIPE_CAM attributes - 0x2790

Part of a Type 3 structure used to load the DPU_RECIPE_CAM.

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x100	
4:3	Skip	0x00	



6.2.17.85 DPU_RECIPE_CAM data - 0x2793

Raw data module length: 1024 words.

Part of a Type 3 structure used to load the DPU_RECIPE_CAM.

6.2.17.86 DPU_RECIPE_MASK attributes - 0x2B95

Part of a Type 3 structure used to load the DPU_RECIPE_MASK.

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x008	
4:3	Skip	0x00	

6.2.17.87 DPU_RECIPE_MASK data - 0x2B98

Raw data module length: 32 words.

Part of a Type 3 structure used to load the DPU_RECIPE_MASK.

6.2.17.88 ANA_IMEM attributes - 0x2BBA

Part of a Type 3 structure used to load the DPU_IMEM.

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	0x00	

6.2.17.89 ANA_IMEM data - 0x2BBD

Raw data module length: 160 words.

Part of a Type 3 structure used to load the DPU_IMEM.

6.2.17.90 ANA_NH attributes - 0x2C5F

Part of a Type 3 structure used to load the DPU_IMEM.

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x28	

Bits	Field Name	Default NVM Value	Description
4:3	Skip	0x00	

6.2.17.91 ANA_NH data - 0x2C62

Raw data module length: 80 words.

Part of a Type 3 structure used to load the DPU_IMEM.

6.2.17.92 ANA_SKIP attributes - 0x2CB4

Part of a Type 3 structure used to load the DPU_IMEM.

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	0x00	

6.2.17.93 ANA_SKIP data - 0x2CB7

Raw data module length: 80 words.

Part of a Type 3 structure used to load the DPU_IMEM.

6.2.17.94 ANA_REPLACE attributes - 0x2D09

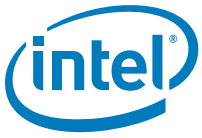
Part of a Type 3 structure used to load the DPU_IMEM.

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	0x00	

6.2.17.95 ANA_REPLACE data - 0x2D0C

Raw data module length: 80 words.

Part of a Type 3 structure used to load the DPU_IMEM.



6.2.17.96 ANA_MERGE attributes - 0x2D5E

Part of a Type 3 structure used to load the DPU_IMEM.

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	0x00	

6.2.17.97 ANA_MERGE data - 0x2D61

Raw data module length: 80 words.

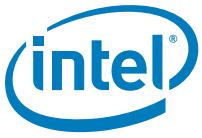
Part of a Type 3 structure used to load the DPU_IMEM.

6.2.18 GLOBR registers auto-load module section summary table

Default setup to registers that load on GLOBR events.

Word Offset	Description	Reference
0x0000	Module Length	453
0x0001 - 0005	NVM contents for PRTMAC_PCS_MUX_KR	453
0x0006 - 0007	NVM contents for PRTMAC_PMD_MUX_KR	454
0x0008 - 0009	NVM contents for PRTMAC_PCS_MUX_KX	454
0x000A - 000B	NVM contents for PRTMAC_PMD_MUX_KX	454
0x000C - 0016	NVM contents for PRTMAC_PCS_LINK_CTRL	454
0x0017 - 001B	NVM contents for PRTMAC_PCS_XAUI_SWAP_A	455
0x001C - 001D	NVM contents for PRTMAC_PCS_XAUI_SWAP_B	456
0x001E - 0028	NVM contents for PRTMAC_PCS_AN_CONTROL1	456
0x0029 - 0033	NVM contents for PRTMAC_PCS_AN_CONTROL2	457
0x0034 - 003E	NVM contents for PRTMAC_PCS_AN_CONTROL4	458
0x003F - 0049	NVM contents for PRTMAC_HLCTL	459

Word Offset	Description	Reference
0x004A - 0054	NVM contents for PRTMAC_PAP	460
0x0055 - 005F	NVM contents for PRTGL_SAL	461
0x0060 - 006A	NVM contents for PRTGL_SAH	462
0x006B - 0075	NVM contents for PRTDCB_MFLCN	463
0x0076 - 007C	NVM contents for PRTMAC_HSEC_CTL_RX_MAX_PACKET_LEN	464
0x007D - 0083	NVM contents for PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE	465
0x0084 - 008A	NVM contents for PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE	466
0x008B - 0091	NVM contents for PRTMAC_HSEC_CTL_RX_ENABLE_GCP	467
0x0092 - 0098	NVM contents for PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP	468
0x0099 - 009F	NVM contents for PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1	469
0x00A0 - 00A6	NVM contents for PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2	470
0x00A7 - 00AD	NVM contents for PRTMAC_HSEC_CTL_RX_CHECK_SA_GCP	471
0x00AE - 00B4	NVM contents for PRTMAC_HSEC_CTL_RX_ENABLE_GPP	472
0x00B5 - 00BB	NVM contents for PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GPP	473
0x00BC - 00C2	NVM contents for PRTMAC_HSEC_CTL_RX_ENABLE_PPP	474
0x00C3 - 00C9	NVM contents for PRTMAC_HSEC_CTL_RX_CHECK_UCAST_PPP	475
0x00CA - 00D0	NVM contents for PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL	476
0x00D1 - 010F	NVM contents for PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA	477
0x0110 - 014E	NVM contents for PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER	478
0x014F - 0155	NVM contents for PRTMAC_HSEC_CTL_TX_SA_PART1	479
0x0156 - 015C	NVM contents for PRTMAC_HSEC_CTL_TX_SA_PART2	480
0x015D - 0163	NVM contents for PRTMAC_HSEC_CTL_INTERNAL	481
0x0164 - 0167	NVM contents for PRTMAC_HSEC_SINGLE_40G_PORT_SELECT	482



Word Offset	Description	Reference
0x0168 - 016E	NVM contents for PRTMAC_HSEC_CTL_XLGMII	482
0x016F - 0175	NVM contents for PRTMAC_HSECTL1	483
0x0176 - 0180	NVM contents for PRTTSYN_CTL0	484
0x0181 - 0185	NVM contents for GLPM_EEE_SU	485
0x0186 - 0187	NVM contents for GLPM_EEE_SU_EXT	486
0x0188 - 0192	NVM contents for PRTPM_EEER	486
0x0193 - 019D	NVM contents for PRTPM_EEEC	487
0x019E - 01C9	NVM contents for PRTDCB_FCTTVN	488
0x01CA - 01D4	NVM contents for PRTDCB_FCRTV	489
0x01D5 - 01DF	NVM contents for PRTDCB_FCCFG	490
0x01E0 - 01EA	NVM contents for PRTMAC_HLCTLA	491

6.2.18.1 Module length - 0x0000

Bits	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 -> Starting Address Low at PRTMAC_HLCTLA, for PRT[0].

6.2.18.2 PRTMAC_PCS_MUX_KR - (0x0001 - 0x0005)

6.2.18.2.1 Starting address high at PRTMAC_PCS_MUX_KR - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_PCS_MUX_KR		

6.2.18.2.2 Data low of PRTMAC_PCS_MUX_KR - 0x0004

6.2.18.2.3 Data high of PRTMAC_PCS_MUX_KR - 0x0005

6.2.18.3 PRTMAC_PMD_MUX_KR - (0x0006 - 0x0007)

6.2.18.3.1 Data low of PRTMAC_PMD_MUX_KR - 0x0006

6.2.18.3.2 Data high of PRTMAC_PMD_MUX_KR - 0x0007

6.2.18.4 PRTMAC_PCS_MUX_KX - (0x0008 - 0x0009)

6.2.18.4.1 Data low of PRTMAC_PCS_MUX_KX - 0x0008

6.2.18.4.2 Data high of PRTMAC_PCS_MUX_KX - 0x0009

6.2.18.5 PRTMAC_PMD_MUX_KX - (0x000A - 0x000B)

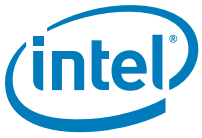
6.2.18.5.1 Data low of PRTMAC_PMD_MUX_KX - 0x000A

6.2.18.5.2 Data high of PRTMAC_PMD_MUX_KX - 0x000B

6.2.18.6 PRTMAC_PCS_LINK_CTRL - (0x000C - 0x0016)

6.2.18.6.1 Starting address low at PRTMAC_PCS_LINK_CTRL[PRT] (0x000C + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_PCS_LINK_CTRL, for PRT[0]	0x8C260	
3:0	Type	0x2	



6.2.18.6.2 Starting address high at PRTMAC_PCS_LINK_CTRL[PRT] (0x000D + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_PCS_LINK_CTRL, for PRT[0]		

6.2.18.6.3 Attributes at PRTMAC_PCS_LINK_CTRL[PRT] (0x000E + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.6.4 Data low of PRTMAC_PCS_LINK_CTRL[PRT] (0x000F + 2*PRT, PRT=0...3)

6.2.18.6.5 Data high of PRTMAC_PCS_LINK_CTRL[PRT] (0x0010 + 2*PRT, PRT=0...3)

6.2.18.7 PRTMAC_PCS_XAUI_SWAP_A - (0x0017 - 0x001B)

6.2.18.7.1 Starting address low at PRTMAC_PCS_XAUI_SWAP_A - 0x0017

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_PCS_XAUI_SWAP_A	0x8C480	
3:0	Type	0x2	

6.2.18.7.2 Starting address high at PRTMAC_PCS_XAUI_SWAP_A - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_PCS_XAUI_SWAP_A		

6.2.18.7.3 Attributes at PRTMAC_PCS_XAUI_SWAP_A - 0x0019

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.7.4 Data low of PRTMAC_PCS_XAUI_SWAP_A - 0x001A

6.2.18.7.5 Data high of PRTMAC_PCS_XAUI_SWAP_A - 0x001B

6.2.18.8 PRTMAC_PCS_XAUI_SWAP_B - (0x001C - 0x001D)

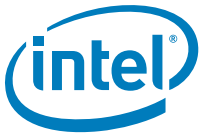
6.2.18.8.1 Data low of PRTMAC_PCS_XAUI_SWAP_B - 0x001C

6.2.18.8.2 Data high of PRTMAC_PCS_XAUI_SWAP_B - 0x001D

6.2.18.9 PRTMAC_PCS_AN_CONTROL1 - (0x001E - 0x0028)

6.2.18.9.1 Starting address low at PRTMAC_PCS_AN_CONTROL1[PRT] (0x001E + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_PCS_AN_CONTROL1, for PRT[0]	0x8C600	
3:0	Type	0x2	



6.2.18.9.2 Starting address high at PRTMAC_PCS_AN_CONTROL1[PRT] (0x001F + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_PCS_AN_CONTROL1, for PRT[0]		

6.2.18.9.3 Attributes at PRTMAC_PCS_AN_CONTROL1[PRT] (0x0020 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.9.4 Data low of PRTMAC_PCS_AN_CONTROL1[PRT] (0x0021 + 2*PRT, PRT=0...3)

6.2.18.9.5 Data high of PRTMAC_PCS_AN_CONTROL1[PRT] (0x0022 + 2*PRT, PRT=0...3)

6.2.18.10 PRTMAC_PCS_AN_CONTROL2 - (0x0029 - 0x0033)

6.2.18.10.1 Starting address low at PRTMAC_PCS_AN_CONTROL2[PRT] (0x0029 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_PCS_AN_CONTROL2, for PRT[0]	0x8C620	
3:0	Type	0x2	

6.2.18.10.2 Starting address high at PRTMAC_PCS_AN_CONTROL2[PRT] (0x002A + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_PCS_AN_CONTROL2, for PRT[0]		

6.2.18.10.3 Attributes at PRTMAC_PCS_AN_CONTROL2[PRT] (0x002B + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

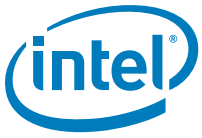
6.2.18.11 PRTMAC_PCS_AN_CONTROL4 - (0x0034 - 0x003E)

6.2.18.11.1 Starting address low at PRTMAC_PCS_AN_CONTROL4[PRT] (0x0034 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_PCS_AN_CONTROL4, for PRT[0]	0x8CCC0	
3:0	Type	0x2	

6.2.18.11.2 Starting address high at PRTMAC_PCS_AN_CONTROL4[PRT] (0x0035 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_PCS_AN_CONTROL4, for PRT[0]		



6.2.18.11.3 Attributes at PRTMAC_PCS_AN_CONTROL4[PRT] (0x0036 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.11.4 Data low of PRTMAC_PCS_AN_CONTROL4[PRT] (0x0037 + 2*PRT, PRT=0...3)

6.2.18.11.5 Data high of PRTMAC_PCS_AN_CONTROL4[PRT] (0x0038 + 2*PRT, PRT=0...3)

6.2.18.12 PRTMAC_HLCTL - (0x003F - 0x0049)

6.2.18.12.1 Starting address low at PRTMAC_HLCTL[PRT] (0x003F + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HLCTL, for PRT[0]	0x1E2000	
3:0	Type	0x2	

6.2.18.12.2 Starting address high at PRTMAC_HLCTL[PRT] (0x0040 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HLCTL, for PRT[0]		

6.2.18.12.3 Attributes at PRTMAC_HLCTL[PRT] (0x0041 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	

Bits	Field Name	Default NVM Value	Description
2:0	Width	0x0	

6.2.18.12.4 Data low of PRTMAC_HLCTL[PRT] (0x0042 + 2*PRT, PRT=0...3)

6.2.18.12.5 Data high of PRTMAC_HLCTL[PRT] (0x0043 + 2*PRT, PRT=0...3)

6.2.18.13 PRTMAC_PAP - (0x004A - 0x0054)

6.2.18.13.1 Starting address low at PRTMAC_PAP[PRT] (0x004A + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_PAP, for PRT[0]	0x1E2040	
3:0	Type	0x2	

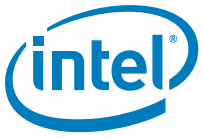
6.2.18.13.2 Starting address high at PRTMAC_PAP[PRT] (0x004B + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_PAP, for PRT[0]		

6.2.18.13.3 Attributes at PRTMAC_PAP[PRT] (0x004C + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.13.4 Data low of PRTMAC_PAP[PRT] (0x004D + 2*PRT, PRT=0...3)



6.2.18.13.5 Data high of PRTMAC_PAP[PRT] (0x004E + 2*PRT, PRT=0...3)

6.2.18.14 PRTGL_SAL - (0x0055 - 0x005F)

6.2.18.14.1 Starting address low at PRTGL_SAL[PRT] (0x0055 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTGL_SAL, for PRT[0]	0x1E2120	
3:0	Type	0x2	

6.2.18.14.2 Starting address high at PRTGL_SAL[PRT] (0x0056 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTGL_SAL, for PRT[0]		

6.2.18.14.3 Attributes at PRTGL_SAL[PRT] (0x0057 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.14.4 Data low of PRTGL_SAL[PRT] (0x0058 + 2*PRT, PRT=0...3)

6.2.18.14.5 Data high of PRTGL_SAL[PRT] (0x0059 + 2*PRT, PRT=0...3)

6.2.18.15 PRTGL_SAH - (0x0060 - 0x006A)

6.2.18.15.1 Starting address low at PRTGL_SAH[PRT] (0x0060 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTGL_SAH, for PRT[0]	0x1E2140	
3:0	Type	0x2	

6.2.18.15.2 Starting address high at PRTGL_SAH[PRT] (0x0061 + 2*PRT, PRT=0)

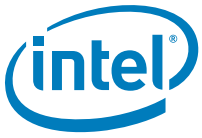
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTGL_SAH, for PRT[0]		

6.2.18.15.3 Attributes at PRTGL_SAH[PRT] (0x0062 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.15.4 Data low of PRTGL_SAH[PRT] (0x0063 + 2*PRT, PRT=0...3)

6.2.18.15.5 Data high of PRTGL_SAH[PRT] (0x0064 + 2*PRT, PRT=0...3)



6.2.18.16 PRTDCB_MFLCN - (0x006B - 0x0075)

6.2.18.16.1 Starting address low at PRTDCB_MFLCN[PRT] (0x006B + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_MFLCN, for PRT[0]	0x1E2400	
3:0	Type	0x2	

6.2.18.16.2 Starting address high at PRTDCB_MFLCN[PRT] (0x006C + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_MFLCN, for PRT[0]		

6.2.18.16.3 Attributes at PRTDCB_MFLCN[PRT] (0x006D + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.16.4 Data low of PRTDCB_MFLCN[PRT] (0x006E + 2*PRT, PRT=0...3)

6.2.18.16.5 Data high of PRTDCB_MFLCN[PRT] (0x006F + 2*PRT, PRT=0...3)

6.2.18.17 PRTMAC_HSEC_CTL_RX_MAX_PACKET_LEN - (0x0076 - 0x007C)

6.2.18.17.1 Starting address low at PRTMAC_HSEC_CTL_RX_MAX_PACKET_LEN[PRT2] (0x0076 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_MAX_PACKET_LEN, for PRT2[0]	0x1E30A0	
3:0	Type	0x2	

6.2.18.17.2 Starting address high at PRTMAC_HSEC_CTL_RX_MAX_PACKET_LEN[PRT2] (0x0077 + 2*PRT2, PRT2=0)

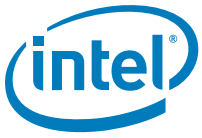
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_MAX_PACKET_LEN, for PRT2[0]		

6.2.18.17.3 Attributes at PRTMAC_HSEC_CTL_RX_MAX_PACKET_LEN[PRT2] (0x0078 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.17.4 Data low of PRTMAC_HSEC_CTL_RX_MAX_PACKET_LEN[PRT2] (0x0079 + 2*PRT2, PRT2=0...1)

6.2.18.17.5 Data high of PRTMAC_HSEC_CTL_RX_MAX_PACKET_LEN[PRT2] (0x007A + 2*PRT2, PRT2=0...1)



6.2.18.18 PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE - (0x007D - 0x0083)

6.2.18.18.1 Starting address low at PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE[PRT2] (0x007D + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE, for PRT2[0]	0x1E30C0	
3:0	Type	0x2	

6.2.18.18.2 Starting address high at PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE[PRT2] (0x007E + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE, for PRT2[0]		

6.2.18.18.3 Attributes at PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE[PRT2] (0x007F + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.18.4 Data low of PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE[PRT2] (0x0080 + 2*PRT2, PRT2=0...1)

6.2.18.18.5 Data high of PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE[PRT2] (0x0081 + 2*PRT2, PRT2=0...1)

6.2.18.19 PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE - (0x0084 - 0x008A)

6.2.18.19.1 Starting address low at PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE[PRT2] (0x0084 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE, for PRT2[0]	0x1E30D0	
3:0	Type	0x2	

6.2.18.19.2 Starting address high at PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE[PRT2] (0x0085 + 2*PRT2, PRT2=0)

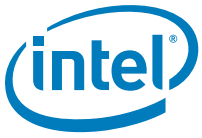
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE, for PRT2[0]		

6.2.18.19.3 Attributes at PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE[PRT2] (0x0086 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.19.4 Data low of PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE[PRT2] (0x0087 + 2*PRT2, PRT2=0...1)

6.2.18.19.5 Data high of PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE[PRT2] (0x0088 + 2*PRT2, PRT2=0...1)



6.2.18.20 PRTMAC_HSEC_CTL_RX_ENABLE_GCP - (0x008B - 0x0091)

6.2.18.20.1 Starting address low at PRTMAC_HSEC_CTL_RX_ENABLE_GCP[PRT2] (0x008B + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_ENABLE_GCP, for PRT2[0]	0x1E30E0	
3:0	Type	0x2	

6.2.18.20.2 Starting address high at PRTMAC_HSEC_CTL_RX_ENABLE_GCP[PRT2] (0x008C + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_ENABLE_GCP, for PRT2[0]		

6.2.18.20.3 Attributes at PRTMAC_HSEC_CTL_RX_ENABLE_GCP[PRT2] (0x008D + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.20.4 Data low of PRTMAC_HSEC_CTL_RX_ENABLE_GCP[PRT2] (0x008E + 2*PRT2, PRT2=0...1)

6.2.18.20.5 Data high of PRTMAC_HSEC_CTL_RX_ENABLE_GCP[PRT2] (0x008F + 2*PRT2, PRT2=0...1)

6.2.18.21 PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP - (0x0092 - 0x0098)

6.2.18.21.1 Starting address low at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP[PRT2] (0x0092 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP, for PRT2[0]	0x1E3100	
3:0	Type	0x2	

6.2.18.21.2 Starting address high at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP[PRT2] (0x0093 + 2*PRT2, PRT2=0)

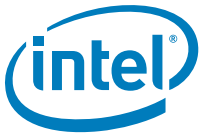
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP, for PRT2[0]		

6.2.18.21.3 Attributes at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP[PRT2] (0x0094 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.21.4 Data low of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP[PRT2] (0x0095 + 2*PRT2, PRT2=0...1)

6.2.18.21.5 Data high of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP[PRT2] (0x0096 + 2*PRT2, PRT2=0...1)



6.2.18.22 PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1 - (0x0099 - 0x009F)

6.2.18.22.1 Starting address low at PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1[PRT2] (0x0099 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1, for PRT2[0]	0x1E3110	
3:0	Type	0x2	

6.2.18.22.2 Starting address high at PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1[PRT2] (0x009A + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1, for PRT2[0]		

6.2.18.22.3 Attributes at PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1[PRT2] (0x009B + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.22.4 Data low of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1[PRT2] (0x009C + 2*PRT2, PRT2=0...1)

6.2.18.22.5 Data high of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1[PRT2] (0x009D + 2*PRT2, PRT2=0...1)

6.2.18.23 PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2 - (0x00A0 - 0x00A6)

6.2.18.23.1 Starting address low at PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2[PRT2] (0x00A0 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2, for PRT2[0]	0x1E3120	
3:0	Type	0x2	

6.2.18.23.2 Starting address high at PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2[PRT2] (0x00A1 + 2*PRT2, PRT2=0)

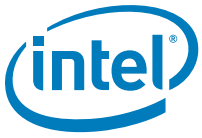
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2, for PRT2[0]		

6.2.18.23.3 Attributes at PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2[PRT2] (0x00A2 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.23.4 Data low of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2[PRT2] (0x00A3 + 2*PRT2, PRT2=0...1)

6.2.18.23.5 Data high of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2[PRT2] (0x00A4 + 2*PRT2, PRT2=0...1)



6.2.18.24 PRTMAC_HSEC_CTL_RX_CHECK_SA_GCP - (0x00A7 - 0x00AD)

6.2.18.24.1 Starting address low at PRTMAC_HSEC_CTL_RX_CHECK_SA_GCP[PRT2] (0x00A7 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_CHECK_SA_GCP, for PRT2[0]	0x1E3130	
3:0	Type	0x2	

6.2.18.24.2 Starting address high at PRTMAC_HSEC_CTL_RX_CHECK_SA_GCP[PRT2] (0x00A8 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_CHECK_SA_GCP, for PRT2[0]		

6.2.18.24.3 Attributes at PRTMAC_HSEC_CTL_RX_CHECK_SA_GCP[PRT2] (0x00A9 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.24.4 Data low of PRTMAC_HSEC_CTL_RX_CHECK_SA_GCP[PRT2] (0x00AA + 2*PRT2, PRT2=0...1)

6.2.18.24.5 Data high of PRTMAC_HSEC_CTL_RX_CHECK_SA_GCP[PRT2] (0x00AB + 2*PRT2, PRT2=0...1)

6.2.18.25 PRTMAC_HSEC_CTL_RX_ENABLE_GPP - (0x00AE - 0x00B4)

6.2.18.25.1 Starting address low at PRTMAC_HSEC_CTL_RX_ENABLE_GPP[PRT2] (0x00AE + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_ENABLE_GPP, for PRT2[0]	0x1E3260	
3:0	Type	0x2	

6.2.18.25.2 Starting address high at PRTMAC_HSEC_CTL_RX_ENABLE_GPP[PRT2] (0x00AF + 2*PRT2, PRT2=0)

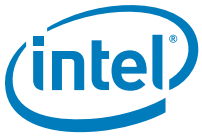
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_ENABLE_GPP, for PRT2[0]		

6.2.18.25.3 Attributes at PRTMAC_HSEC_CTL_RX_ENABLE_GPP[PRT2] (0x00B0 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.25.4 Data low of PRTMAC_HSEC_CTL_RX_ENABLE_GPP[PRT2] (0x00B1 + 2*PRT2, PRT2=0...1)

6.2.18.25.5 Data high of PRTMAC_HSEC_CTL_RX_ENABLE_GPP[PRT2] (0x00B2 + 2*PRT2, PRT2=0...1)



6.2.18.26 PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GPP - (0x00B5 - 0x00BB)

6.2.18.26.1 Starting address low at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GPP[PRT2] (0x00B5 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GPP, for PRT2[0]	0x1E3280	
3:0	Type	0x2	

6.2.18.26.2 Starting address high at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GPP[PRT2] (0x00B6 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GPP, for PRT2[0]		

6.2.18.26.3 Attributes at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GPP[PRT2] (0x00B7 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.26.4 Data low of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GPP[PRT2] (0x00B8 + 2*PRT2, PRT2=0...1)

6.2.18.26.5 Data high of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GPP[PRT2] (0x00B9 + 2*PRT2, PRT2=0...1)

6.2.18.27 PRTMAC_HSEC_CTL_RX_ENABLE_PPP - (0x00BC - 0x00C2)

6.2.18.27.1 Starting address low at PRTMAC_HSEC_CTL_RX_ENABLE_PPP[PRT2] (0x00BC + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_ENABLE_PPP, for PRT2[0]	0x1E32E0	
3:0	Type	0x2	

6.2.18.27.2 Starting address high at PRTMAC_HSEC_CTL_RX_ENABLE_PPP[PRT2] (0x00BD + 2*PRT2, PRT2=0)

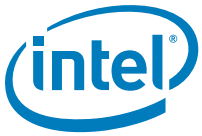
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_ENABLE_PPP, for PRT2[0]		

6.2.18.27.3 Attributes at PRTMAC_HSEC_CTL_RX_ENABLE_PPP[PRT2] (0x00BE + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.27.4 Data low of PRTMAC_HSEC_CTL_RX_ENABLE_PPP[PRT2] (0x00BF + 2*PRT2, PRT2=0...1)

6.2.18.27.5 Data high of PRTMAC_HSEC_CTL_RX_ENABLE_PPP[PRT2] (0x00C0 + 2*PRT2, PRT2=0...1)



6.2.18.28 PRTMAC_HSEC_CTL_RX_CHECK_UCAST_PPP - (0x00C3 - 0x00C9)

6.2.18.28.1 Starting address low at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_PPP[PRT2] (0x00C3 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_PPP, for PRT2[0]	0x1E3300	
3:0	Type	0x2	

6.2.18.28.2 Starting address high at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_PPP[PRT2] (0x00C4 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_PPP, for PRT2[0]		

6.2.18.28.3 Attributes at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_PPP[PRT2] (0x00C5 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.28.4 Data low of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_PPP[PRT2] (0x00C6 + 2*PRT2, PRT2=0...1)

6.2.18.28.5 Data high of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_PPP[PRT2] (0x00C7 + 2*PRT2, PRT2=0...1)

6.2.18.29 PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL - (0x00CA - 0x00D0)

6.2.18.29.1 Starting address low at PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL[PRT2] (0x00CA + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL, for PRT2[0]	0x1E3360	
3:0	Type	0x2	

6.2.18.29.2 Starting address high at PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL[PRT2] (0x00CB + 2*PRT2, PRT2=0)

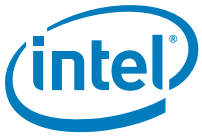
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL, for PRT2[0]		

6.2.18.29.3 Attributes at PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL[PRT2] (0x00CC + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.29.4 Data low of PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL[PRT2] (0x00CD + 2*PRT2, PRT2=0...1)

6.2.18.29.5 Data high of PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL[PRT2] (0x00CE + 2*PRT2, PRT2=0...1)



6.2.18.30 PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA - (0x00D1 - 0x010F)

6.2.18.30.1 Starting address low at PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n,PRT2] (0x00D1 + 7*n, n=0...8)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA, for PRT2[0]	0x1E3370	
3:0	Type	0x2	

6.2.18.30.2 Starting address high at PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n,PRT2] (0x00D2 + 7*n, n=0...8)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA, for PRT2[0]		

6.2.18.30.3 Attributes at PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n,PRT2] (0x00D3 + 7*n, n=0...8)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.30.4 Data low of PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n,PRT2] (0x00D4 + 7*n + 2*PRT2, n=0...8, PRT2=0...1)

6.2.18.30.5 Data high of PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n,PRT2] (0x00D5 + 7*n + 2*PRT2, n=0...8, PRT2=0...1)

6.2.18.31 PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER - (0x0110 - 0x014E)

6.2.18.31.1 Starting address low at PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n, PRT2] (0x0110 + 7*n, n=0...8)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER, for PRT2[0]	0x1E3400	
3:0	Type	0x2	

6.2.18.31.2 Starting address high at PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n, PRT2] (0x0111 + 7*n, n=0...8)

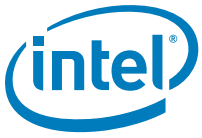
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER, for PRT2[0]		

6.2.18.31.3 Attributes at PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n, PRT2] (0x0112 + 7*n, n=0...8)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.31.4 Data low of PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n, PRT2] (0x0113 + 7*n + 2*PRT2, n=0...8, PRT2=0...1)

6.2.18.31.5 Data high of PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n,



PRT2] (0x0114 + 7*n + 2*PRT2, n=0...8, PRT2=0...1)

6.2.18.32 PRTMAC_HSEC_CTL_TX_SA_PART1 - (0x014F - 0x0155)

6.2.18.32.1 Starting address low at PRTMAC_HSEC_CTL_TX_SA_PART1[PRT2] (0x014F + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_TX_SA_PART1, for PRT2[0]	0x1E34B0	
3:0	Type	0x2	

6.2.18.32.2 Starting address high at PRTMAC_HSEC_CTL_TX_SA_PART1[PRT2] (0x0150 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_TX_SA_PART1, for PRT2[0]		

6.2.18.32.3 Attributes at PRTMAC_HSEC_CTL_TX_SA_PART1[PRT2] (0x0151 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.32.4 Data low of PRTMAC_HSEC_CTL_TX_SA_PART1[PRT2] (0x0152 + 2*PRT2, PRT2=0...1)

6.2.18.32.5 Data high of PRTMAC_HSEC_CTL_TX_SA_PART1[PRT2] (0x0153 + 2*PRT2, PRT2=0...1)

6.2.18.33 PRTMAC_HSEC_CTL_TX_SA_PART2 - (0x0156 - 0x015C)

6.2.18.33.1 Starting address low at PRTMAC_HSEC_CTL_TX_SA_PART2[PRT2] (0x0156 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_TX_SA_PART2, for PRT2[0]	0x1E34C0	
3:0	Type	0x2	

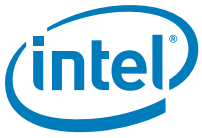
6.2.18.33.2 Starting address high at PRTMAC_HSEC_CTL_TX_SA_PART2[PRT2] (0x0157 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_TX_SA_PART2, for PRT2[0]		

6.2.18.33.3 Attributes at PRTMAC_HSEC_CTL_TX_SA_PART2[PRT2] (0x0158 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.33.4 Data low of PRTMAC_HSEC_CTL_TX_SA_PART2[PRT2] (0x0159 + 2*PRT2, PRT2=0...1)



6.2.18.33.5 Data high of PRTMAC_HSEC_CTL_TX_SA_PART2[PRT2] (0x015A + 2*PRT2, PRT2=0...1)

6.2.18.34 PRTMAC_HSEC_CTL_INTERNAL - (0x015D - 0x0163)

6.2.18.34.1 Starting address low at PRTMAC_HSEC_CTL_INTERNAL[PRT2] (0x015D + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_INTERNAL, for PRT2[0]	0x1E3530	
3:0	Type	0x2	

6.2.18.34.2 Starting address high at PRTMAC_HSEC_CTL_INTERNAL[PRT2] (0x015E + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_INTERNAL, for PRT2[0]		

6.2.18.34.3 Attributes at PRTMAC_HSEC_CTL_INTERNAL[PRT2] (0x015F + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.34.4 Data low of PRTMAC_HSEC_CTL_INTERNAL[PRT2] (0x0160 + 2*PRT2, PRT2=0...1)

6.2.18.34.5 Data high of PRTMAC_HSEC_CTL_INTERNAL[PRT2] (0x0161 + 2*PRT2, PRT2=0...1)

6.2.18.35 PRTMAC_HSEC_SINGLE_40G_PORT_SELECT - (0x0164 - 0x0167)

6.2.18.35.1 Address high at PRTMAC_HSEC_SINGLE_40G_PORT_SELECT - 0x0165

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_SINGLE_40G_PORT_SELECT		

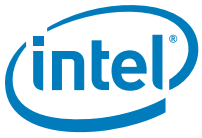
6.2.18.35.2 Data low of PRTMAC_HSEC_SINGLE_40G_PORT_SELECT - 0x0166

6.2.18.35.3 Data high of PRTMAC_HSEC_SINGLE_40G_PORT_SELECT - 0x0167

6.2.18.36 PRTMAC_HSEC_CTL_XLGMII - (0x0168 - 0x016E)

6.2.18.36.1 Starting address low at PRTMAC_HSEC_CTL_XLGMII[PRT2] (0x0168 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_XLGMII, for PRT2[0]	0x1E3550	
3:0	Type	0x2	



6.2.18.36.2 Starting address high at PRTMAC_HSEC_CTL_XLGMII[PRT2] (0x0169 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_XLGMII, for PRT2[0]		

6.2.18.36.3 Attributes at PRTMAC_HSEC_CTL_XLGMII[PRT2] (0x016A + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.36.4 Data low of PRTMAC_HSEC_CTL_XLGMII[PRT2] (0x016B + 2*PRT2, PRT2=0...1)

6.2.18.36.5 Data high of PRTMAC_HSEC_CTL_XLGMII[PRT2] (0x016C + 2*PRT2, PRT2=0...1)

6.2.18.37 PRTMAC_HSECTL1 - (0x016F - 0x0175)

6.2.18.37.1 Starting address low at PRTMAC_HSECTL1[PRT2] (0x016F + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSECTL1, for PRT2[0]	0x1E3560	
3:0	Type	0x2	

6.2.18.37.2 Starting address high at PRTMAC_HSECTL1[PRT2] (0x0170 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSECTL1, for PRT2[0]		

6.2.18.37.3 Attributes at PRTMAC_HSECTL1[PRT2] (0x0171 + 2*PRT2, PRT2=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.37.4 Data low of PRTMAC_HSECTL1[PRT2] (0x0172 + 2*PRT2, PRT2=0...1)

6.2.18.37.5 Data high of PRTMAC_HSECTL1[PRT2] (0x0173 + 2*PRT2, PRT2=0...1)

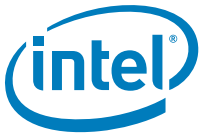
6.2.18.38 PRTTSYN_CTL0 - (0x0176 - 0x0180)

6.2.18.38.1 Starting address low at PRTTSYN_CTL0[PRT] (0x0176 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTTSYN_CTL0, for PRT[0]	0x1E4200	
3:0	Type	0x2	

6.2.18.38.2 Starting address high at PRTTSYN_CTL0[PRT] (0x0177 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTTSYN_CTL0, for PRT[0]		



6.2.18.38.3 Attributes at PRTTSYN_CTL0[PRT] (0x0178 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.38.4 Data low of PRTTSYN_CTL0[PRT] (0x0179 + 2*PRT, PRT=0...3)

6.2.18.38.5 Data high of PRTTSYN_CTL0[PRT] (0x017A + 2*PRT, PRT=0...3)

6.2.18.39 GLPM_EEE_SU - (0x0181 - 0x0185)

6.2.18.39.1 Starting address low at GLPM_EEE_SU - 0x0181

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPM_EEE_SU	0x1E4340	
3:0	Type	0x2	

6.2.18.39.2 Starting address high at GLPM_EEE_SU - 0x0182

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPM_EEE_SU		

6.2.18.39.3 Attributes at GLPM_EEE_SU - 0x0183

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.39.4 Data low of GLPM_EEE_SU - 0x0184

6.2.18.39.5 Data high of GLPM_EEE_SU - 0x0185

6.2.18.40 GLPM_EEE_SU_EXT - (0x0186 - 0x0187)

6.2.18.40.1 Data low of GLPM_EEE_SU_EXT - 0x0186

6.2.18.40.2 Data high of GLPM_EEE_SU_EXT - 0x0187

6.2.18.41 PRTPM_EEER - (0x0188 - 0x0192)

6.2.18.41.1 Starting address low at PRTPM_EEER[PRT] (0x0188 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTPM_EEER, for PRT[0]	0x1E4360	
3:0	Type	0x2	

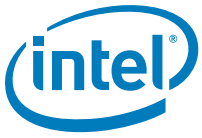
6.2.18.41.2 Starting address high at PRTPM_EEER[PRT] (0x0189 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTPM_EEER, for PRT[0]		

6.2.18.41.3 Attributes at PRTPM_EEER[PRT] (0x018A + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.41.4 Data low of PRTPM_EEER[PRT] (0x018B + 2*PRT, PRT=0...3)



6.2.18.41.5 Data high of PRTPM_EEER[PRT] (0x018C + 2*PRT, PRT=0...3)

6.2.18.42 PRTPM_EEEC - (0x0193 - 0x019D)

6.2.18.42.1 Starting address low at PRTPM_EEEC[PRT] (0x0193 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTPM_EEEC, for PRT[0]	0x1E4380	
3:0	Type	0x2	

6.2.18.42.2 Starting address high at PRTPM_EEEC[PRT] (0x0194 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTPM_EEEC, for PRT[0]		

6.2.18.42.3 Attributes at PRTPM_EEEC[PRT] (0x0195 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.42.4 Data low of PRTPM_EEEC[PRT] (0x0196 + 2*PRT, PRT=0...3)

6.2.18.42.5 Data high of PRTPM_EEEC[PRT] (0x0197 + 2*PRT, PRT=0...3)

6.2.18.43 PRTDCB_FCTTVN - (0x019E - 0x01C9)

6.2.18.43.1 Starting address low at PRTDCB_FCTTVN[n,PRT] (0x019E + 11*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_FCTTVN, for PRT[0]	0x1E4580	
3:0	Type	0x2	

6.2.18.43.2 Starting address high at PRTDCB_FCTTVN[n,PRT] (0x019F + 11*n, n=0...3)

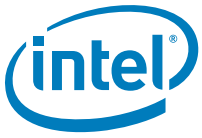
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_FCTTVN, for PRT[0]		

6.2.18.43.3 Attributes at PRTDCB_FCTTVN[n,PRT] (0x01A0 + 11*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.43.4 Data low of PRTDCB_FCTTVN[n,PRT] (0x01A1 + 11*n + 2*PRT, n=0...3, PRT=0...3)

6.2.18.43.5 Data high of PRTDCB_FCTTVN[n,PRT] (0x01A2 + 11*n + 2*PRT, n=0...3, PRT=0...3)



6.2.18.44 PRTDCB_FCRTV - (0x01CA - 0x01D4)

6.2.18.44.1 Starting address low at PRTDCB_FCRTV[PRT] (0x01CA + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_FCRTV, for PRT[0]	0x1E4600	
3:0	Type	0x2	

6.2.18.44.2 Starting address high at PRTDCB_FCRTV[PRT] (0x01CB + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_FCRTV, for PRT[0]		

6.2.18.44.3 Attributes at PRTDCB_FCRTV[PRT] (0x01CC + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.44.4 Data low of PRTDCB_FCRTV[PRT] (0x01CD + 2*PRT, PRT=0...3)

6.2.18.44.5 Data high of PRTDCB_FCRTV[PRT] (0x01CE + 2*PRT, PRT=0...3)

6.2.18.45 PRTDCB_FCCFG - (0x01D5 - 0x01DF)

6.2.18.45.1 Starting address low at PRTDCB_FCCFG[PRT] (0x01D5 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_FCCFG, for PRT[0]	0x1E4640	
3:0	Type	0x2	

6.2.18.45.2 Starting address high at PRTDCB_FCCFG[PRT] (0x01D6 + 2*PRT, PRT=0)

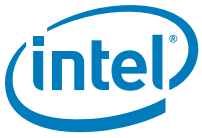
Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_FCCFG, for PRT[0]		

6.2.18.45.3 Attributes at PRTDCB_FCCFG[PRT] (0x01D7 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

6.2.18.45.4 Data low of PRTDCB_FCCFG[PRT] (0x01D8 + 2*PRT, PRT=0...3)

6.2.18.45.5 Data high of PRTDCB_FCCFG[PRT] (0x01D9 + 2*PRT, PRT=0...3)



6.2.18.46 PRTMAC_HLCTLA - (0x01E0 - 0x01EA)

6.2.18.46.1 Starting address low at PRTMAC_HLCTLA[PRT] (0x01E0 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HLCTLA, for PRT[0]	0x1E4760	
3:0	Type	0x2	

6.2.18.46.2 Starting address high at PRTMAC_HLCTLA[PRT] (0x01E1 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HLCTLA, for PRT[0]		

6.2.18.46.3 Attributes at PRTMAC_HLCTLA[PRT] (0x01E2 + 2*PRT, PRT=0)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x4	
4:3	Skip	0x0	
2:0	Width	0x0	

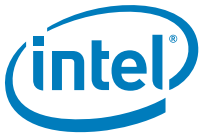
6.2.18.46.4 Data low of PRTMAC_HLCTLA[PRT] (0x01E3 + 2*PRT, PRT=0...3)

6.2.18.46.5 Data high of PRTMAC_HLCTLA[PRT] (0x01E4 + 2*PRT, PRT=0...3)

6.2.19 EMP SR settings module Header section summary table

This section contains the modes of operation of the EMP that must be stored in shadow RAM.

Word Offset	Description	Reference
0x0000	Section Header	493
0x0001	SMBus Slave Addresses	493
0x0002	SMBus Slave Addresses 2	493
0x0003	OEM Capabilities	493
0x0004	OEM Technologies Enabled	494
0x0005	SR PF Allocations Pointer	494
0x0006	Max PF and VF Per Port	495
0x0007	PF LAN Device ID	495
0x0008	PF SAN Device ID	495
0x0009	VF LAN Device ID	495
0x000A	VF SAN Device ID	495
0x000B	PF LAN Subsystem ID	496
0x000C	PF SAN Subsystem ID	496
0x000D	VF LAN Subsystem ID	496
0x000E	VF SAN Subsystem ID	496



6.2.19.1 Section header - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 bytes unit - 1. First section -> Word: EMP SR Settings Module Header -> Section Header. Last section -> Word: EMP SR Settings Module Header -> VF SAN Subsystem ID. Section length in words.

6.2.19.2 SMBus slave addresses - 0x0001

Bits	Field Name	Default NVM Value	Description
15:9	SMBus 1 slave Address	0x4A	Dual address mode only.
8	Reserved	0x0	
7:1	SMBus 0 slave Address	0x49	
0	Reserved	0x0	

6.2.19.3 SMBus slave addresses 2 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:9	SMBus 3 Slave Addresses	0x4C	
8	res	0x0	
7:1	SMBus 2 Slave Addresses	0x4B	
0	res	0x0	

6.2.19.4 OEM capabilities - 0x0003

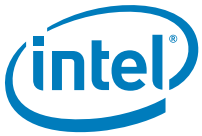
Defines the OEM technologies supported in the XL710.

Bits	Field Name	Default NVM Value	Description
15:4	Reserved	0x0	Reserved

6.2.19.5 SR PF allocations pointer - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	PF allocations pointer	0xFFFF	

The field PF allocations pointer points to SR PF allocations section. For more details about the SR PF allocations inner structure, see 496.



6.2.19.6 Max PF and VF per port - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Max VF per PF	0x80	Maximum number of VFs allocated to a PF.
7:3	Reserved	0x00	
2:0	Max PF per Port	0x2	Valid values are: 0x0 = 1 PF per Port. 0x1 = 2 PFs per Port. 0x2 = 4 PFs per Port. 0x3 = 8 PFs per Port. 0x4 = 16 PFs per Port. 0x5 = Reserved. 0x6 = Reserved. 0x7 = Reserved.

6.2.19.7 PF LAN device ID - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	PF LAN Device ID	0x154B	

6.2.19.8 PF SAN device ID - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	PF SAN Device ID	0x154B	

6.2.19.9 VF LAN device ID - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	VF LAN Device ID	0x154C	

6.2.19.10 VF SAN device ID - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	VF SAN Device ID	0x154C	

6.2.19.11 PF LAN subsystem ID - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	PF LAN Subsystem ID	0x0	

6.2.19.12 PF SAN subsystem ID - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	PF SAN Subsystem ID	0x0	

6.2.19.13 VF LAN subsystem ID - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	VF LAN Subsystem ID	0x0	

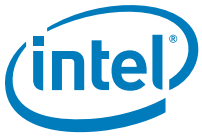
6.2.19.14 VF SAN subsystem ID - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	VF SAN Subsystem ID	0x0	

6.2.20 SR PF allocations section summary table

This section contains the allocation of resources per PF that must be stored in shadow RAM.

Word Offset	Description	Reference
0x0000	Section Header	497
0x0001 + 2*n, n=0...15	PF Flags	497
0x0002 + 2*n, n=0...15	PF BW	497



6.2.20.1 Section header - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 bytes unit - 1. First section -> Word: SR PF Allocations -> Section Header. Last section -> Word: SR PF Allocations -> PF BW. Section length in words.

6.2.20.2 PF Flags[n] (0x0001 + 2*n, n=0...15)

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	Reserved.
1	FCoE	0x0	FCoE Personality Bit. Valid values are: 0x0 = LAN. 0x1 = FCoE.
0	Reserved	0x1	Reserved.

6.2.20.3 PF BW[n] (0x0002 + 2*n, n=0...15)

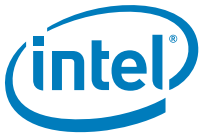
Bits	Field Name	Default NVM Value	Description
15:8	PF Max BW	0x64	This field contains the maximum Tx bandwidth allocation of the specified partition expressed in percent of the maximum physical port link speed. The percent value ranges from 0 to 100.
7:0	PF Min BW	0x0	This field contains the minimum Tx bandwidth allocation of the specified partition expressed in percent of the maximum physical port link speed. The percent value ranges from 0 to 100.

6.2.21 Auto generated pointers module section summary sable

Pointers to Type 1/2 words used by the EMP and software.

Word Offset	Description	Reference
0x7D80	Module Length	499
0x7D81	Pointer to PFPM_APM Section	499
0x7D82	Pointer to PFPM_APM Offset	499
0x7D83	Pointer to PRTPM_GC Section	499
0x7D84	Pointer to PRTPM_GC Offset	499
0x7D85	Pointer to GLGEN_STAT Section	499

Word Offset	Description	Reference
0x7D86	Pointer to GLGEN_STAT Offset	500
0x7D87	Pointer to GLPCI_SERL Section	500
0x7D88	Pointer to GLPCI_SERL Offset	500
0x7D89	Pointer to GLPCI_SERH Section	500
0x7D8A	Pointer to GLPCI_SERH Offset	500
0x7D8B	Pointer to PRTGL_SAL Section	500
0x7D8C	Pointer to PRTGL_SAL Offset	501
0x7D8D	Pointer to PRTGL_SAH Section	501
0x7D8E	Pointer to PRTGL_SAH Offset	501
0x7D8F	Pointer to GLPCI_CAPSUP Section	501
0x7D90	Pointer to GLPCI_CAPSUP Offset	501
0x7D91	Pointer to PRTDCB_MFLCN Section	501
0x7D92	Pointer to PRTDCB_MFLCN Offset	502
0x7D93	Pointer to PRTDCB_FCCFG Section	502
0x7D94	Pointer to PRTDCB_FCCFG Offset	502
0x7D95	Pointer to PFGEN_PORTNUM Section	502
0x7D96	Pointer to PFGEN_PORTNUM Offset	502
0x7D97	Pointer to PFPCI_FUNC2 Section	503
0x7D98	Pointer to PFPCI_FUNC2 Offset	503
0x7D99	Pointer to PFPCI_CLASS Section	503
0x7D9A	Pointer to PFPCI_CLASS Offset	503
0x7D9B	Pointer to PF_VT_PFALLOC_PCIE Section	503
0x7D9C	Pointer to PF_VT_PFALLOC_PCIE Offset	503
0x7D9D	Pointer to PF_VT_PFALLOC Section	504
0x7D9E	Pointer to PF_VT_PFALLOC Offset	504
0x7D9F	Pointer to GLGEN_PCIFCNCNT Section	504
0x7DA0	Pointer to GLGEN_PCIFCNCNT Offset	504
0x7DA1	Pointer to GLPCI_REVID Section	505
0x7DA2	Pointer to GLPCI_REVID Offset	505
0x7DA3	Pointer to PFPCI_DEVID Section	505
0x7DA4	Pointer to PFPCI_DEVID Offset	505
0x7DA5	Pointer to GLPCI_SUBVENID Section	505
0x7DA6	Pointer to GLPCI_SUBVENID Offset	505



6.2.21.1 Module length - 0x7D80

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1. First section -> Word: Auto Generated Pointers Module -> Module Length. Last section -> Word: Auto Generated Pointers Module -> Pointer to GLPCI_SUBVENID offset.

6.2.21.2 Pointer to PFPM_APM section - 0x7D81

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PFPM_APM section	0x0	

6.2.21.3 Pointer to PFPM_APM offset - 0x7D82

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PFPM_APM offset	0xB4	

6.2.21.4 Pointer to PRTPM_GC section - 0x7D83

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to PRTPM_GC section		

6.2.21.5 Pointer to PRTPM_GC offset - 0x7D84

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PRTPM_GC offset	0xE2	

6.2.21.6 Pointer to GLGEN_STAT section - 0x7D85

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to GLGEN_STAT section	0x0	

6.2.21.7 Pointer to GLGEN_STAT offset - 0x7D86

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to GLGEN_STAT offset	0x17	

6.2.21.8 Pointer to GLPCI_SERL section - 0x7D87

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_SERL section	0x0	

6.2.21.9 Pointer to GLPCI_SERL offset - 0x7D88

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_SERL offset	0xBC	

6.2.21.10 Pointer to GLPCI_SERH section - 0x7D89

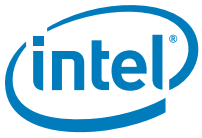
Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to GLPCI_SERH section		

6.2.21.11 Pointer to GLPCI_SERH offset - 0x7D8A

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_SERH offset	0xBE	

6.2.21.12 Pointer to PRTGL_SAL section - 0x7D8B

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PRTGL_SAL section	0x0	



6.2.21.13 Pointer to PRTGL_SAL offset - 0x7D8C

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PRTGL_SAL offset	0x57	

6.2.21.14 Pointer to PRTGL_SAH section - 0x7D8D

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to PRTGL_SAH section		

6.2.21.15 Pointer to PRTGL_SAH offset - 0x7D8E

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PRTGL_SAH offset	0x62	

6.2.21.16 Pointer to GLPCI_CAPSUP section - 0x7D8F

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to GLPCI_CAPSUP section		

6.2.21.17 Pointer to GLPCI_CAPSUP offset - 0x7D90

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_CAPSUP offset	0xC5	

6.2.21.18 Pointer to PRTDCB_MFLCN section - 0x7D91

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to PRTDCB_MFLCN section		

6.2.21.19 Pointer to PRTDCB_MFLCN offset - 0x7D92

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PRTDCB_MFLCN offset	0x6D	

6.2.21.20 Pointer to PRTDCB_FCCFG section - 0x7D93

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to PRTDCB_FCCFG section		

6.2.21.21 Pointer to PRTDCB_FCCFG offset - 0x7D94

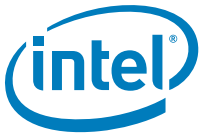
Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PRTDCB_FCCFG offset	0x1D7	

6.2.21.22 Pointer to PFGEN_PORTNUM section - 0x7D95

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PFGEN_PORTNUM section	0x0	

6.2.21.23 Pointer to PFGEN_PORTNUM offset - 0x7D96

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PFGEN_PORTNUM offset	0x208	



6.2.21.24 Pointer to PFPCI_FUNC2 section - 0x7D97

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to PFPCI_FUNC2 section		

6.2.21.25 Pointer to PFPCI_FUNC2 offset - 0x7D98

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PFPCI_FUNC2 offset	0x49	

6.2.21.26 Pointer to PFPCI_CLASS section - 0x7D99

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to PFPCI_CLASS section		

6.2.21.27 Pointer to PFPCI_CLASS offset - 0x7D9A

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PFPCI_CLASS offset	0x8F	

6.2.21.28 Pointer to PF_VT_PFALLOC_PCIE section - 0x7D9B

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to PF_VT_PFALLOC_PCIE section		

6.2.21.29 Pointer to PF_VT_PFALLOC_PCIE offset -

0x7D9C

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PF_VT_PFALLOC_PCIE offset	0x6C	

6.2.21.30 Pointer to PF_VT_PFALLOC section - 0x7D9D

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to PF_VT_PFALLOC section		

6.2.21.31 Pointer to PF_VT_PFALLOC offset - 0x7D9E

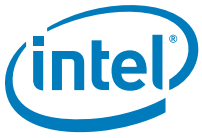
Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PF_VT_PFALLOC offset	0x22B	

6.2.21.32 Pointer to GLGEN_PCIFCNCNT section - 0x7D9F

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to GLGEN_PCIFCNCNT section		

6.2.21.33 Pointer to GLGEN_PCIFCNCNT offset - 0x7DA0

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to GLGEN_PCIFCNCNT offset	0xDD	



6.2.21.34 Pointer to GLPCI_REVID section - 0x7DA1

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to GLPCI_REVID section		

6.2.21.35 Pointer to GLPCI_REVID offset - 0x7DA2

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_REVID offset	0xCB	

6.2.21.36 Pointer to PFPCI_DEVID section - 0x7DA3

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to PFPCI_DEVID section		

6.2.21.37 Pointer to PFPCI_DEVID offset - 0x7DA4

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PFPCI_DEVID offset	0x26	

6.2.21.38 Pointer to GLPCI_SUBVENID section - 0x7DA5

Bits	Field Name	Default NVM Value	Description
15:0	Cloned pointer to GLPCI_SUBVENID section		

6.2.21.39 Pointer to GLPCI_SUBVENID offset - 0x7DA6

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_SUBVENID offset	0xB6	

6.2.21.40 Last word of shadow RAM - 0x7DFF

Bits	Field Name	Default NVM Value	Description
15:0	Last Word of Shadow RAM	0xFFFF	

6.2.21.41 Module length - 0x7E00

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit First Section -> Word: PCIe ALT Auto-load Module -> Module Length Last Section -> Word: PCIe ALT Auto-load Module -> Module Length

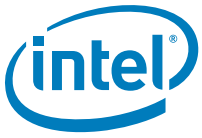
6.2.21.42 Last word of shadow RAM - 0xFFFF

Bits	Field Name	Default NVM Value	Description
15:0	Last Word of Shadow RAM	0xFFFF	

6.2.22 PCIe analog section summary table

Configures the analog section of the PCIe PHY.

Word Offset	Description	Reference
0x0000	Section Length	508
0x0001	PCIe* Analog Data	508
0x0002	moduleTypeL	509
0x0003	moduleTypeH	509
0x0004	headerLenL	509
0x0005	headerLenH	509
0x0006	headerVersionL	509
0x0007	headerVersionH	509
0x0008	moduleIDL	510
0x0009	moduleIDH	510
0x000A	moduleVendorL	510
0x000B	moduleVendorH	510
0x000C	dateL	510



Word Offset	Description	Reference
0x000D	dateH	510
0x000E	sizeL	511
0x000F	sizeH	511
0x0010	keySizeL	511
0x0011	keySizeH	511
0x0012	modulusSizeL	511
0x0013	modulusSizeH	511
0x0014	exponentSizeL	512
0x0015	exponentSizeH	512
0x0016	lad_srevL	512
0x0017	lad_srevH	512
0x0018	reserved1L	512
0x0019	reserved1H	512
0x001A	lad_fw_entry_offsetL	513
0x001B	lad_fw_entry_offsetH	513
0x001C	reserved2L	513
0x001D	reserved2H	513
0x001E	lad_image_unique_idL	513
0x001F	lad_image_unique_idH	513
0x0020	lad_module_idL	514
0x0021	lad_module_idH	514
0x0022 + 1*n, n=0...31	Reserved	514
0x0042 + 1*n, n=0...127	RSA Public Key	514
0x00C2	RSA ExponentL	514
0x00C3	RSA ExponentH	514
0x00C4 + 1*n, n=0...127	Encrypted SHA256 Hash	515
0x0144	Device Blank NVM Device ID	515
0x0145	Max Module AreaL	515
0x0146	Max Module AreaH	515
0x0147	Current Module AreaL	515
0x0148	Current Module AreaH	515
0x0149	Format Version + CRC	516
0x014A	Code Revision	516
0x014B	Reserved Spare Word	516

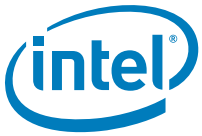
6.2.22.1 Section Length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		

6.2.22.2 PCIe* analog data - 0x0001

Raw data module length: variable.



6.2.22.3 moduleTypeL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.2.22.4 moduleTypeH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.2.22.5 headerLenL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0xa1	

6.2.22.6 headerLenH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.2.22.7 headerVersionL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.2.22.8 headerVersionH - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.2.22.9 moduleIDL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.2.22.10 moduleIDH - 0x0009

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.2.22.11 moduleVendorL - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.2.22.12 moduleVendorH - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.2.22.13 dateL - 0x000C

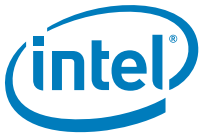
0xMMDD.

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.2.22.14 dateH - 0x000D

0xYYYY.

Bits	Field Name	Default NVM Value	Description
15:0	DateH		



6.2.22.15 sizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x00000800	

6.2.22.16 sizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.2.22.17 keySizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x40	

6.2.22.18 keySizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.2.22.19 modulusSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x40	

6.2.22.20 modulusSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.2.22.21 exponentSizeL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.2.22.22 exponentSizeH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.2.22.23 lad_srevL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.2.22.24 lad_srevH - 0x0017

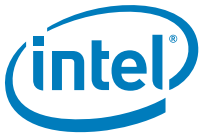
Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.2.22.25 reserved1L - 0x0018

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

6.2.22.26 reserved1H - 0x0019

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



6.2.22.27 lad_fw_entry_offsetL - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x14c	

6.2.22.28 lad_fw_entry_offsetH - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.2.22.29 reserved2L - 0x001C

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

6.2.22.30 reserved2H - 0x001D

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

6.2.22.31 lad_image_unique_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.2.22.32 lad_image_unique_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.2.22.33 lad_module_idL - 0x0020

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0x3	Valid values are: 0x1 = EMP image. 0x2 = Reserved. 0x3 = PCIe analog. 0x4 = PHY analog. 0x5 = Option ROM.

6.2.22.34 lad_module_idH - 0x0021

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.2.22.35 reserved[n] (0x0022 + 1*n, n=0...31)

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

6.2.22.36 RSA public key [n] (0x0042 + 1*n, n=0...127)

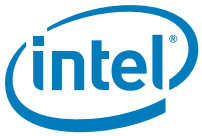
Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.2.22.37 RSA exponentL - 0x00C2

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.2.22.38 RSA exponentH - 0x00C3

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	



6.2.22.39 Encrypted SHA256 Hash[n] (0x00C4 + 1*n, n=0...127)

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.2.22.40 Device blank NVM device ID - 0x0144

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x154B	

6.2.22.41 Max module areaL - 0x0145

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x1000	

6.2.22.42 Max module areaH - 0x0146

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0000	

6.2.22.43 Current module areaL - 0x0147

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x1000	

6.2.22.44 Current module areaH - 0x0148

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	

6.2.22.45 Format version + CRC - 0x0149

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Format Version	0x2	
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: PCIe Analog -> Section Length. End Section -> Word: PCIe Analog -> Reserved Spare Word.

6.2.22.46 Code revision - 0x014A

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

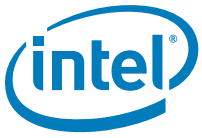
6.2.22.47 Reserved spare word - 0x014B

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

6.2.23 PHY Analog Section Summary Table

Configure the analog section of the SerDes PHY.

Word Offset	Description	Reference
0x0000	Section Length	518
0x0001	PHY Analog Data	518
0x0002	moduleTypeL	519
0x0003	moduleTypeH	519
0x0004	headerLenL	519
0x0005	headerLenH	519
0x0006	headerVersionL	519
0x0007	headerVersionH	519
0x0008	moduleIDL	520
0x0009	moduleIDH	520



Word Offset	Description	Reference
0x000A	moduleVendorL	520
0x000B	moduleVendorH	520
0x000C	dateL	520
0x000D	dateH	520
0x000E	sizeL	521
0x000F	sizeH	521
0x0010	keySizeL	521
0x0011	keySizeH	521
0x0012	modulusSizeL	521
0x0013	modulusSizeH	521
0x0014	exponentSizeL	522
0x0015	exponentSizeH	522
0x0016	lad_srevL	522
0x0017	lad_srevH	522
0x0018	reserved1L	522
0x0019	reserved1H	522
0x001A	lad_fw_entry_offsetL	523
0x001B	lad_fw_entry_offsetH	523
0x001C	reserved2L	523
0x001D	reserved2H	523
0x001E	lad_image_unique_idL	523
0x001F	lad_image_unique_idH	523
0x0020	lad_module_idL	524
0x0021	lad_module_idH	524
0x0022 + 1*n, n=0...31	Reserved	524
0x0042 + 1*n, n=0...127	RSA Public Key	524
0x00C2	RSA ExponentL	524
0x00C3	RSA ExponentH	524
0x00C4 + 1*n, n=0...127	Encrypted SHA256 Hash	525
0x0144	Device Blank NVM Device ID	525
0x0145	Max Module AreaL	525
0x0146	Max Module AreaH	525
0x0147	Current Module AreaL	525
0x0148	Current Module AreaH	525
0x0149	Format Version + CRC	526
0x014A	Code Revision	526
0x014B	Reserved Spare Word	526

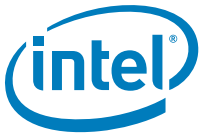
6.2.23.1 Section length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		

6.2.23.2 PHY analog data - 0x0001

Raw data module length: variable.



6.2.23.3 moduleTypeL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.2.23.4 moduleTypeH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.2.23.5 headerLenL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0xa1	

6.2.23.6 headerLenH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.2.23.7 headerVersionL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.2.23.8 headerVersionH - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.2.23.9 moduleIDL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.2.23.10 moduleIDH - 0x0009

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.2.23.11 moduleVendorL - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.2.23.12 moduleVendorH - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.2.23.13 dateL - 0x000C

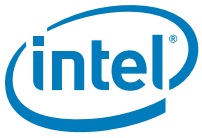
0xMMDD.

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.2.23.14 dateH - 0x000D

0xYYYY.

Bits	Field Name	Default NVM Value	Description
15:0	DateH		



6.2.23.15 sizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x00000800	

6.2.23.16 sizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.2.23.17 keySizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x40	

6.2.23.18 keySizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.2.23.19 modulusSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x40	

6.2.23.20 modulusSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.2.23.21 exponentSizeL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.2.23.22 exponentSizeH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.2.23.23 lad_srevL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.2.23.24 lad_srevH - 0x0017

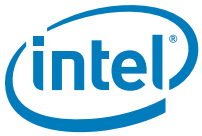
Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.2.23.25 reserved1L - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.23.26 reserved1H - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	



6.2.23.27 lad_fw_entry_offsetL - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x14c	

6.2.23.28 lad_fw_entry_offsetH - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.2.23.29 reserved2L - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.23.30 reserved2H - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.23.31 lad_image_unique_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.2.23.32 lad_image_unique_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.2.23.33 lad_module_idL - 0x0020

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0x4	Valid values are: 0x1 = EMP Image. 0x2 = Reserved. 0x3 = PCIe Analog. 0x4 = PHY Analog. 0x5 = Option ROM.

6.2.23.34 lad_module_idH - 0x0021

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.2.23.35 reserved[n] (0x0022 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.23.36 RSA Public Key [n] (0x0042 + 1*n, n=0...127)

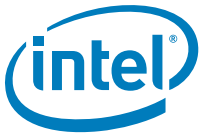
Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.2.23.37 RSA ExponentL - 0x00C2

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.2.23.38 RSA ExponentH - 0x00C3

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	



6.2.23.39 Encrypted SHA256 Hash[n] (0x00C4 + 1*n, n=0...127)

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.2.23.40 Device blank NVM device ID - 0x0144

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x154B	

6.2.23.41 Max module AreaL - 0x0145

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x1000	

6.2.23.42 Max module AreaH - 0x0146

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0000	

6.2.23.43 Current module AreaL - 0x0147

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x1000	

6.2.23.44 Current module AreaH - 0x0148

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	

6.2.23.45 Format version + CRC - 0x0149

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Format Version	0x2	
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: PHY Analog -> Section Length. End Section -> Word: PHY Analog -> Reserved Spare Word.

6.2.23.46 Code revision - 0x014A

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

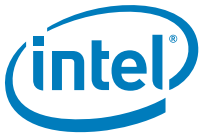
6.2.23.47 Reserved spare word - 0x014B

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

6.2.23.48 Section length - 0x12000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in words of the section covered by CRC.



6.2.23.49 CRC8 - 0x12003

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:0	Reserved	0x0	Reserved.

6.2.24 EMP global module section summary table

This section contains two sub-sections:

1. Vendor Specific Settings — Settings for an external PHY or Modules. This part has a proprietary format in order to enable advanced, vendor specific, PHY settings. These settings are loaded into the external devices via the MDIO/I^2C interface.
1. List of Qualified Modules — Parameter list of up to 16 modules. Per module list holds OUI, revision and version numbers.

Word Offset	Description	Reference
0x13000	Section Length	528
0x13001	Number of Qualified Modules	528
0x13002 + 12*n, n=0...15	Module OUI Bytes 0-1	528
0x13003 + 12*n, n=0...15	Module OUI Byte 2	528
0x13004 + 12*n, n=0...15	Vendor Part Number Bytes 0-1	529
0x13005 + 12*n, n=0...15	Vendor Part Number Bytes 2-3	529
0x13006 + 12*n, n=0...15	Vendor Part Number Bytes 4-5	529
0x13007 + 12*n, n=0...15	Vendor Part Number Bytes 6-7	530
0x13008 + 12*n, n=0...15	Vendor Part Number Bytes 8-9	530
0x13009 + 12*n, n=0...15	Vendor Part Number Bytes 10-11	530
0x1300A + 12*n, n=0...15	Vendor Part Number Bytes 12-13	531
0x1300B + 12*n, n=0...15	Vendor Part Number Bytes 14-15	531
0x1300C + 12*n, n=0...15	Module Revision Number Bytes 0-1	531
0x1300D + 12*n, n=0...15	Module Revision Number Bytes 2-3	532
0x130C2	Vendor Specific PHY Configurations	532
0x130C3	CRC8	532

6.2.24.1 Section length - 0x13000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in words of the section covered by CRC.

6.2.24.2 Number of qualified modules - 0x13001

Number of valid entries, 0 thru 15, in the qualified modules' list.

Bits	Field Name	Default NVM Value	Description
15:0	Number of Qualified Modules	0x10	

6.2.24.3 Module OUI bytes 0-1[n] (0x13002 + 12*n, n=0...15)

OUI of qualified external modules.

SFP+: Address 0xA0, Byte 3-5.

QSFP+: Address 133:131 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

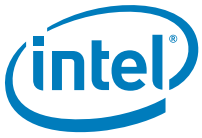
6.2.24.4 Module OUI byte 2[n] (0x13003 + 12*n, n=0...15)

OUI of qualified external modules.

SFP+: Address 0xA0, Byte 3-5.

QSFP+: Address 133:131 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	
7:0	Byte 0	0x0	



6.2.24.5 Vendor part number bytes 0-1[n] (0x13004 + 12*n, n=0...15)

Vendor part number of qualified external modules.

SFP+: Address 0xA0 Bytes 55:40.

QSFP+: Address 183:168 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.2.24.6 Vendor part number bytes 2-3[n] (0x13005 + 12*n, n=0...15)

Vendor part number of qualified external modules.

SFP+: Address 0xA0 Bytes 55:40.

QSFP+: Address 183:168 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.2.24.7 Vendor part number bytes 4-5[n] (0x13006 + 12*n, n=0...15)

Vendor part number of qualified external modules.

SFP+: Address 0xA0 Bytes 55:40.

QSFP+: Address 183:168 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.2.24.8 Vendor part number bytes 6-7[n] (0x13007 + 12*n, n=0...15)

Vendor part number of qualified external modules.

SFP+: Address 0xA0 Bytes 55:40.

QSFP+: Address 183:168 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.2.24.9 Vendor part number bytes 8-9[n] (0x13008 + 12*n, n=0...15)

Vendor Part Number of qualified external modules.

SFP+: Address 0xA0 Bytes 55:40.

QSFP+: Address 183:168 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

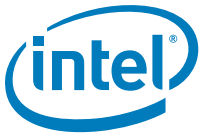
6.2.24.10 Vendor part number bytes 10-11[n] (0x13009 + 12*n, n=0...15)

Vendor part number of qualified external modules.

SFP+: Address 0xA0 Bytes 55:40.

QSFP+: Address 183:168 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	



6.2.24.11 Vendor part number bytes 12-13[n] (0x1300A + 12*n, n=0...15)

Vendor part number of qualified external modules.

SFP+: Address 0xA0 Bytes 55:40.

QSFP+: Address 183:168 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.2.24.12 Vendor part number bytes 14-15[n] (0x1300B + 12*n, n=0...15)

Vendor part number of qualified external modules.

SFP+: Address 0xA0 Bytes 55:40.

QSFP+: Address 183:168 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.2.24.13 Module revision number bytes 0-1[n] (0x1300C + 12*n, n=0...15)

Revision Number of qualified external modules.

SFP+: Address 0xA0 Bytes 59:56.

QSFP+: Address 185:184 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.2.24.14 Module revision number bytes 2-3[n] (0x1300D + 12*n, n=0...15)

Revision Number of qualified external modules.

SFP+: Address 0xA0 Bytes 59:56.

QSFP+: Address 185:184 page 0.

Bits	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.2.24.15 Vendor specific PHY configurations - 0x130C2

Raw data module length: variable.

The section contains vendor specific settings and configurations written in a proprietary code.

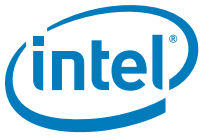
6.2.24.16 CRC8 - 0x130C3

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Reserved	0x0	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: EMP Global Module -> Section Length. End Section -> Word: EMP Global Module -> Vendor Specific PHY Configurations.

6.2.25 Manageability module header section summary table

This section contains parameters related to the manageability functionality such as connection type and others, It also points to sub-sections configuring the filters and the sideband interfaces.

Word Offset	Description	Reference
0x14000	Section Length	533
0x14001	Common Manageability Parameters	534
0x14002	Common Manageability Parameters 2	534
0x14003	Pass Through LAN 0 Configuration Pointer	535



Word Offset	Description	Reference
0x14004	Pass Through LAN 1 Configuration Pointer	535
0x14005	Pass Through LAN 2 Configuration Pointer	535
0x14006	Pass Through LAN 3 Configuration Pointer	536
0x14007	SideBand Configuration Pointer	536
0x14008	Flexible TCO Filter Configuration Pointer	536
0x14009	Traffic Types Parameters	537
0x1400A	OEM Structure Pointer	537
0x1400B	CRC8	538

6.2.25.1 Section length - 0x14000

The length of the section in words. Note that section length does not include a count for the section length word.

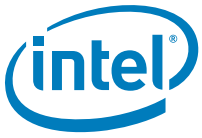
Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in words of the section covered by CRC.

6.2.25.2 Common manageability parameters - 0x14001

Bits	Field Name	Default NVM Value	Description
15	Reserved	0x0	Reserved.
10:8	Manageability Mode	0x2	Valid values are: 0x0 = None. 0x2 = Pass through (PT) mode.
5	Reserved	0x0	Reserved.
4	Force TCO Reset Disable	0x1	If cleared, enables the MC to do a global reset of the XL710 using the Force TCO SMBus or NC-SI commands Valid values are: 0x0 = Enable Force TCO reset. 0x1 = Disable Force TCO reset.
3	Reserved	0x0	0b = Disable. 1b = Enable.
2	OS2BMC Capable	0x1	0b = Disable. 1b = Enable. Valid values are: 0x0 = Disabled. 0x1 = Enabled.
1:0	Reserved	0x0	Reserved.

6.2.25.3 Common manageability parameters 2 - 0x14002

Bits	Field Name	Default NVM Value	Description
15:12	Reserved		Reserved.
11	Multi-Drop NC-SI	0x1	Valid values are: 0x0 = Point-to-point. 0x1 = Multi-drop.
10	Reserved		Reserved.
9	EMP_LINK_ON	0x0	Copy of the PRTPM_GC.EMP_LINK_ON bit for EMP use Valid values are: 0x0 = Disable. 0x1 = Enable.
8:4	Reserved	0x0	Reserved.
3	LAN3_FTCO_ISOL_DIS	0x1	Valid values are: 0x0 = Enabled. 0x1 = Disabled.
2	LAN2_FTCO_ISOL_DIS	0x1	Valid values are: 0x0 = Enabled. 0x1 = Disabled.
1	LAN1_FTCO_ISOL_DIS	0x1	Valid values are: 0x0 = Enabled. 0x1 = Disabled.



Bits	Field Name	Default NVM Value	Description
0	LAN0_FTCO_ISOL_DIS	0x1	Valid values are: 0x0 = Enabled. 0x1 = Disabled.

6.2.25.4 Pass through LAN 0 configuration pointer - 0x14003

Bits	Field Name	Default NVM Value	Description
15:0	Pass Through LAN 0 Configuration Pointer	0xFFFF	

The field pass through LAN 0 configuration pointer points to the pass trough control words structure 0 section. For more details about the pass through control words structure 0 inner structure, see page 538.

6.2.25.5 Pass Through LAN 1 Configuration Pointer - 0x14004

Bits	Field Name	Default NVM Value	Description
15:0	Pass Through LAN 1 Configuration Pointer	0xFFFF	

The field pass through LAN 1 configuration pointer points to the pass trough control words structure 0 section. For more details about the pass through control words structure 1 inner structure, see page 538.

6.2.25.6 Pass through LAN 2 configuration pointer - 0x14005

Bits	Field Name	Default NVM Value	Description
15:0	Pass Through LAN 2 Configuration Pointer	0xFFFF	

The field pass through LAN 2 configuration pointer points to the pass trough control words structure 0 section. For more details about the pass through control words structure 2 inner structure, see page 538.

6.2.25.7 Pass through LAN 3 configuration pointer - 0x14006

Bits	Field Name	Default NVM Value	Description
15:0	Pass Through LAN 3 Configuration Pointer	0xFFFF	

The field pass through LAN 3 configuration pointer points to the pass trough control words structure 0 section. For more details about the pass through control words structure 3 inner structure, see page 538.

6.2.25.8 Sideband configuration pointer - 0x14007

This module is 28 bytes long and must be mapped in the first valid 4 KB sector of the Flash.

Bits	Field Name	Default NVM Value	Description
15:0	SideBand Configuration Pointer	0xFFFF	

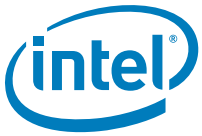
The field sideband configuration pointer points to the sideband configuration structure section. For more details about the sideband configuration structure inner structure, see page 549.

6.2.25.9 Flexible TCO filter configuration pointer - 0x14008

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.

Bits	Field Name	Default NVM Value	Description
15:0	Flexible TCO Filter Configuration Pointer	0xFFFF	

The field Flexible TCO filter configuration pointer points to the flexible TCO filter configuration structure section. For more details about the flexible TCO filter configuration structure inner structure, see page 548.



6.2.25.10 Traffic types parameters - 0x14009

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	Reserved.
13:12	Port 3 traffic types	0x1	Defines which type of traffic can flow to and from a primary MC 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. Valid values are: 0x0 = Reserved. 0x1 = Network to MC traffic only. 0x2 = OS2BMC traffic only. 0x3 = Both network to MC traffic and OS2BMC traffic.
11:10	Reserved	0x0	Reserved.
9:8	Port 2 traffic types	0x1	Defines which type of traffic can flow to and from a primary MC 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. Valid values are: 0x0 = Reserved. 0x1 = Network to MC traffic only. 0x2 = OS2BMC traffic only. 0x3 = Both network to MC traffic and OS2BMC traffic.
7:6	Reserved	0x0	Reserved.
5:4	Port 1 traffic types	0x1	Defines which type of traffic can flow to and from a primary MC connection on port 1. The traffic types defined by this field are enabled by the *Manageability Mode* field and the *OS2BMC Capable* bit in the Common Manageability Parameters 1 NVM word. Valid values are: 0x0 = Reserved. 0x1 = Network to MC traffic only. 0x2 = OS2BMC traffic only. 0x3 = Both network to MC traffic and OS2BMC traffic.
3:2	Reserved	0x0	Reserved.
1:0	Port 0 traffic types	0x1	Defines which type of traffic can flow to and from a primary MC connection on port 0. The traffic types defined by this field are enabled by the *Manageability Mode* field and the *OS2BMC Capable* bit in the Common Manageability Parameters 1 NVM word. Valid values are: 0x0 = Reserved. 0x1 = Network to MC traffic only. 0x2 = OS2BMC traffic only. 0x3 = Both network to MC traffic and OS2BMC traffic.

6.2.25.11 OEM structure pointer - 0x1400A

Bits	Field Name	Default NVM Value	Description
15:0	OEM Structure Pointer	0xFFFF	

The field OEM structure pointer points to OEM section. For more details about the OEM section inner structure, see page 555.

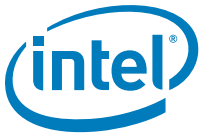
6.2.25.12 CRC8 - 0x1400B

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Reserved	0x0	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: Manageability Module Header -> Section Length. End Section -> Word: Manageability Module Header -> OEM Structure Pointer.

6.2.26 Pass through control words structure 0 section summary table

This section contains the initial setting of the pass through filters for port 0. This section is used only in SMBus legacy pass through mode.

Word Offset	Description	Reference
0x0000	Section Length	539
0x0001 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	540
0x0002 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	540
0x0009 + 3*n, n=0...3	LAN Ethernet MAC Address (LSB) MMAL	540
0x000A + 3*n, n=0...3	LAN Ethernet MAC Address (Mid) MMAL	540
0x000B + 3*n, n=0...3	LAN Ethernet MAC Address (MSB) MMAH	541
0x0015 + 2*n, n=0...15	LAN Flexible Filter Port	542
0x0016 + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	542
0x0035 + 1*n, n=0...7	LAN VLAN Filter	542
0x003D	LAN MANC Value LSB	542
0x003E	LAN MANC Value MSB	543
0x003F	Receive Enable 1 (LRXEN1)	543
0x0040	Receive Enable 2 (LRXEN2)	543
0x0041	LAN MNGONLY LSB	544



Word Offset	Description	Reference
0x0042	LAN MNGONLY MSB	544
0x0043 + 4*n, n=0...6	Manageability Decision Filters LSB	544
0x0044 + 4*n, n=0...6	Manageability Decision Filters MSB	544
0x0045 + 4*n, n=0...6	Manageability Extended Decision Filters LSB	544
0x0046 + 4*n, n=0...6	Manageability Extended Decision Filters MSB	544
0x005F + 2*n, n=0...3	Manageability Ethertype Filter - METF - LSB	545
0x0060 + 2*n, n=0...3	Manageability Ethertype Filter METF - MSB	545
0x0067	ARP Response IPv4 Address (LSB)	545
0x0068	ARP Response IPv4 Address (MSB)	545
0x0069 + 8*n, n=0...3	IPv6 Address Bytes 0-1	545
0x006A + 8*n, n=0...3	IPv6 Address Bytes 2-3	546
0x006B + 8*n, n=0...3	IPv6 Address Bytes 4-5	546
0x006C + 8*n, n=0...3	IPv6 Address Bytes 6-7	546
0x006D + 8*n, n=0...3	IPv6 Address Bytes 8-9	546
0x006E + 8*n, n=0...3	IPv6 Address Bytes 10-11	546
0x006F + 8*n, n=0...3	IPv6 Address Bytes 12-13	547
0x0070 + 8*n, n=0...3	IPv6 Address Bytes 14-15	547
0x0089	Manageability Special Modifiers - LSB	547
0x008A	Manageability Special Modifiers - MSB	547
0x008B	CRC8	548

6.2.26.1 Section Length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in words of the section covered by CRC.

6.2.26.2 LAN IPv4 Address (LSB) MIPAF0[n] (0x0001 + 2*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:8	IPv4 Address, Byte 1	0xFF	
7:0	IPv4 Address, Byte 0	0xFF	

6.2.26.3 LAN IPv4 Address (MSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:8	IPv4 Address, Byte 3	0xFF	
7:0	IPv4 Address, Byte 2	0xFF	

6.2.26.4 LAN Ethernet MAC Address (LSB) MMAL[n] (0x0009 + 3*n, n=0...3)

This word is loaded by firmware to the 16 LS bits of the MMAL[0-3] register.

It is used as the MAC address when dedicated MAC address mode is used in legacy SMBus.

This is a per-device, per-port value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:8	Ethernet MAC Address, Byte 1	0xFF	
7:0	Ethernet MAC Address, Byte 0	0xFF	

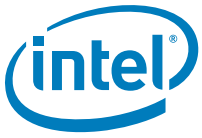
6.2.26.5 LAN Ethernet MAC address (Mid) MMAL[n] (0x000A + 3*n, n=0...3)

This word is loaded by firmware to the 16 MS bits of the MMAL[0-3] register.

It is used as the MAC address when dedicated MAC address mode is used in legacy SMBus.

This is a per-device, per-port value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:8	Ethernet MAC Address, Byte 3	0xFF	



Bits	Field Name	Default NVM Value	Description
7:0	Ethernet MAC Address, Byte 2	0xFF	

6.2.26.6 LAN Ethernet MAC address (MSB) MMAH[n] (0x000B + 3*n, n=0...3)

This word is loaded by firmware to the MMAH[0-3] register.

It is used as the MAC address when dedicated MAC address mode is used in legacy SMBus.

This is a per device per port value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:8	Ethernet MAC Address, Byte 5	0xFF	
7:0	Ethernet MAC Address, Byte 4	0xFF	

6.2.26.7 LAN flexible filter Port[n] (0x0015 + 2*n, n=0...15)

Bits	Field Name	Default NVM Value	Description
15:0	LAN UDP Flexible Filter Port0	0xFFFF	

6.2.26.8 LAN flexible filter Port - Modifiers[n] (0x0016 + 2*n, n=0...15)

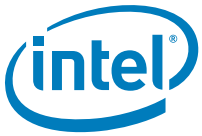
Bits	Field Name	Default NVM Value	Description
15:3	Reserved		Reserved.
2	Source Destination	0x0	Valid values are: 0x0 = Destination. 0x1 = Source.
1	Accept TCP	0x0	If set, filter match if packet is TCP. Valid values are: 0x0 = Filter. 0x1 = Accept.
0	Accept UDP	0x0	If set, filter match if packet is UDP. Valid values are: 0x0 = Filter. 0x1 = Accept.

6.2.26.9 LAN VLAN filter [n] (0x0035 + 1*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	
11:0	VLAN Filter 0 Value	0xFFF	

6.2.26.10 LAN MANC value LSB - 0x003D

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



6.2.26.11 LAN MANC value MSB - 0x003E

Bits	Field Name	Default NVM Value	Description
15:11	Reserved	0x0	Reserved.
10	NET_TYPE	0x0	
9	FIXED_NET_TYPE	0x0	
8	Reserved	0x0	
7	EN_XSUM_FILTER	0x0	Should be used to set the RBCR.EN_XSUM_FILTERn bit (n is the port number).
6:0	Reserved	0x0	Reserved.

6.2.26.12 Receive enable 1 (LRXEN1) - 0x003F

Bits	Field Name	Default NVM Value	Description
15:9	Receive Enable byte 12	0x00	MC SMBus slave address.
8	Reserved	0x0	Reserved.
7	Enable BMC Dedicated	0x0	When set, the MC receives traffic sent to the MAC address defined in MMAH/MMAL[3]. Firmware configures MDEF, MDEF_EXT (7) to a dedicated MAC configured into MMAL/H (3) to implement this mode.
6	Reserved	0x1	Reserved.
5:4	Notification method	0x0	Valid values are: 0x0 = SMBus alert. 0x1 = Asynchronous notify. 0x2 = Direct receive. 0x3 = Reserved.
3	Enable ARP Response	0x0	When set, firmware offloads ARP handling sent to the IP address defined in the ARP Response IPv4 Address NVM words. Firmware uses MDEF, MDEF_EXT (6) and MIPAF4[3] to implement this mode.
2	Enable Status Reporting	0x1	
1	Enable Receive All	0x0	
0	Enable Receive TCO	0x1	

6.2.26.13 Receive enable 2 (LRXEN2) - 0x0040

Bits	Field Name	Default NVM Value	Description
15:8	Receive Enable byte 14	0x00	Alert Value.
7:0	Receive Enable byte 13	0x00	Interface Data.

6.2.26.14 LAN MNGONLY LSB - 0x0041

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	
7:0	Exclusive to MNG	0x0	

6.2.26.15 LAN MNGONLY MSB - 0x0042

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

6.2.26.16 Manageability decision filters LSB[n] (0x0043 + 4*n, n=0...6)

Bits	Field Name	Default NVM Value	Description
15:0	MDEF0_L	0x0	

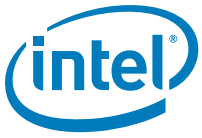
6.2.26.17 Manageability decision filters MSB[n] (0x0044 + 4*n, n=0...6)

Bits	Field Name	Default NVM Value	Description
15:0	MDEF0_M	0x0	

6.2.26.18 Manageability extended decision filters LSB[n] (0x0045 + 4*n, n=0...6)

Bits	Field Name	Default NVM Value	Description
15:0	MDEFEXT0_L	0x0	

6.2.26.19 Manageability extended decision filters MSB[n]



(0x0046 + 4*n, n=0...6)

Bits	Field Name	Default NVM Value	Description
15:0	MDEFEXT0_M	0x0	

6.2.26.20 Manageability Ethertype filter - METF - LSB[n] (0x005F + 2*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	METF0_L	0x0000	Loaded to 16 LS bits of METF[0] register.

6.2.26.21 Manageability Ethertype filter METF - MSB[n] (0x0060 + 2*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	METF0_M	0x0000	Loaded to 16 MS bits of the METF[0] register (reserved bits in the METF registers should be set in the NVM to the register default values).

6.2.26.22 ARP response IPv4 address (LSB) - 0x0067

Bits	Field Name	Default NVM Value	Description
15:8	Byte 1	0x00	Firmware use.
7:0	Byte 0	0x00	Firmware use.

6.2.26.23 ARP response IPv4 address (MSB) - 0x0068

Bits	Field Name	Default NVM Value	Description
15:8	Byte 3	0x00	Firmware use.
7:0	Byte 2	0x00	Firmware use.

6.2.26.24 IPv6 address bytes 0-1[n] (0x0069 + 8*n,

n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 0-1	0x0000	

6.2.26.25 IPv6 address bytes 2-3[n] (0x006A + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 2-3	0x0000	

6.2.26.26 IPv6 address bytes 4-5[n] (0x006B + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 2-3	0x0000	

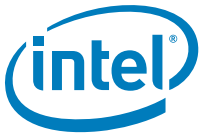
6.2.26.27 IPv6 address bytes 6-7[n] (0x006C + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 6-7	0x0000	

6.2.26.28 IPv6 address bytes 8-9[n] (0x006D + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 8-9	0x0000	

6.2.26.29 IPv6 address bytes 10-11[n] (0x006E + 8*n,



n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 10-11	0x0000	

6.2.26.30 IPv6 address bytes 12-13[n] (0x006F + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 12-13	0x0000	

6.2.26.31 IPv6 address bytes 14-15[n] (0x0070 + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 14-15	0x0000	

6.2.26.32 Manageability special modifiers - LSB - 0x0089

Bits	Field Name	Default NVM Value	Description
15:0	MSFM_L	0x000F	

6.2.26.33 Manageability special modifiers - MSB - 0x008A

Bits	Field Name	Default NVM Value	Description
15:0	MSFM_M	0x0000	

6.2.26.34 CRC8 - 0x008B

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Reserved	0x0	
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: Pass Through Control Words Structure 0 -> Section Length. End Section -> Word: Pass Through Control Words Structure 0 -> Manageability Special Modifiers - MSB.

6.2.27 Flexible TCO filter configuration structure section summary table

This section contains the setting of the Manageability flexible filters for all the ports. Its format is described in the Manageability section under Flexible TCO Filter Configuration.

Word Offset	Description	Reference
0x0000	Section Length	548
0x0001	Flexible Filter Data	548
0x0002	CRC8	549

6.2.27.1 Section length - 0x0000

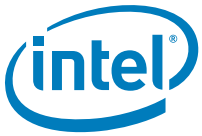
The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 bytes unit - 1. First Section -> Word: Flexible TCO Filter Configuration Structure -> Section Length. Last Section -> Word: Flexible TCO Filter Configuration Structure -> CRC8.

6.2.27.2 Flexible filter data - 0x0001

Raw data module length: variable.

Flexible filter data.



6.2.27.3 CRC8 - 0x0002

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Reserved	0x0	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: Flexible TCO Filter Configuration Structure -> Section Length. End Section -> Word: Flexible TCO Filter Configuration Structure -> Flexible Filter Data.

6.2.28 Sideband configuration structure section summary table

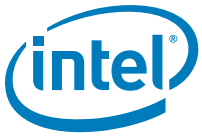
This section describes the setting of the different pass through interfaces (SMBus, NC-SI and MCTP).

Word Offset	Description	Reference
0x0000	Section Length	550
0x0001	SMBus Maximum Fragment Size	551
0x0002	SMBus Notification Timeout and Flags	551
0x0003	NC-SI Configuration 1	552
0x0004	NC-SI Configuration 2	552
0x0005	NC-SI Flow Control XOFF	552
0x0006	NC-SI Flow Control XON	553
0x0007	NC-SI HW Arbitration Configuration	553
0x0008	MCTP UUID - Time Low LSB	553
0x0009	MCTP UUID - Time Low MSB	553
0x000A	MCTP UUID - Time MID	553
0x000B	MCTP UUID - Time High and Version	553
0x000C	MCTP UUID - Clock Seq	553
0x000D	OEM IANA	553
0x000E	NC-SI over MCTP Message Types	553
0x000F	NC-SI over MCTP configuration	553
0x0010	MCTP Rate Limiter Config 1	554
0x0011	MCTP Rate Limiter Config 2	554
0x0012	NC-SI Channel to Port Mapping	554
0x0013	CRC8	555

6.2.28.1 Section Length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in words of the section covered by CRC.



6.2.28.2 SMBus maximum fragment size - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Fragment size	0x0020	SMBus Maximum Fragment Size (bytes). Supported range is between 32 and 240 bytes. **Note:** In MCTP mode, this value should be set to 0x45 (64 bytes payload + 5 bytes of MCTP header),

6.2.28.3 SMBus notification timeout and flags - 0x0002

Bits	Field Name	Default NVM Value	Description
15:8	SMBus Notification Timeout (ms)	0xFF	
7:6	SMBus Connection Speed	0x0	Valid values are: 0x0 = Standard SMBus connection. 0x1 = 400 Kb/s fast I^2C. 0x2 = 1 Mb/s fast + I^2C. 0x3 = Reserved.
5	SMBus Block Read command	0x0	Valid values are: 0x0 = Block read command is 0xC0. 0x1 = Block read command is 0xD0.
4	Reserved	0x0	Reserved.
3	Enable Fairness Arbitration	0x1	MCTP over SMBus feature. 0b = Disable fairness arbitration. 1b = Enable fairness arbitration. Valid values are: 0x0 = Disabled. 0x1 = Enabled.
2	Disable SMBus ARP Functionality	0x0	Valid values are: 0x0 = SMBus ARP - Enabled. 0x1 = SMBus ARP -Disabled.
1	SMBus ARP PEC	0x1	SMBus ARP PEC. Should be set in MCTP modes. Valid values are: 0x0 = Disable PEC - Disable SMBus ARP PEC. 0x1 = Enable PEC - Enable SMBus ARP PEC.
0	SMBus Transaction PEC	0x0	SMBus Transactions PEC. Should be set in MCTP modes. Valid values are: 0x0 = Disable PEC - If this bit is cleared, PEC is not added to master write or slave read transactions, a slave write transaction with PEC is dropped. 0x1 = Enable PEC - If this bit is set, PEC is added for master SMBus write transactions. A PEC is added to slave read transactions and can be received in slave write transaction.

6.2.28.4 NC-SI configuration 1 - 0x0003

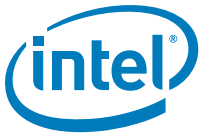
Bits	Field Name	Default NVM Value	Description
15	Reserved	0x0	Reserved.
14	Flow Control	0x0	Valid values are: 0x0 = NC-SI flow control disable. 0x1 = NC-SI flow control enable.
13:10	Reserved	0x0	Reserved.
9	NC-SI HW arbitration enable	0x0	Valid values are: 0x0 = Not supported. 0x1 = supported.
7:5	Package ID	0x0	Meaningful only when bit 15 of NC-SI Configuration 2 word (offset 0x07) is cleared.
4:1	Reserved	0x0	Reserved, must be zero.

6.2.28.5 NC-SI configuration 2 - 0x0004

Bits	Field Name	Default NVM Value	Description
15	Read NC-SI Package ID from SDP	0x0	Read NC-SI Package ID From. 0b = NVM NC-SI Configuration 1 word bits 7:5 (offset 0x06). 1b = The SDP defined in the *PackageID* SDP field defines the package ID. In this case, the package ID can be 0 or 2 according to (0, SDP value, 0). Valid values are: 0x0 = PackageID from NVM. 0x1 = PackageID from SDP.
14:10	PackageID SDP	0x0	Defines the SDP from which the NC-SI package ID is taken
9:8	Reserved	0x000	Reserved.
3:0	Max XOFF Renewal	0x0	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 XL710. 0x2 = Up to 3 consecutive XOFFs frames can be sent by the XL710. . . . 0xF = Up to 16 consecutive XOFFs frames can be sent by the XL710.

6.2.28.6 NC-SI flow control XOFF - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	XOFF Threshold	0x0	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. **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 6 KB value of field should be 0xAB8 (2,744 bytes).



6.2.28.7 NC-SI flow control XON - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	XON Threshold	0x0	Tx buffer water mark for sending a XON NC-SI flow control packet in bytes. The XON threshold value refers to the available space in the Tx buffer. **Notes:** Field relevant for NC-SI operation mode only. To support a maximum packet size of 1.5 KB, the value programmed should be a positive value that equals: XOFF threshold + 1536 bytes. Assuming a Tx buffer size is 6 KB and the XOFF threshold is 2,744 bytes, the value of the field should be 0x1348 (4,936 bytes).

6.2.28.8 NC-SI HW arbitration configuration - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	TOKEN Timeout	0xA000	NC-SI HW-Arbitration Token Timeout (in NC-SI REF_CLK cycles - 20 ns). Setting this value to 0x0 disables the timeout mechanism.

6.2.28.9 OEM IANA - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	OEM IANA	0x0	If not 0x0 and not 0x157, the XL710 accepts additional OEM commands with this IANA number. These commands are accepted only if the OEM NVM structure pointer is valid. The set of commands accepted depends on the IANA value in this field. The regular Intel OEM commands are accepted only with IANA 0x157.

6.2.28.10 NC-SI over MCTP message types - 0x000E

Bits	Field Name	Default NVM Value	Description
15:8	NC-SI Control Message type	0x2	Defines the MCTP message type used to identify NC-SI control packets.
7:0	NC-SI Pass Through Message type	0x3	Defines the MCTP message type used to identify NC-SI pass through packets.

6.2.28.11 NC-SI over MCTP configuration - 0x000F

Bits	Field Name	Default NVM Value	Description
15:7	Reserved	0x0	Reserved.
6	Simplified MCTP	0x0	If set, only SOM and EOM bits are used for the reassembly process. Relevant only in SMBus mode.

Bits	Field Name	Default NVM Value	Description
5	Disable ACLs	0x0	If set, the ACLs on the PCIe VDMs are disabled.
4:1	Reserved	0x0	Reserved.

6.2.28.12 MCTP rate limiter config 1 - 0x0010

MCTP rate limiter configuration for first channel.

Bits	Field Name	Default NVM Value	Description
15:0	MCTP Rate	0x3E80	Defines the number of cycles between accesses of the MCTP send client to the memory arbiter. Current value assumes a clock of 125 MHz and a bus width of 128 bits. This value provides a bit rate of 1 Mb/s.

6.2.28.13 MCTP rate limiter config 2 - 0x0011

MCTP rate limiter configuration for first channel.

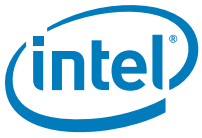
Bits	Field Name	Default NVM Value	Description
15	Decision Point	0x0	Defines if, when credits are available, a full MCTP message is sent or a single VDM is sent. Valid values are: 0x0 = Per VDM. 0x1 = Per packet.
14:0	MCTP max credits	0x5	Defines the maximum number of 16 bytes credit that can be accumulated. These credits include the VDM header line (one line for each 64 byte VDM).

6.2.28.14 NC-SI channel to port mapping - 0x0012

Defines the mapping of NC-SI channels to physical ports.

Note: The list of channel IDs must start at zero and be consecutive.

Bits	Field Name	Default NVM Value	Description
15	Table Valid	0x0	Valid values are: 0x0 = Table Invalid. Use default algorithm described in the channel ID mapping section. 0x1 = Table Valid. Use the mapping defined in this word.
14:13	Port #3 channel ID	0x0	
12	Port #3 Channel Enable	0x1	Valid values are: 0x0 = Disabled. 0x1 = Enabled.
11	Reserved	0x0	Reserved.



Bits	Field Name	Default NVM Value	Description
10:9	Port #2 channel ID	0x0	
8	Port #2 Channel Enable	0x1	Valid values are: 0x0 = Disabled. 0x1 = Enabled.
7	Reserved	0x0	Reserved.
6:5	Port #1 channel ID	0x0	
4	Port #1 Channel Enable	0x1	Valid values are: 0x0 = Disabled. 0x1 = Enabled.
3	Reserved	0x0	Reserved.
2:1	Port #0 channel ID	0x0	
0	Port #0 Channel Enable	0x1	Valid values are: 0x0 = Disabled. 0x1 = Enabled.

6.2.28.15 CRC8 - 0x0013

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used 0x1 = CRC Used
14:8	Reserved	0x0	
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: Sideband Configuration Structure -> Section Length End Section -> Word: Sideband Configuration Structure -> NC-SI Channel to Port Mapping

6.2.29 OEM section summary table

This section is the header of the OEM specific data identifying the OEM for which this data is defined.

Word Offset	Description	Reference
0x0000	Section Length	556
0x0001	OEM header	556
0x0002	Inventory Parameters Structure Pointer	556
0x0003	Extended Capabilities Parameters Structure Pointer	556
0x0004	Partition Information Structure Pointer	556
0x0005	CEM Flags	556
0x0006	CRC8	556

6.2.29.1 Section length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1. First Section -> Word: OEM Section -> Section Length. Last Section -> Word: OEM Section -> CRC8.

6.2.29.2 OEM header - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	OEM Identifier	0x02A2	Identify the OEM for which this section is defined. Should be equal to the OEM IANA value.

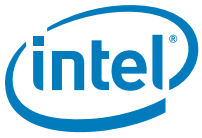
6.2.29.3 CRC8 - 0x0006

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Reserved	0x0	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: OEM Section -> Section Length. End Section -> Word: OEM Section -> Partition Information structure pointer.

6.2.30 EMP settings module header section summary table

This section contains the modes of operation of the EMP.

Word Offset	Description	Reference
0x15000	Section Length	557
0x15001	Common Firmware Parameters	558
0x15002	Reserved	558
0x15003	Features Enable	558
0x15005	PF Allocations Pointer	559
0x15006	EEE Variables	559



Word Offset	Description	Reference
0x15007	PHY Capability LAN 0 Pointer	559
0x15008	PHY Capability LAN 1 Pointer	560
0x15009	PHY Capability LAN 2 Pointer	560
0x1500A	PHY Capability LAN 3 Pointer	560
0x1500B	EXT to INT PHY Mapping LAN 0 Pointer	560
0x1500C	EXT to INT PHY Mapping LAN 1 Pointer	561
0x1500D	EXT to INT PHY Mapping LAN 2 Pointer	561
0x1500E	EXT to INT PHY Mapping LAN 3 Pointer	561
0x1500F	LLDP Configuration Pointer	561
0x15010	LTR MAX SNOOP	562
0x15011	LTR MAX NO SNOOP	562
0x15012	LTR DELTA	562
0x15013	LTR GRADE VALUES	562
0x15014	LLDP TLVs Pointer	562
0x15015	CRC8	562

6.2.30.1 Section length - 0x15000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in words of the section covered by CRC.

6.2.30.2 Common firmware parameters - 0x15001

Bits	Field Name	Default NVM Value	Description
15:10	Reserved	0x5	Reserved.
9	PF reset on queue overflow	0x0	Valid values are: 0x0 = Disabled. 0x1 = Enabled.
8:0	Reserved	0x3	Reserved.

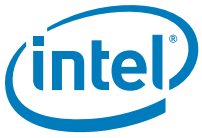
6.2.30.3 Reserved - 0x15002

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

6.2.30.4 Features enable - 0x15003

Describes support for various features.

Bits	Field Name	Default NVM Value	Description
15:9	Reserved	0x0	Reserved.
8	Proxy Enabled for Port 3	0x1	If cleared, the proxy functionality embedded in EMP is disabled on Port 3. It could be re-enabled on this port only by resetting this bit to 1b and performing an EMPR cycle. Valid values are: 0x0 = Disabled. 0x1 = Enabled.
7	Proxy Enabled for Port 2	0x1	If cleared, the proxy functionality embedded in EMP is disabled on Port 2. It could be re-enabled on this port only by resetting this bit to 1b and performing an EMPR cycle. Valid values are: 0x0 = Disabled. 0x1 = Enabled.
6	Proxy Enabled for Port 1	0x1	If cleared, the proxy functionality embedded in EMP is disabled on Port 1. It could be re-enabled on this port only by resetting this bit to 1b and performing an EMPR cycle. Valid values are: 0x0 = Disabled. 0x1 = Enabled.
5	Proxy Enabled for Port 0	0x1	If cleared, the proxy functionality embedded in EMP is disabled on Port 0. It could be re-enabled on this port only by resetting this bit to 1b and performing an EMPR cycle. Valid values are: 0x0 = Disabled. 0x1 = Enabled.
4:1	Reserved	0x0	Reserved.



Bits	Field Name	Default NVM Value	Description
0	EVB Protocols Enabled	0x1	If set, filtering according to IEEE 802.1QBg & 802.1BR specs is enabled. Valid values are: 0x0 = Disabled. 0x1 = Enabled.

6.2.30.5 PF allocations pointer - 0x15005

Bits	Field Name	Default NVM Value	Description
15:0	PF Allocations Pointer	0x0	

The field PF allocations pointer points to the PF allocations section. For more details about the PF allocations inner structure, see page 562.

6.2.30.6 EEE variables - 0x15006

Bits	Field Name	Default NVM Value	Description
15:8	receiveTW	0xe	Value determines the time (expressed in microseconds) that the receiving link partner is requesting the transmitting link partner to wait before starting the transmission data following the LPI. This value is platform dependent and depends on the OEM platform.
7:0	transmitTW	0xe	Value determines the time (expressed in microseconds) that the transmitting link partner 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.

6.2.30.7 PHY capability LAN 0 pointer - 0x15007

Bits	Field Name	Default NVM Value	Description
15:0	PHY	0xFFFF	

The field PHY points to the PHY capability data structure 0 Section. For more details about the PHY capability data structure 0 inner structure, see page 603.

6.2.30.8 PHY capability LAN 1 pointer - 0x15008

Bits	Field Name	Default NVM Value	Description
15:0	PHY	0xFFFF	

The field PHY points to the PHY capability data structure 1 Section. For more details about the PHY capability data structure 1 inner structure, see page 603.

6.2.30.9 PHY capability LAN 2 pointer - 0x15009

Bits	Field Name	Default NVM Value	Description
15:0	PHY	0xFFFF	

The field PHY points to the PHY capability data structure 2 Section. For more details about the PHY capability data structure 2 inner structure, see page 603.

6.2.30.10 PHY capability LAN 3 pointer - 0x1500A

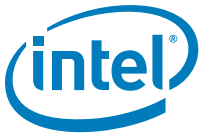
Bits	Field Name	Default NVM Value	Description
15:0	PHY	0xFFFF	

The field PHY points to the PHY capability data structure 3 Section. For more details about the PHY capability data structure 3 inner structure, see page 603.

6.2.30.11 EXT to INT PHY mapping LAN 0 pointer - 0x1500B

Bits	Field Name	Default NVM Value	Description
15:0	LAN 0 Pointer	0xFFFF	

The field LAN 0 pointer points to external-to-internal PHY mapping 0 section. For more details about the external-to-internal PHY mapping 0 inner structure, see page 609.



6.2.30.12 EXT to INT PHY mapping LAN 1 pointer - 0x1500C

Bits	Field Name	Default NVM Value	Description
15:0	LAN 1 Pointer	0xFFFF	

The field LAN 0 pointer points to external-to-internal PHY mapping 1 section. For more details about the external-to-internal PHY mapping 1 inner structure, see page 609.

6.2.30.13 EXT to INT PHY mapping LAN 2 pointer - 0x1500D

Bits	Field Name	Default NVM Value	Description
15:0	LAN 2 Pointer	0xFFFF	

The field LAN 0 pointer points to external-to-internal PHY mapping 2 section. For more details about the external-to-internal PHY mapping 2 inner structure, see page 609.

6.2.30.14 EXT to INT PHY mapping LAN 3 pointer - 0x1500E

Bits	Field Name	Default NVM Value	Description
15:0	LAN 3 Pointer	0xFFFF	

The field LAN 0 pointer points to external-to-internal PHY mapping 3 section. For more details about the external-to-internal PHY mapping 3 inner structure, see page 609.

6.2.30.15 LLDP configuration pointer - 0x1500F

Bits	Field Name	Default NVM Value	Description
15:0	LLDP Configuration Pointer	0xFFFF	

The field LLDP configuration pointer points to the LLDP configuration section. For more details about the LLDP configuration inner structure, see page 612.

6.2.30.16 LLDP TLVs pointer - 0x15014

Pointer to a set of fixed-structure TLVs used by the EMP.

Bits	Field Name	Default NVM Value	Description
15:0	LLDP TLVs Pointer	0xFFFF	Pointer to a set of fixed-structure TLVs used by the EMP.

The field LLDP TLVs pointer points to the LLDP OEM TLVs Section. For more details about the LLDP OEM TLVs inner structure, see page 614.

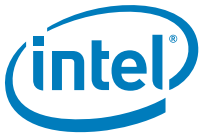
6.2.30.17 CRC8 - 0x15015

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Reserved	0x0	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: EMP Settings Module Header -> Section Length. End Section -> Word: EMP Settings Module Header -> LLDP TLVs pointer.

6.2.31 PF allocations section summary table

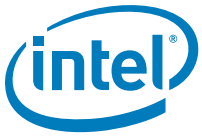
This section contains the allocation of resources per PF and the PF MAC address.

Word Offset	Description	Reference
0x0000	Section Length	569
0x0001	PF Allocations - VSI and VEB	569
0x0002	PF Allocations - MACs	569
0x0003	PF Allocations - Outer Tags	569
0x0004	PF Allocations - Inner Tags	570
0x0005	PF Allocations - Inner MACs	570
0x0006	PF Allocations - IPs	570
0x0007	PF Allocations - stats	571
0x0008	PF Allocations - mirror rules	571
0x0009	PF Allocations - Queue Sets	571
0x000A	PF Allocations - Teredo	571
0x000B	PF Flags	572
0x000C	PFPM_SAL0	572



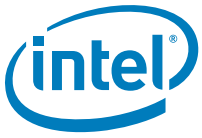
Word Offset	Description	Reference
0x000D	PFPM_SAL1	572
0x000E	PFPM_SAH0	572
0x000F	PFPM_SAH1	573
0x0010	PF Allocations - VSI and VEB	573
0x0011	PF Allocations - MACs	573
0x0012	PF Allocations - Outer Tags	573
0x0013	PF Allocations - Inner Tags	573
0x0014	PF Allocations - Inner MACs	573
0x0015	PF Allocations - IPs	574
0x0016	PF Allocations - Stats	574
0x0017	PF Allocations - Mirror Rules	574
0x0018	PF Allocations - Queue Sets	574
0x0019	PF Allocations - Teredo	574
0x001A	PF Flags	574
0x001B	PFPM_SAL0	574
0x001C	PFPM_SAL1	574
0x001D	PFPM_SAH0	575
0x001E	PFPM_SAH1	575
0x001F	PF Allocations - VSI and VEB	575
0x0020	PF Allocations - MACs	575
0x0021	PF Allocations - Outer Tags	575
0x0022	PF Allocations - Inner Tags	575
0x0023	PF Allocations - Inner MACs	575
0x0024	PF Allocations - IPs	576
0x0025	PF Allocations - Stats	576
0x0026	PF Allocations - Mirror Rules	576
0x0027	PF Allocations - Queue Sets	576
0x0028	PF Allocations - Teredo	576
0x0029	PF Flags	576
0x002A	PFPM_SAL0	576
0x002B	PFPM_SAL1	576
0x002C	PFPM_SAH0	577
0x002D	PFPM_SAH1	577
0x002E	PF Allocations - VSI and VEB	577
0x002F	PF Allocations - MACs	577
0x0030	PF Allocations - Outer Tags	577
0x0031	PF Allocations - Inner tags	577
0x0032	PF Allocations - Inner MACs	577
0x0033	PF Allocations - IPs	578

Word Offset	Description	Reference
0x0034	PF Allocations - Stats	578
0x0035	PF Allocations - Mirror Rules	578
0x0036	PF Allocations - Queue Sets	578
0x0037	PF Allocations - Teredo	578
0x0038	PF Flags	578
0x0039	PFPM_SAL0	578
0x003A	PFPM_SAL1	578
0x003B	PFPM_SAH0	579
0x003C	PFPM_SAH1	579
0x003D	PF Allocations - VSI and VEB	579
0x003E	PF Allocations - MACs	579
0x003F	PF Allocations - Outer Tags	579
0x0040	PF Allocations - Inner Tags	579
0x0041	PF Allocations - Inner MACs	579
0x0042	PF Allocations - IPs	580
0x0043	PF Allocations - Stats	580
0x0044	PF Allocations - Mirror Rules	580
0x0045	PF Allocations - Queue Sets	580
0x0046	PF Allocations - Teredo	580
0x0047	PF Flags	580
0x0048	PFPM_SAL0	580
0x0049	PFPM_SAL1	580
0x004A	PFPM_SAH0	581
0x004B	PFPM_SAH1	581
0x004C	PF Allocations - VSI and VEB	581
0x004D	PF Allocations - MACs	581
0x004E	PF Allocations - Outer Tags	581
0x004F	PF Allocations - Inner Tags	581
0x0050	PF Allocations - Inner MACs	581
0x0051	PF Allocations - IPs	582
0x0052	PF Allocations - Stats	582
0x0053	PF Allocations - Mirror Rules	582
0x0054	PF Allocations - Queue Sets	582
0x0055	PF Allocations - Teredo	582
0x0056	PF Flags	582
0x0057	PFPM_SAL0	582
0x0058	PFPM_SAL1	582
0x0059	PFPM_SAH0	583
0x005A	PFPM_SAH1	583



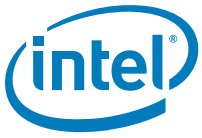
Word Offset	Description	Reference
0x005B	PF Allocations - VSI and VEB	583
0x005C	PF Allocations - MACs	583
0x005D	PF Allocations - Outer Tags	583
0x005E	PF Allocations - Inner Tags	583
0x005F	PF Allocations - Inner MACs	583
0x0060	PF Allocations - IPs	584
0x0061	PF Allocations - Stats	584
0x0062	PF Allocations - Mirror Rules	584
0x0063	PF Allocations - Queue Sets	584
0x0064	PF Allocations - Teredo	584
0x0065	PF Flags	584
0x0066	PFPM_SAL0	584
0x0067	PFPM_SAL1	584
0x0068	PFPM_SAH0	585
0x0069	PFPM_SAH1	585
0x006A	PF Allocations - VSI and VEB	585
0x006B	PF Allocations - MACs	585
0x006C	PF Allocations - Outer Tags	585
0x006D	PF Allocations - Inner Tags	585
0x006E	PF Allocations - Inner MACs	585
0x006F	PF Allocations - IPs	586
0x0070	PF Allocations - Stats	586
0x0071	PF Allocations - Mirror Rules	586
0x0072	PF Allocations - Queue Sets	586
0x0073	PF Allocations - Teredo	586
0x0074	PF Flags	586
0x0075	PFPM_SAL0	586
0x0076	PFPM_SAL1	586
0x0077	PFPM_SAH0	587
0x0078	PFPM_SAH1	587
0x0079	PF Allocations - VSI and VEB	587
0x007A	PF Allocations - MACs	587
0x007B	PF Allocations - Outer Tags	587
0x007C	PF Allocations - Inner Tags	587
0x007D	PF Allocations - Inner MACs	587
0x007E	PF Allocations - IPs	588
0x007F	PF Allocations - Stats	588
0x0080	PF Allocations - Mirror Rules	588
0x0081	PF Allocations - Queue Sets	588

Word Offset	Description	Reference
0x0082	PF Allocations - Teredo	588
0x0083	PF Flags	588
0x0084	PFPM_SAL0	588
0x0085	PFPM_SAL1	588
0x0086	PFPM_SAH0	589
0x0087	PFPM_SAH1	589
0x0088	PF Allocations - VSI and VEB	589
0x0089	PF Allocations - MACs	589
0x008A	PF Allocations - Outer Tags	589
0x008B	PF Allocations - Inner Tags	589
0x008C	PF Allocations - Inner MACs	589
0x008D	PF Allocations - IPs	590
0x008E	PF Allocations - Stats	590
0x008F	PF Allocations - Mirror Rules	590
0x0090	PF Allocations - Queue Sets	590
0x0091	PF Allocations - Teredo	590
0x0092	PF Flags	590
0x0093	PFPM_SAL0	590
0x0094	PFPM_SAL1	590
0x0095	PFPM_SAH0	591
0x0096	PFPM_SAH1	591
0x0097	PF Allocations - VSI and VEB	591
0x0098	PF Allocations - MACs	591
0x0099	PF Allocations - Outer Tags	591
0x009A	PF Allocations - Inner Tags	591
0x009B	PF Allocations - Inner MACs	591
0x009C	PF Allocations - IPs	592
0x009D	PF Allocations - Stats	592
0x009E	PF Allocations - Mirror Rules	592
0x009F	PF Allocations - Queue Sets	592
0x00A0	PF Allocations - Teredo	592
0x00A1	PF Flags	592
0x00A2	PFPM_SAL0	592
0x00A3	PFPM_SAL1	592
0x00A4	PFPM_SAH0	593
0x00A5	PFPM_SAH1	593
0x00A6	PF Allocations - VSI and VEB	593
0x00A7	PF Allocations - MACs	593
0x00A8	PF Allocations - Outer Tags	593



Word Offset	Description	Reference
0x00A9	PF Allocations - Inner Tags	593
0x00AA	PF Allocations - inner MACs	593
0x00AB	PF Allocations - IPs	594
0x00AC	PF Allocations - Stats	594
0x00AD	PF Allocations - Mirror Rules	594
0x00AE	PF Allocations - Queue Sets	594
0x00AF	PF Allocations - Teredo	594
0x00B0	PF Flags	594
0x00B1	PFPM_SAL0	594
0x00B2	PFPM_SAL1	594
0x00B3	PFPM_SAH0	595
0x00B4	PFPM_SAH1	595
0x00B5	PF Allocations - VSI and VEB	595
0x00B6	PF Allocations - MACs	595
0x00B7	PF Allocations - Outer Tags	595
0x00B8	PF Allocations - Inner Tags	595
0x00B9	PF Allocations - Inner MACs	595
0x00BA	PF Allocations - IPs	596
0x00BB	PF allocations - Stats	596
0x00BC	PF allocations - Mirror Rules	596
0x00BD	PF allocations - Queue Sets	596
0x00BE	PF allocations - Teredo	596
0x00BF	PF Flags	596
0x00C0	PFPM_SAL0	596
0x00C1	PFPM_SAL1	596
0x00C2	PFPM_SAH0	597
0x00C3	PFPM_SAH1	597
0x00C4	PF allocations - VSI and VEB	597
0x00C5	PF allocations - MACs	597
0x00C6	PF allocations - Outer Tags	597
0x00C7	PF allocations - Inner Tags	597
0x00C8	PF allocations - Inner MACs	597
0x00C9	PF allocations - IPs	598
0x00CA	PF allocations - Stats	598
0x00CB	PF allocations - Mirror Rules	598
0x00CC	PF allocations - Queue Sets	598
0x00CD	PF allocations - Teredo	598
0x00CE	PF Flags	598
0x00CF	PFPM_SAL0	598

Word Offset	Description	Reference
0x00D0	PFPM_SAL1	598
0x00D1	PFPM_SAH0	599
0x00D2	PFPM_SAH1	599
0x00D3	PF allocations - VSI and VEB	599
0x00D4	PF allocations - MACs	599
0x00D5	PF allocations - Outer Tags	599
0x00D6	PF allocations - Inner Tags	599
0x00D7	PF allocations - Inner MACs	599
0x00D8	PF allocations - IPs	600
0x00D9	PF allocations - Stats	600
0x00DA	PF allocations - Mirror Rules	600
0x00DB	PF allocations - Queue Sets	600
0x00DC	PF allocations - Teredo	600
0x00DD	PF Flags	600
0x00DE	PFPM_SAL0	600
0x00DF	PFPM_SAL1	600
0x00E0	PFPM_SAH0	601
0x00E1	PFPM_SAH1	601
0x00E2	PF Allocations - VSI and VEB	601
0x00E3	PF Allocations - MACs	601
0x00E4	PF Allocations - Outer Tags	601
0x00E5	PF Allocations - Inner Tags	601
0x00E6	PF Allocations - Inner MACs	601
0x00E7	PF Allocations - IPs	602
0x00E8	PF Allocations - Stats	602
0x00E9	PF Allocations - Mirror Rules	602
0x00EA	PF Allocations - Queue Sets	602
0x00EB	PF Allocations - Teredo	602
0x00EC	PF Flags	602
0x00ED	PFPM_SAL0	602
0x00EE	PFPM_SAL1	602
0x00EF	PFPM_SAH0	603
0x00F0	PFPM_SAH1	603
0x00F1	CRC8	603



6.2.31.1 Section length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 bytes unit - 1. First Section -> Word: PF Allocations -> Section Length. Last Section -> Word: PF Allocations -> CRC8.

6.2.31.2 PF allocations - VSI and VEB - 0x0001

Defines the allocation of VSIs and VEBs to each PF.

Bits	Field Name	Default NVM Value	Description
15:10	VEBs	0x0	Defines the number of VEBs guaranteed to this PF. A value of zero means no VEBs are guaranteed to this PF.
9:0	VSIs	0x0	Defines the number of VSIs guaranteed to this PF. A value of zero means no VSIs are guaranteed to this PF.

6.2.31.3 PF allocations - MACs - 0x0002

Defines the allocation of MACs to each PF.

Bits	Field Name	Default NVM Value	Description
15:11	Reserved		Reserved.
10:0	MACs	0x0	Defines the number of MACs guaranteed to this PF. A value of zero means no MACs are guaranteed to this PF.

6.2.31.4 PF allocations - outer tags - 0x0003

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

Bits	Field Name	Default NVM Value	Description
15:10	Reserved		Reserved.

Bits	Field Name	Default NVM Value	Description
9:0	S-tags/Outer VLANs	0x0	Defines the number of outer tags guaranteed to this PF. A value of zero means no tags are guaranteed to this PF. The tags allocated by this field depends on the image used: 1. For EVB image - indicates the number of S-tags allocated to the PF. 2. For Cloud images - N/A. 3. For CPM image - indicates the number of outer VLANs allocated to the PF.

6.2.31.5 PF allocations - inner tags - 0x0004

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

Bits	Field Name	Default NVM Value	Description
15:10	Reserved		Reserved.
9:0	Inner VLANs	0x0	Defines the number of inner VLAN tags guaranteed to this PF. A value of zero means no tags are guaranteed to this PF. The tags allocated by this field depends on the image used: 1. For EVB image - N/A. 2. For Cloud images - indicates the number of inner VLANs allocated to the PF. 3. For OVEB image - indicates the number of regular VLANs allocated to the PF.

6.2.31.6 PF allocations - inner MACs - 0x0005

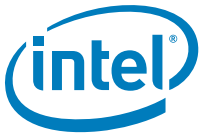
Defines the allocation of inner MACs to each PF - relevant only for cloud images.

Bits	Field Name	Default NVM Value	Description
15:11	Reserved	0x0	Reserved.
10:0	Inner MACs	0x0	Defines the number of inner MACs guaranteed to this PF. A value of zero means no inner MACs are guaranteed to this PF.

6.2.31.7 PF allocations - IPs - 0x0006

Defines the allocation of IP to each PF - relevant only for cloud images.

Bits	Field Name	Default NVM Value	Description
15:11	Reserved	0x0	Reserved.
10:0	IPs	0x0	Defines the number of IP addresses guaranteed to this PF. A value of zero means no IP addresses are guaranteed to this PF.



6.2.31.8 PF allocations - stats - 0x0007

Defines the allocation of statistics pools to each PF.

Bits	Field Name	Default NVM Value	Description
15:14	Reserved		Reserved.
13:7	smonPrioStats pools	0x0	Defines the number of smonPrioStats statistics pools guaranteed to this PF. A value of zero means no pools are guaranteed to this PF.
6:0	smonVlanStats pools	0x0	Defines the number of smonVlanStats statistics pools guaranteed to this PF. A value of zero means no pools are guaranteed to this PF.

6.2.31.9 PF allocations - mirror rules - 0x0008

Defines the allocation of mirror rules to each PF.

Bits	Field Name	Default NVM Value	Description
15:7	Reserved		Reserved
6:0	Mirror rules	0x0	Defines the number of mirror rules pools guaranteed to this PF. A value of zero means no rules are guaranteed to this PF.

6.2.31.10 PF allocations - queue sets - 0x0009

Defines the allocation of Tx queue sets to each PF.

Bits	Field Name	Default NVM Value	Description
15:12	Reserved		Reserved.
11:0	Queue Sets	0x0	Defines the number of queue sets guaranteed to this PF. A value of zero means no queue sets are guaranteed to this PF.

6.2.31.11 PF allocations - teredo - 0x000A

Defines the allocation of Teredo ports to each PF.

Bits	Field Name	Default NVM Value	Description
15:8	Reserved		Reserved.
7:0	Teredo UDP ports	0x0	Teredo/tunneling UDP ports allocated to this function.

6.2.31.12 PF flags - 0x000B

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	Load PF MAC Address	0x1	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 S-tag are forwarded to the PF.

6.2.31.13 PFPM_SAL0 - 0x000C

This word is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1,0].

One MAC address per enabled PF and up to 4 PFs per port.

This is a per-device, per-enabled PF value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:0	PFPM_SAL0	0x0000	See respective bits in the PRTPM_SAL register.

6.2.31.14 PFPM_SAL1 - 0x000D

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

One MAC address per enabled PF and up to 4 PFs per port.

This is a per-device, per-enabled PF value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:0	PFPM_SAL1	0x0000	See respective bits in the PRTPM_SAL register.

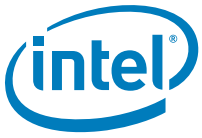
6.2.31.15 PFPM_SAH0 - 0x000E

This register is loaded by FW to the appropriate PRTPM_SAH register, bytes [1,0].

One MAC address per enabled PF and up to 4 PFs per port.

This is a per-device, per-enabled PF value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:0	PFPM_SAH0	0x0000	See respective bits in the PRTPM_SAH register.



6.2.31.16 PFPM_SAH1 - 0x000F

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

One MAC address per enabled PF,and up to 4 PFs per port.

This is a per-device, per-enabled PF value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:0	PFPM_SAH1	0x0000	See respective bits in the PRTPM_SAH register.

6.2.31.17 PF allocations - VSI and VEB - 0x0010

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.18 PF allocations - MACs - 0x0011

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.19 PF allocations - outer tags - 0x0012

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.20 PF allocations - inner tags - 0x0013

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.21 PF allocations - inner MACs - 0x0014

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.22 PF allocations - IPs - 0x0015

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570

6.2.31.23 PF allocations - stats - 0x0016

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.24 PF allocations - mirror rules - 0x0017

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.25 PF allocations - queue sets - 0x0018

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571

6.2.31.26 PF allocations - teredo - 0x0019

Defines the allocation of Teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.27 PF flags - 0x001A

For more details about the inner structure, see page 572.

6.2.31.28 PFPM_SAL0 - 0x001B

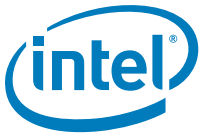
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.29 PFPM_SAL1 - 0x001C

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.30 PFPM_SAH0 - 0x001D

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.31 PFPM_SAH1 - 0x001E

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.32 PF allocations - VSI and VEB - 0x001F

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.33 PF allocations - MACs - 0x0020

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.34 PF allocations - outer tags - 0x0021

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569

6.2.31.35 PF allocations - inner tags - 0x0022

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.36 PF allocations - inner MACs - 0x0023

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.37 PF allocations - IPs - 0x0024

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.38 PF allocations - stats - 0x0025

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.39 PF allocations - mirror rules - 0x0026

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.40 PF allocations - queue sets - 0x0027

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.41 PF allocations - teredo - 0x0028

Defines the allocation of Teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.42 PF flags - 0x0029

For more details about the inner structure, see page 572.

6.2.31.43 PFPM_SAL0 - 0x002A

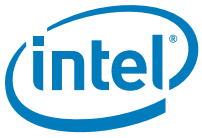
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.44 PFPM_SAL1 - 0x002B

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.45 PFPM_SAH0 - 0x002C

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.46 PFPM_SAH1 - 0x002D

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.47 PF allocations - VSI and VEB - 0x002E

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.48 PF allocations - MACs - 0x002F

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.49 PF allocations - outer tags - 0x0030

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.50 PF allocations - inner tags - 0x0031

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.51 PF allocations - inner MACs - 0x0032

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.52 PF allocations - IPs - 0x0033

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.53 PF allocations - stats - 0x0034

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.54 PF allocations - mirror rules - 0x0035

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.55 PF allocations - queue sets - 0x0036

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.56 PF allocations - teredo - 0x0037

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.57 PF flags - 0x0038

For more details about the inner structure, see page 572.

6.2.31.58 PFPM_SAL0 - 0x0039

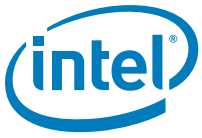
This register is loaded by FW to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.59 PFPM_SAL1 - 0x003A

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.60 PFPM_SAH0 - 0x003B

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.61 PFPM_SAH1 - 0x003C

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.62 PF allocations - VSI and VEB - 0x003D

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.63 PF allocations - MACs - 0x003E

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.64 PF allocations - outer tags - 0x003F

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, seepage 569.

6.2.31.65 PF allocations - inner tags - 0x0040

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.66 PF allocations - inner MACs - 0x0041

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.67 PF allocations - IPs - 0x0042

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.68 PF allocations - stats - 0x0043

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.69 PF allocations - mirror rules - 0x0044

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.70 PF allocations - queue sets - 0x0045

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.71 PF allocations - teredo - 0x0046

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.72 PF flags - 0x0047

For more details about the inner structure, see page 572.

6.2.31.73 PFPM_SAL0 - 0x0048

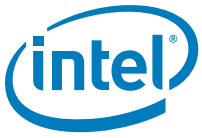
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.74 PFPM_SAL1 - 0x0049

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.75 PFPM_SAH0 - 0x004A

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.76 PFPM_SAH1 - 0x004B

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.77 PF allocations - VSI and VEB - 0x004C

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.78 PF allocations - MACs - 0x004D

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.79 PF allocations - outer tags - 0x004E

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.80 PF allocations - inner tags - 0x004F

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.81 PF allocations - inner MACs - 0x0050

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.82 PF allocations - IPs - 0x0051

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.83 PF allocations - stats - 0x0052

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.84 PF allocations - mirror rules - 0x0053

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.85 PF allocations - queue sets - 0x0054

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.86 PF allocations - teredo - 0x0055

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.87 PF flags - 0x0056

For more details about the inner structure, see page 572.

6.2.31.88 PFPM_SAL0 - 0x0057

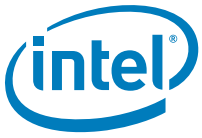
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.89 PFPM_SAL1 - 0x0058

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.90 PFPM_SAH0 - 0x0059

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.91 PFPM_SAH1 - 0x005A

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.92 PF allocations - VSI and VEB - 0x005B

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.93 PF allocations - MACs - 0x005C

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.94 PF allocations - outer tags - 0x005D

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.95 PF allocations - inner tags - 0x005E

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.96 PF allocations - inner MACs - 0x005F

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.97 PF allocations - IPs - 0x0060

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.98 PF allocations - stats - 0x0061

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.99 PF allocations - mirror rules - 0x0062

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.100 PF allocations - queue sets - 0x0063

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.101 PF allocations - teredo - 0x0064

Defines the allocation of teredo ports to each PF

For more details about the inner structure, see page 571

6.2.31.102 PF flags - 0x0065

For more details about the inner structure, see page 572.

6.2.31.103 PFPM_SAL0 - 0x0066

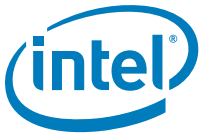
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.104 PFPM_SAL1 - 0x0067

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.105 PFPM_SAH0 - 0x0068

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.106 PFPM_SAH1 - 0x0069

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.107 PF allocations - VSI and VEB - 0x006A

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.108 PF allocations - MACs - 0x006B

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.109 PF allocations - outer tags - 0x006C

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.110 PF allocations - inner tags - 0x006D

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.111 PF allocations - inner MACs - 0x006E

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.112 PF allocations - IPs - 0x006F

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.113 PF allocations - stats - 0x0070

Defines the allocation of statistics pools to each PF.

For inner structure please See section 6.2.31.8

6.2.31.114 PF allocations - mirror rules - 0x0071

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.115 PF allocations - queue sets - 0x0072

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.116 PF allocations - teredo - 0x0073

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.117 PF flags - 0x0074

For more details about the inner structure, see page 572.

6.2.31.118 PFPM_SAL0 - 0x0075

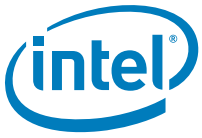
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.119 PFPM_SAL1 - 0x0076

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.120 PFPM_SAH0 - 0x0077

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.121 PFPM_SAH1 - 0x0078

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.122 PF allocations - VSI and VEB - 0x0079

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.123 PF allocations - MACs - 0x007A

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.124 PF allocations - outer tags - 0x007B

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.125 PF allocations - inner tags - 0x007C

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.126 PF allocations - inner MACs - 0x007D

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.127 PF allocations - IPs - 0x007E

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.128 PF allocations - stats - 0x007F

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.129 PF allocations - mirror rules - 0x0080

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.130 PF allocations - queue sets - 0x0081

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.131 PF allocations - teredo - 0x0082

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.132 PF flags - 0x0083

For more details about the inner structure, see page 572.

6.2.31.133 PFPM_SAL0 - 0x0084

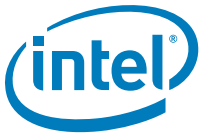
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.134 PFPM_SAL1 - 0x0085

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.135 PFPM_SAH0 - 0x0086

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.136 PFPM_SAH1 - 0x0087

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.137 PF allocations - VSI and VEB - 0x0088

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.138 PF allocations - MACs - 0x0089

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.139 PF allocations - outer tags - 0x008A

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.140 PF allocations - inner tags - 0x008B

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.141 PF allocations - inner MACs - 0x008C

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.142 PF allocations - IPs - 0x008D

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.143 PF allocations - stats - 0x008E

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.144 PF allocations - mirror rules - 0x008F

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.145 PF allocations - queue sets - 0x0090

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.146 PF allocations - teredo - 0x0091

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.147 PF flags - 0x0092

For more details about the inner structure, see page 572.

6.2.31.148 PFPM_SAL0 - 0x0093

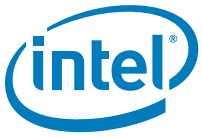
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.149 PFPM_SAL1 - 0x0094

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.150 PFPM_SAH0 - 0x0095

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.151 PFPM_SAH1 - 0x0096

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.152 PF allocations - VSI and VEB - 0x0097

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.153 PF allocations - MACs - 0x0098

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.154 PF allocations - outer tags - 0x0099

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.155 PF allocations - inner tags - 0x009A

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.156 PF allocations - inner MACs - 0x009B

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.157 PF allocations - IPs - 0x009C

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.158 PF allocations - stats - 0x009D

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see Note 6.2.31.8.

6.2.31.159 PF allocations - mirror rules - 0x009E

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.160 PF allocations - queue sets - 0x009F

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.161 PF allocations - teredo - 0x00A0

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.162 PF flags - 0x00A1

For more details about the inner structure, see page 572.

6.2.31.163 PFPM_SAL0 - 0x00A2

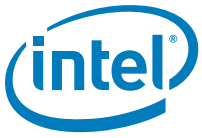
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:,0].

For more details about the inner structure, see page 572.

6.2.31.164 PFPM_SAL1 - 0x00A3

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.165 PFPM_SAH0 - 0x00A4

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.166 PFPM_SAH1 - 0x00A5

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.167 PF allocations - VSI and VEB - 0x00A6

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.168 PF allocations - MACs - 0x00A7

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.169 PF allocations - outer tags - 0x00A8

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For inner structure please See section 6.2.31.4 .

6.2.31.170 PF allocations - inner tags - 0x00A9

Defines the allocation of S-tags/Inner VLANs/Outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.171 PF allocations - inner MACs - 0x00AA

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.172 PF allocations - IPs - 0x00AB

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.173 PF allocations - stats - 0x00AC

Defines the allocation of statistics pools to each PF

For more details about the inner structure, see page 571.

6.2.31.174 PF allocations - mirror rules - 0x00AD

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.175 PF allocations - queue sets - 0x00AE

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.176 PF allocations - teredo - 0x00AF

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.177 PF flags - 0x00B0

For more details about the inner structure, see page 572.

6.2.31.178 PFPM_SAL0 - 0x00B1

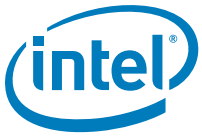
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.179 PFPM_SAL1 - 0x00B2

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.180 PFPM_SAH0 - 0x00B3

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:,0].

For more details about the inner structure, see page 572.

6.2.31.181 PFPM_SAH1 - 0x00B4

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.182 PF allocations - VSI and VEB - 0x00B5

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.183 PF allocations - MACs - 0x00B6

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.184 PF allocations - outer tags - 0x00B7

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.185 PF allocations - inner tags - 0x00B8

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.186 PF allocations - inner MACs - 0x00B9

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.187 PF allocations - IPs - 0x00BA

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.188 PF allocations - stats - 0x00BB

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.189 PF allocations - mirror rules - 0x00BC

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.190 PF allocations - queue sets - 0x00BD

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.191 PF allocations - teredo - 0x00BE

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.192 PF flags - 0x00BF

For more details about the inner structure, see page 572.

6.2.31.193 PFPM_SAL0 - 0x00C0

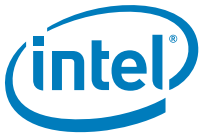
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.194 PFPM_SAL1 - 0x00C1

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.195 PFPM_SAH0 - 0x00C2

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.196 PFPM_SAH1 - 0x00C3

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.197 PF allocations - VSI and VEB - 0x00C4

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.198 PF allocations - MACs - 0x00C5

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.199 PF allocations - outer tags - 0x00C6

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.200 PF allocations - inner tags - 0x00C7

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.201 PF allocations - inner MACs - 0x00C8

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.202 PF allocations - IPs - 0x00C9

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.203 PF allocations - stats - 0x00CA

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.204 PF allocations - mirror rules - 0x00CB

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.205 PF allocations - queue sets - 0x00CC

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.206 PF allocations - teredo - 0x00CD

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.207 PF flags - 0x00CE

For more details about the inner structure, see page 572.

6.2.31.208 PFPM_SAL0 - 0x00CF

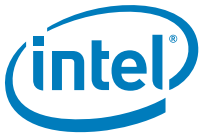
This register is loaded by FW to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.209 PFPM_SAL1 - 0x00D0

This register is loaded by FW to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.210 PFPM_SAH0 - 0x00D1

This register is loaded by FW to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.211 PFPM_SAH1 - 0x00D2

This register is loaded by FW to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.212 PF allocations - VSI and VEB - 0x00D3

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.213 PF allocations - MACs - 0x00D4

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.214 PF allocations - outer tags - 0x00D5

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.215 PF allocations - inner tags - 0x00D6

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570.

6.2.31.216 PF allocations - inner MACs - 0x00D7

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.217 PF allocations - IPs - 0x00D8

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.218 PF allocations - stats - 0x00D9

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.219 PF allocations - mirror rules - 0x00DA

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.220 PF allocations - queue sets - 0x00DB

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.221 PF allocations - teredo - 0x00DC

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.222 PF flags - 0x00DD

For more details about the inner structure, see page 572.

6.2.31.223 PFPM_SAL0 - 0x00DE

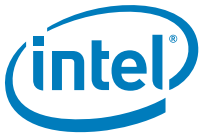
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.224 PFPM_SAL1 - 0x00DF

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.225 PFPM_SAH0 - 0x00E0

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.226 PFPM_SAH1 - 0x00E1

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.227 PF allocations - VSI and VEB - 0x00E2

Defines the allocation of VSIs and VEBs to each PF.

For more details about the inner structure, see page 569.

6.2.31.228 PF allocations - MACs - 0x00E3

Defines the allocation of MACs to each PF.

For more details about the inner structure, see page 569.

6.2.31.229 PF allocations - outer tags - 0x00E4

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 569.

6.2.31.230 PF allocations - inner tags - 0x00E5

Defines the allocation of S-tags/inner VLANs/outer VLANs to each PF. The type of resource allocated depends on the switching table configuration.

For more details about the inner structure, see page 570

6.2.31.231 PF allocations - inner MACs - 0x00E6

Defines the allocation of inner MACs to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.232 PF allocations - IPs - 0x00E7

Defines the allocation of IP to each PF - relevant only for cloud images.

For more details about the inner structure, see page 570.

6.2.31.233 PF allocations - stats - 0x00E8

Defines the allocation of statistics pools to each PF.

For more details about the inner structure, see page 571.

6.2.31.234 PF allocations - mirror rules - 0x00E9

Defines the allocation of mirror rules to each PF.

For more details about the inner structure, see page 571.

6.2.31.235 PF allocations - queue sets - 0x00EA

Defines the allocation of Tx queue sets to each PF.

For more details about the inner structure, see page 571.

6.2.31.236 PF allocations - teredo - 0x00EB

Defines the allocation of teredo ports to each PF.

For more details about the inner structure, see page 571.

6.2.31.237 PF flags - 0x00EC

For more details about the inner structure, see page 572.

6.2.31.238 PFPM_SAL0 - 0x00ED

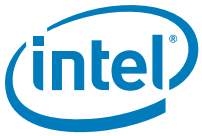
This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.239 PFPM_SAL1 - 0x00EE

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2].

For more details about the inner structure, see page 572.



6.2.31.240 PFPM_SAH0 - 0x00EF

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1:0].

For more details about the inner structure, see page 572.

6.2.31.241 PFPM_SAH1 - 0x00F0

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2].

For more details about the inner structure, see page 573.

6.2.31.242 CRC8 - 0x00F1

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used 0x1 = CRC used
14:8	Reserved	0x0	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: PF Allocations -> Section Length. End Section -> Word: PF Allocations -> PFPM_SAH1.

6.2.32 PHY capability data structure 0 section summary table

The PHY capabilities data structure contains all the parameters to control the internal and external PHY operation. For example, interface type and mode, EEE parameters, etc.

Data structure is repeated per port.

Word Offset	Description	Reference
0x0000	Section Length	604
0x0001	INT-EXT PHY Select	604
0x0002	Internal PHY Type	604
0x0003	External PHY Type	605
0x0004	PHY ID 0	605
0x0005	PHY ID 1	606
0x0006	Module Type 0	606
0x0007	Module Type 1	607
0x0008	PHY Capabilities Misc0	608

Word Offset	Description	Reference
0x0009	PHY Capabilities Misc1	608
0x000A	Reserved	609
0x000B	CRC8	609

6.2.32.1 Section length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in words of the section covered by CRC.

6.2.32.2 INT-EXT PHY select - 0x0001

Bits	Field Name	Default NVM Value	Description
15:4	Reserved	0x0	Reserved.
3:0	Int_Ext PHY	0x0	Valid values are: 0x0 = Internal PHY only. 0x1 = External 10GBASE-T PHY. 0x2 = External SFP+ module. 0x3 = External QSFP+ module.

6.2.32.3 Internal PHY type - 0x0002

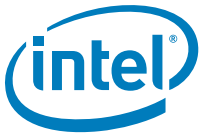
This parameter is used by firmware to find out the various internal PHY types to be supported on the Port. The XL710 supports multiple PHY types on the same port.

One of the PHY types might be enabled due to the result of auto-negotiation or manually set by firmware when auto-negotiation is disabled or not used.

One bit per PHY type.

Note: The 10GBASE-CR1 configuration is not an IEEE standard; however, this configuration could be enabled with switches supporting CR4 to obtain 4 x 10 Gb/s over QSFP+ module.

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	
11	10GBASE-CR1	0x0	
10	40GBASE-CR4	0x0	
9	XLPPI	0x0	
8	XLAUI	0x0	



Bits	Field Name	Default NVM Value	Description
7	SFI	0x0	
5	XAUI	0x0	
4	40GBASE-KR4	0x0	
3	10GBASE-KR	0x1	
2	10GBASE-KX4	0x0	
1	1000BASE-KX	0x0	
0	SGMII	0x0	

6.2.32.4 External PHY type - 0x0003

This parameter is used by firmware to find out the various external PHY types to be supported on the Port. The XL710 might support multiple PHY types on the same port.

One of the PHY types might be enabled due to the result of auto-negotiation or manually set by firmware when auto-negotiation is disabled or not used.

One bit per PHY type.

Bits	Field Name	Default NVM Value	Description
15:11	Reserved1	0x0	
10	40GBASE-LR4	0x0	
9	40GBASE-SR4	0x0	
8	40GBASE-CR4	0x0	
7	10GBASE-CR1	0x0	4 x 10 GbE QSFP + CR over direct attach copper.
6	10GBASE-SFP+	0x0	
5	10GBASE-LR	0x0	
4	10GBASE-SR	0x0	
3	10GBASE-T	0x0	
2	1000BASE-T	0x0	
1	100BASE-TX	0x0	
0	Reserved0	0x0	

6.2.32.5 PHY ID 0 - 0x0004

This parameter is used by firmware to verify the 10GBASE-T PHY-ID connected on the port.

The format of the PHY ID is defined in IEEE Std 802.3 and contains OUI, manufacturers model number and the revision number of the PHY.

Bits	Field Name	Default NVM Value	Description
15:8	PHY ID Byte1	0x0	

Bits	Field Name	Default NVM Value	Description
7:0	PHY ID Byte0	0x0	

6.2.32.6 PHY ID 1 - 0x0005

This parameter is used by firmware to verify the 10GBASE-T PHY-ID connected on the port.

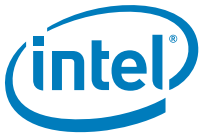
The format of the PHY ID is defined in IEEE Std 802.3 and contains OUI, manufacturers model number and the revision number of the PHY.

Bits	Field Name	Default NVM Value	Description
15:8	PHY ID Byte3	0x0	
7:0	PHY ID Byte2	0x0	

6.2.32.7 Module Type 0 - 0x0006

This parameter is used by firmware to find out the module type on the port when connected to external modules. For example, the XL710 might be connected to an SFP+ or QSFP+ optical or direct attached copper modules. The format of the module type returns the ID and extended ID fields as defined in the SFP+ or QSFP+ specifications.

Bits	Field Name	Default NVM Value	Description
15:8	10G/40G Compliance Code	0x0	SFF standard defines an eight byte field used to indicate the electrical or optical support modes. These are bit significant indicators that define the electronic or optical interfaces that are supported by the transceiver. At least one bit shall be set in this field. SFP standard defines this field in bytes 3-10 of the standard address space. SFP standard defines this field in bytes 131-138 of the standard address space. For 10 GbE SFP+ modules this byte is decoded as follows: Bit 0 = SFP + copper passive. Bit 1 = SFP + copper active. Bits 3-2 = Reserved. Bits 7:4 = Decoded according to the SFP+ 10 GBe Compliance Codes (byte 3). Bit 4 = 10GBASE-SR. Bit 5 = 10GBASE-LR. Bit 6 = Reserved. Bit 7 = Reserved. For 40 GbE QSFP+ modules, this byte is decoded based on the 10 GbE / 40 GbE compliance codes (byte 131).
7:0	Module Identifier	0x3	Defined by SFP+ (address 0xA0, byte 0) or QSFP+ (address 128, page 0) specifications. 0x3 = SFP+. 0xD = QSFP+. Valid values are: 0x3 = SFP+. 0xD = QSFP+.



6.2.32.8 Module Type 1 - 0x0007

This parameter is used by firmware to find out the module type on the port when connected to external modules. For example, the XL710 might be connected to an SFP+ or QSFP+ optical or direct attached copper modules. The format of the module type returns the ID and extended ID fields as defined in the SFP+ or QSFP+ specifications.

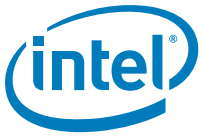
Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	Reserved.
7:0	1G Compliance Code	0x0	The SFP standard defines an 8-byte field used to indicate the electrical or optical support modes. These are bit significant indicators that define the electronic or optical interfaces that are supported by the transceiver. At least one bit must be set in this field. The SFP standard defines this field in byte 3-10 of the standard address space. The SFP standard defines this field in byte 131-138 of the standard address space. 1 GbE compliance codes are in SFP byte 6. 1 GbE compliance codes are in QSFP byte 134.

6.2.32.9 PHY capabilities Misc0 - 0x0008

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	
11	Enable Module Qualification	0x0	When set to 1b, the module qualification process is enabled.
10	Low Power Ability	0x1	Certain external PHYs connected to the port might have the ability to support low power mode. Setting the PHY in low power mode affects normal operation. 0x1 = PHY supports low power ability. 0x0 = Otherwise.
9:8	Pause Ability	0x3	This parameter is used by firmware to configure or advertise the IEEE 802.3x pause ability of the port. **Note:** The Link Flow Control (LFC or Link Pause) is enabled through pause ability bits during PHY auto-negotiation and the Priority Flow Control (PFC) is enabled through DCBX protocol. The LFC and PFC features are mutually exclusive. So, management should disable link pause on links where PFC should be enabled. Bit 0.0 - Set to 1b to enable IEEE 802.3x Tx link pause ability, or set to 0b to disable pause ability. Bit 0.1 = Set to 1b to enable IEEE 802.3x Rx link pause ability, or set to 0b to disable pause ability.
7:0	Link Speed	0x8	This parameter is used by firmware to configure or advertise various link speeds supported on the port. The XL710 might have the ability to support multiple link speeds on the same port. One of the link speeds might be enabled due to the result of auto-negotiation or manually set by firmware when auto-negotiation is disabled or not used. Link rate supported on the port: Bit 0.0 = Reserved. Bit 0.1 = 100 Mb/s. Bit 0.2 = 1 Gb/s. Bit 0.3 = 10 Gb/s. Bit 0.4 = 40 Gb/s. Bit 0.5 = Reserved. Other bits - Reserved, Must be zero.

6.2.32.10 PHY capabilities Misc1 - 0x0009

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	Reserved.
7:0	EEE Capability	0x40	This parameter is used by firmware to configure the EEE capability of various PHY types supported on the port. The XL710 might support multiple PHY types on the same port and EEE capability is indicated for each PHY type. One bit per PHY type. The XL710 might be capable of supporting multiple PHY types. The following parameter indicates the bit number. Bit 0.0 = Reserved. Bit 0.1 = EEE is supported for 100BASE-TX. Bit 0.2 = EEE is supported for 1000BASE-T. Bit 0.3 = EEE is supported for 10GBASE-T. Bit 0.4 = EEE is supported for 1000BASE-KX. Bit 0.5 = EEE is supported for 10GBASE-KX4. Bit 0.6 = EEE is supported for 10GBASE-KR. Other bits = Reserved, Must be zero.



6.2.32.11 Reserved - 0x000A

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

6.2.32.12 CRC8 - 0x000B

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Reserved	0x0	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: PHY Capability Data Structure 0 -> Section Length. End Section -> Word: PHY Capability Data Structure 0 -> Reserved.

6.2.33 External to internal PHY mapping 0 section summary table

This section provides the internal PHY mode to be selected for each external PHY/module counter part.

The XL710 provides multiple PHY interfaces for connecting different types of external PHYs and optical or copper modules. The end user can plug-in different types of modules into the SFP+ or QSFP+ connectors and choose from several connectivity options when using external PHYs. For example, a 10GBASE-T PHY could use a XAUI or KR connection to connect to the XL710.

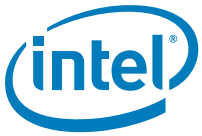
Firmware has to read the PHY or module ID and choose the appropriate interface to operate with that external module. Further when working with an external 10GBASE-T PHY auto-negotiation might result in several speed modes. For example, when a 10GBASE-T PHY can auto-negotiate to 1 GbE in which case firmware needs to reduce internal link to SGMII.

Word Offset	Description	Reference
0x0000	Section Length	610
0x0001	Mapping Word0	611
0x0002	Mapping Word1	611
0x0003	CRC8	612

6.2.33.1 Section length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in words of the section covered by CRC.



6.2.33.2 Mapping Word0 - 0x0001

Bits	Field Name	Default NVM Value	Description
15:14	10GBASE-CR1	0x0	Internal Interface For 10GBASE-CR1 External Module. 0x0 = XAUI (this option can be used with an external SFI PHY). 0x1 = CR1 (This option can be used for 4 x 10 Gb/s with QSFP+ modules). Other values reserved.
13:12	10GBASE-SFP+Cu	0x0	Internal Interface For 10GBASE-SFP+Cu External Module. 0x0 = XAUI (this option can be used with an external SFI PHY). 0x1 = SFI. Other values reserved.
11:10	10GBASE-LR	0x0	Internal Interface For 10GBASE-LR External Module. 0x0 = XAUI (this option can be used with an external SFI PHY). 0x1 = SFI. Other values reserved.
9:8	10GBASE-SR	0x0	Internal Interface For 10GBASE-SR External Module. 2-bit field [1:0]. 0x0 = XAUI (this option can be used with an external SFI PHY). 0x1 = SFI. Other values reserved.
7:6	10GBASE-T	0x0	Internal Interface For 10GBASE-TX External PHY. 0x0 = XAUI 0x1 = 10GBASE-KR. Other values reserved.
5:4	1000BASE-T	0x0	Internal Interface For 1000BASE-T External PHY. 0x0 = SGMII. Other values reserved.
3:2	100BASE-TX	0x0	Internal Interface For 100BASE-TX External PHY. 0x0 = SGMII. Other values reserved.
1:0	Reserved	0x0	Reserved.

6.2.33.3 Mapping Word1 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:6	Reserved	0x0	Reserved.
5:4	40GBASE-LR4	0x0	Internal Interface For 40GBASE-LR4 External Module. 0x0 = XLAUI (this option can be used with an external PHY). 0x1 = XLPPI (default for LR4 optical modules). Other values reserved.
3:2	40GBASE-SR4	0x0	Internal Interface For 40GBASE-SR4 External Module. 0x0 = XLAUI (this option can be used with an external PHY). 0x1 = XLPPI (default for SR4 optical modules). Other values reserved.
1:0	40GBASE-CR4	0x0	Internal Interface For 40GBASE-CR4 External Module. 0x0 = XLAUI (this option can be used with an external CR4 PHY). 0x1 = 40GBASE-CR4 (default for QSFP+ CR4 copper modules). Other values reserved.

6.2.33.4 CRC8 - 0x0003

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Reserved	0x0	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: External to Internal PHY Mapping 0 -> Section Length. End Section -> Word: External to Internal PHY Mapping 0 -> Mapping Word1.

6.2.34 LLDP configuration section summary table

Default settings to the embedded LLDP agent.

Word Offset	Description	Reference
0x0000	Section Length	612
0x0001	LLDP Admin Status	612
0x0002	msgFastTx	613
0x0003	msgTxInterval	613
0x0004	LLDP Tx parameters	613
0x0005	LLDP Initialization Timers	613
0x0006	CRC8	614

6.2.34.1 Section length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

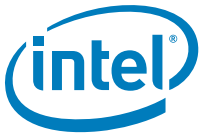
Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in words of the section covered by CRC.

6.2.34.2 LLDP admin status - 0x0001

Defines the status of an LLDP agent. Each LAN port has independent status.

3: Both receive and transmit enabled.

2: LLDP is configured for transmits only.



1: LLDP is configured for receives only.

0: LLDP agent is disabled.

Bits	Field Name	Default NVM Value	Description
15:12	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.

6.2.34.3 msgFastTx - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	msgFastTx	0x1	Time interval in timer ticks (seconds) between PDU transmits during fast transmits periods.

6.2.34.4 msgTxInterval - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	msgTxInterval	0x1e	The time in timer ticks (seconds) between transmissions during normal transmission.

6.2.34.5 LLDP Tx parameters - 0x0004

Bits	Field Name	Default NVM Value	Description
15:8	txCreditMax	0x5	Determines the 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.

6.2.34.6 LLDP initialization timers - 0x0005

Timers used by an LLDP agent during initialization and when to re-initialize. All times are in seconds.

Bits	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.

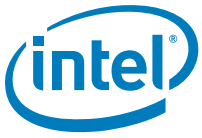
6.2.34.7 CRC8 - 0x0006

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Reserved	0x0	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: LLDP Configuration -> Section Length. End Section -> Word: LLDP Configuration -> LLDP Initialization Timers.

6.2.35 LLDP OEM TLVs Section Summary Table

A set of fixed-structure LLDP TLVs for EMP use.

Word Offset	Description	Reference
0x0000	Section Length	615
0x0001	TLV length	616
0x0002	TLV Data 0	616
0x0003	TLV Data 1	616
0x0004	TLV Data 2	616
0x0005	TLV Data 3	616
0x0006	TLV length	616
0x0007	TLV Data 0	616
0x0008	TLV Data 1	616
0x0009	TLV Data 2	617
0x000A	TLV Data 3	617
0x000B	TLV Data 4	617
0x000C	TLV Data 5	617
0x000D	TLV Data 6	617
0x000E	TLV length	617
0x000F	TLV Data 0	618
0x0010	TLV Data 1	618
0x0011	TLV Data 2	618
0x0012	TLV Data 3	618
0x0013	TLV Data 4	618
0x0014	TLV Data 5	618
0x0015	TLV Data 6	619
0x0016	TLV length	619
0x0017	TLV Data 0	619



Word Offset	Description	Reference
0x0018	TLV Data 1	619
0x0019	TLV Data 2	619
0x001A	TLV Data 3	619
0x001B	CRC8	619

6.2.35.1 Section length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in words of the section covered by CRC.

6.2.35.2 TLV data 0 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x05FE	Contains 16 bits of TLV Data. **Note:** The MS byte is the first byte on the wire.

6.2.35.3 TLV data 1 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x2400	Contains 16 bits of TLV Data. **Note:** The MS byte is the first byte on the wire.

6.2.35.4 TLV data 2 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0181	Contains 16 bits of TLV Data. **Note:** The MS byte is the first byte on the wire.

6.2.35.5 TLV data 3 - 0x0005

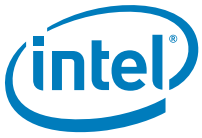
Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0000	Contains 16 bits of TLV Data. **Note:** The MS byte is the first byte on the wire.

6.2.35.6 TLV data 0 - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0CFE	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.7 TLV data 1 - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x2400	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.



6.2.35.8 TLV data 2 - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0381	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.9 TLV data 3 - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0000	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.10 TLV data 4 - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0000	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.11 TLV data 5 - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0000	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.12 TLV data 6 - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0000	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.13 TLV length - 0x000E

6.2.35.14 TLV data 0 - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0CFE	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.15 TLV data 1 - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x2400	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.16 TLV data 2 - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0481	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.17 TLV data 3 - 0x0012

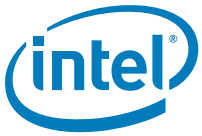
Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0000	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.18 TLV data 4 - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0000	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.19 TLV data 5 - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0000	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.



6.2.35.20 TLV data 6 - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0000	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.21 TLV data 0 - 0x0017

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x06FE	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.22 TLV data 1 - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x2400	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.23 TLV data 2 - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0581	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.24 TLV data 3 - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	TLV Data	0x0000	Contains 16 bits of TLV data. **Note:** The MS byte is the first byte on the wire.

6.2.35.25 CRC8 - 0x001B

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 CRC Not Used 0x1 CRC Used

Bits	Field Name	Default NVM Value	Description
14:8	Reserved	0x0	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: LLDP OEM TLVs -> Section Length. End Section -> Word: LLDP OEM TLVs -> TLV Data 3.

6.2.36 2nd free provisioning area section summary table

Free area used as the new bank when updating 8 KB long modules that are mapped outside shadow RAM.

Word Offset	Description	Reference
0x16000	[New Word]	620

6.2.36.1 [New Word] - 0x16000

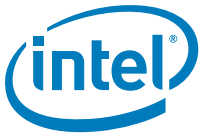
Raw data module length: variable.

6.2.37 OROM section summary table

Option ROM Module.

Contains pre-boot code and settings read by BIOS.

Word Offset	Description	Reference
0x0000	OROM Data	621
0x0001	moduleTypeL	622
0x0002	moduleTypeH	622
0x0003	headerLenL	622
0x0004	headerLenH	622
0x0005	headerVersionL	622
0x0006	headerVersionH	622
0x0007	moduleIDL	623
0x0008	moduleIDH	623
0x0009	moduleVendorL	623
0x000A	moduleVendorH	623
0x000B	dateL	623
0x000C	dateH	623



Word Offset	Description	Reference
0x000D	sizeL	624
0x000E	sizeH	624
0x000F	keySizeL	624
0x0010	keySizeH	624
0x0011	modulusSizeL	624
0x0012	modulusSizeH	624
0x0013	exponentSizeL	625
0x0014	exponentSizeH	625
0x0015	lad_srevL	625
0x0016	lad_srevH	625
0x0017	reserved1L	625
0x0018	reserved1H	625
0x0019	lad_fw_entry_offsetL	626
0x001A	lad_fw_entry_offsetH	626
0x001B	reserved2L	626
0x001C	reserved2H	626
0x001D	lad_image_unique_idL	626
0x001E	lad_image_unique_idH	626
0x001F	lad_module_idL	627
0x0020	lad_module_idH	627
0x0021 + 1*n, n=0...31	reserved	627
0x0041 + 1*n, n=0...127	RSA Public Key	627
0x00C1	RSA ExponentL	627
0x00C2	RSA ExponentH	627
0x00C3 + 1*n, n=0...127	Encrypted SHA256 Hash	628
0x0143	Device Blank NVM Device ID	628
0x0144	Max Module AreaL	628
0x0145	Max Module AreaH	628
0x0146	Current Module AreaL	628
0x0147	Current Module AreaH	628
0x0148	Format Version + CRC	629
0x0149	Code Revision	629
0x014A	Reserved Spare Word	629

6.2.37.1 OROM Data - 0x0000

Raw data module length: variable.

6.2.37.2 moduleTypeL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.2.37.3 moduleTypeH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.2.37.4 headerLenL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0xa1	

6.2.37.5 headerLenH - 0x0004

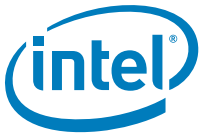
Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.2.37.6 headerVersionL - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.2.37.7 headerVersionH - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		



6.2.37.8 moduleIDL - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.2.37.9 moduleIDH - 0x0008

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.2.37.10 moduleVendorL - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.2.37.11 moduleVendorH - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.2.37.12 dateL - 0x000B

0xMMDD.

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.2.37.13 dateH - 0x000C

0xYYYY.

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.2.37.14 sizeL - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x00000800	

6.2.37.15 sizeH - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.2.37.16 keySizeL - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x40	

6.2.37.17 keySizeH - 0x0010

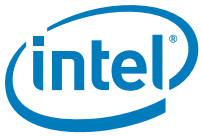
Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.2.37.18 modulusSizeL - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x40	

6.2.37.19 modulusSizeH - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		



6.2.37.20 exponentSizeL - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.2.37.21 exponentSizeH - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.2.37.22 lad_srevL - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.2.37.23 lad_srevH - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.2.37.24 reserved1L - 0x0017

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.37.25 reserved1H - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.37.26 lad_fw_entry_offsetL - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x14c	

6.2.37.27 lad_fw_entry_offsetH - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.2.37.28 reserved2L - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.37.29 reserved2H - 0x001C

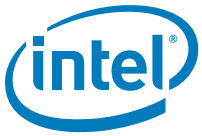
Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.37.30 lad_image_unique_idL - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.2.37.31 lad_image_unique_idH - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		



6.2.37.32 lad_module_idL - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0x5	Valid values are: 0x1 = EMP image. 0x2 = Reserved. 0x3 = PCIe analog. 0x4 = PHY analog. 0x5 = Option ROM.

6.2.37.33 lad_module_idH - 0x0020

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.2.37.34 reserved[n] (0x0021 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.37.35 RSA public key [n] (0x0041 + 1*n, n=0...127)

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.2.37.36 RSA ExponentL - 0x00C1

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.2.37.37 RSA ExponentH - 0x00C2

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.2.37.38 Encrypted SHA256 Hash[n] (0x00C3 + 1*n, n=0...127)

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.2.37.39 Device blank NVM device ID - 0x0143

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x154B	

6.2.37.40 Max module AreaL - 0x0144

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0xE000	

6.2.37.41 Max module AreaH - 0x0145

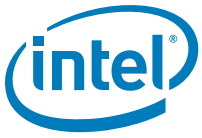
Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0006	

6.2.37.42 Current module AreaL - 0x0146

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x1000	

6.2.37.43 Current module AreaH - 0x0147

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	



6.2.37.44 Format version + CRC - 0x0148

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x0	Valid values are: 0x0 CRC Not Used 0x1 CRC Used
14:8	Format Version	0x2	
7:0	CRC8	0xFF	

6.2.37.45 Code revision - 0x0149

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.2.37.46 Reserved spare word - 0x014A

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

6.2.37.47 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.2.37.48 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.2.37.49 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0xa1	

6.2.37.50 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.2.37.51 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.2.37.52 headerVersionH - 0x0005

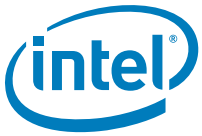
Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.2.37.53 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.2.37.54 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		



6.2.37.55 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.2.37.56 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.2.37.57 dateL - 0x000A

0xMMDD.

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.2.37.58 dateH - 0x000B

0xYYYY.

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.2.37.59 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x00037000	

6.2.37.60 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.2.37.61 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x40	

6.2.37.62 keySizeH - 0x000F

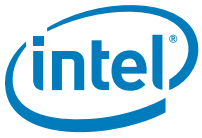
Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.2.37.63 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x40	

6.2.37.64 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		



6.2.37.65 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.2.37.66 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.2.37.67 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.2.37.68 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.2.37.69 reserved1L - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.37.70 reserved1H - 0x0017

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.37.71 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x14c	

6.2.37.72 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.2.37.73 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.37.74 reserved2H - 0x001B

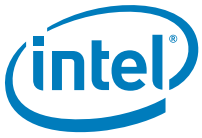
Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.37.75 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.2.37.76 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		



6.2.37.77 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0x2	Valid values are: 0x1 = EMP image. 0x2 = Reserved. 0x3 = PCIe analog. 0x4 = PHY analog. 0x5 = Option ROM.

6.2.37.78 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.2.37.79 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.37.80 RSA public key [n] (0x0040 + 1*n, n=0...127)

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.2.37.81 RSA ExponentL - 0x00C0

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.2.37.82 RSA ExponentH - 0x00C1

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.2.37.83 Encrypted SHA256 Hash[n] (0x00C2 + 1*n, n=0...127)

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.2.37.84 Device blank NVM device ID - 0x0142

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x154B	

6.2.37.85 Max module AreaL - 0x0143

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0xE000	

6.2.37.86 Max module AreaH - 0x0144

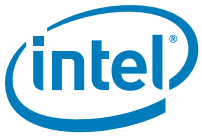
Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0006	

6.2.37.87 Current module AreaL - 0x0145

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0xE000	

6.2.37.88 Current module AreaH - 0x0146

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0006	



6.2.37.89 Format version + CRC - 0x0147

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Format Version	0x2	
7:0	Reserved		Reserved.

6.2.37.90 Code revision - 0x0148

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.2.37.91 Reserved spare word - 0x0149

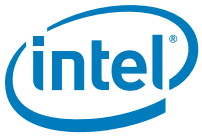
Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.2.37.92 EMP image raw data - 0x014A

Raw data module length: variable.

6.2.38 EMP image section summary table

Word Offset	Description	Reference
0x0000	moduleTypeL	640
0x0001	moduleTypeH	640
0x0002	headerLenL	640
0x0003	headerLenH	640
0x0004	headerVersionL	640
0x0005	headerVersionH	640
0x0006	moduleIDL	641
0x0007	moduleIDH	641
0x0008	moduleVendorL	641
0x0009	moduleVendorH	641
0x000A	dateL	641
0x000B	dateH	641
0x000C	sizeL	642
0x000D	sizeH	642
0x000E	keySizeL	642
0x000F	keySizeH	642
0x0010	modulusSizeL	642
0x0011	modulusSizeH	642
0x0012	exponentSizeL	643
0x0013	exponentSizeH	643
0x0014	lad_srevL	643
0x0015	lad_srevH	643
0x0016	reserved1L	643
0x0017	reserved1H	643
0x0018	lad_fw_entry_offsetL	644
0x0019	lad_fw_entry_offsetH	644
0x001A	reserved2L	644
0x001B	reserved2H	644
0x001C	lad_image_unique_idL	644
0x001D	lad_image_unique_idH	644
0x001E	lad_module_idL	645
0x001F	lad_module_idH	645
0x0020 + 1*n, n=0...31	reserved	645
0x0040 + 1*n, n=0...127	RSA Public Key	645
0x00C0	RSA ExponentL	645



Word Offset	Description	Reference
0x00C1	RSA ExponentH	645
0x00C2 + 1*n, n=0...127	Encrypted SHA256 Hash	646
0x0142	Device Blank NVM Device ID	646
0x0143	Max Module AreaL	646
0x0144	Max Module AreaH	646
0x0145	Current Module AreaL	646
0x0146	Current Module AreaH	646
0x0147	Format Version + CRC	647
0x0148	Code Revision	647
0x0149	Reserved Spare Word	647
0x014A	EMP Image Raw Data	647

6.2.38.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.2.38.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.2.38.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0xa1	

6.2.38.4 headerLenH - 0x0003

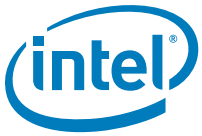
Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.2.38.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.2.38.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		



6.2.38.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.2.38.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.2.38.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.2.38.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.2.38.11 dateL - 0x000A

0xMMDD.

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.2.38.12 dateH - 0x000B

0xYYYY.

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.2.38.13 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x00037000	

6.2.38.14 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.2.38.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x40	

6.2.38.16 keySizeH - 0x000F

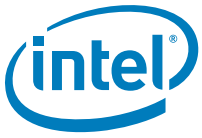
Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.2.38.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x40	

6.2.38.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		



6.2.38.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.2.38.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.2.38.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.2.38.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.2.38.23 reserved1L - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.38.24 reserved1H - 0x0017

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.38.25 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x14c	

6.2.38.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.2.38.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.38.28 reserved2H - 0x001B

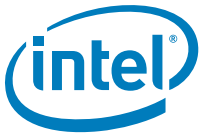
Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.38.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.2.38.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		



6.2.38.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0x1	Valid values are: 0x1 EMP Image 0x2 Reserved. 0x3 PCIe Analog 0x4 PHY Analog 0x5 Option ROM

6.2.38.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.2.38.33 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.2.38.34 RSA public key [n] (0x0040 + 1*n, n=0...127)

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.2.38.35 RSA ExponentL - 0x00C0

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.2.38.36 RSA ExponentH - 0x00C1

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.2.38.37 Encrypted SHA256 Hash[n] (0x00C2 + 1*n, n=0...127)

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.2.38.38 Device blank NVM device ID - 0x0142

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x154B	

6.2.38.39 Max module AreaL - 0x0143

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0xE000	

6.2.38.40 Max module AreaH - 0x0144

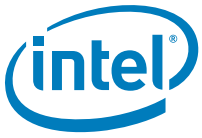
Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0006	

6.2.38.41 Current module AreaL - 0x0145

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0xE000	

6.2.38.42 Current module AreaH - 0x0146

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0006	



6.2.38.43 Format version + CRC - 0x0147

Bits	Field Name	Default NVM Value	Description
15	CRC Field Used	0x1	Valid values are: 0x0 = CRC not used. 0x1 = CRC used.
14:8	Format Version	0x2	Reserved.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word: EMP Image -> moduleTypeL. End Section -> Word: EMP Image -> EMP Image Raw Data.

6.2.38.44 Code revision - 0x0148

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.2.38.45 Reserved spare word - 0x0149

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

6.2.38.46 EMP image raw data - 0x014A

Raw data module length: variable.

6.2.39 1st free provisioning area section summary table

Free area used as the new bank when updating 1160 KB long modules.

Word Offset	Description	Reference
0x0000	[New Word]	647

6.2.39.1 [New Word] - 0x0000

Raw data module length: variable.

6.2.40 4th free provisioning area section summary table

Free area to be used as the new bank when updating future 64 KB long modules mapped outside shadow RAM.

Word Offset	Description	Reference
0x0000	[New Word]	648

6.2.40.1 [New Word] - 0x0000

Raw data module length: variable.

6.2.41 3rd free provisioning area section summary table

Free area to be used as the new bank when updating future 16 KB long modules mapped outside shadow RAM.

Word Offset	Description	Reference
0x0000	[New Word]	648

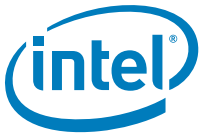
6.2.41.1 [New Word] - 0x0000

Raw data module length: variable.

6.2.42 FCoE scratch pad area section summary table

The FCoE scratch pad area used by software only. Accessing this module via NVM AQ commands requires using the flat addressing mode. For example, the module_pointer field set to 0 in the AQ command.

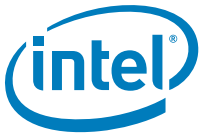
Word Offset	Description	Reference
0x0000	[New Word]	649



6.2.42.1 [New Word] - 0x0000

Raw data module length: variable.

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



7.0 Shared Resources

7.1 Receive Classification Filters

7.1.1 Introduction

This section describes the receive classification filters that direct inbound packets to a processing engine or cluster out of the following: LAN engine and FCoE engine. 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 which is defined by the switch filters detailed in Section 7.4.4.3.

The classification filters and other header processing units in the device inspect the first 480 bytes of the received packets. If the identified headers exceed 480 bytes, they are reported as an unidentified L2 packet type with no offloads other than Ethernet CRC. They are also handled this way by the classification filters.

7.1.1.1 Association with a Packet Engine

The general rule is that a frame is tested to be FCoE or LAN. The FCoE protocol uses the services of the LAN engine for certain frames. Such frames are forwarded to the LAN engine.

- FCoE frames are identified by their L2 Type value. The FCoE frames are checked against matched FCoE context. Exact rules on which FCoE packets are posted to the FCoE DDP queues vs. packets that are posted the LAN queues are described in Section 9.0. The FCoE context filters are described in Section 7.1.9.2 and FCoE LAN filters are described in Section 7.1.9.1.

7.1.1.2 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 programming interface of the filters are described in the following sections.

The following filters operate on received frames listed in priority ordering as illustrated in Figure 7-1. A frame can match multiple filters. If multiple filters define the same action, then the higher priority enabled matched filter takes precedence. If a frame does not match any of the filters below (with a queue action), the frame is directed to the default queue (queue zero of the VSI).

- Ethertype - see Section 7.1.5
- FCoE context filters - see Section 7.1.9.2
- Flow Director (FD) - see Section 7.1.8
- MAC/VLAN - see Section 7.1.5
- Hash filters (including RSS)

Note: In proper programming, the following filters are expected to be mutually exclusive. Meaning, the same packet is not expected to match more than one of them: Ethertype filter, FCoE context filter and FD filter.

The FCoE context filter are enabled per VSI by the VSIQF_CTL registers. The Hash filter is enabled per function per packet type by the PFQF_HENA for the PFs and the VFQF_HENA for the VFs. The Ethertype filter, MAC/VLAN filter and FD filter are enabled per PF and its VFs by the PFQF_CTL_0 registers.

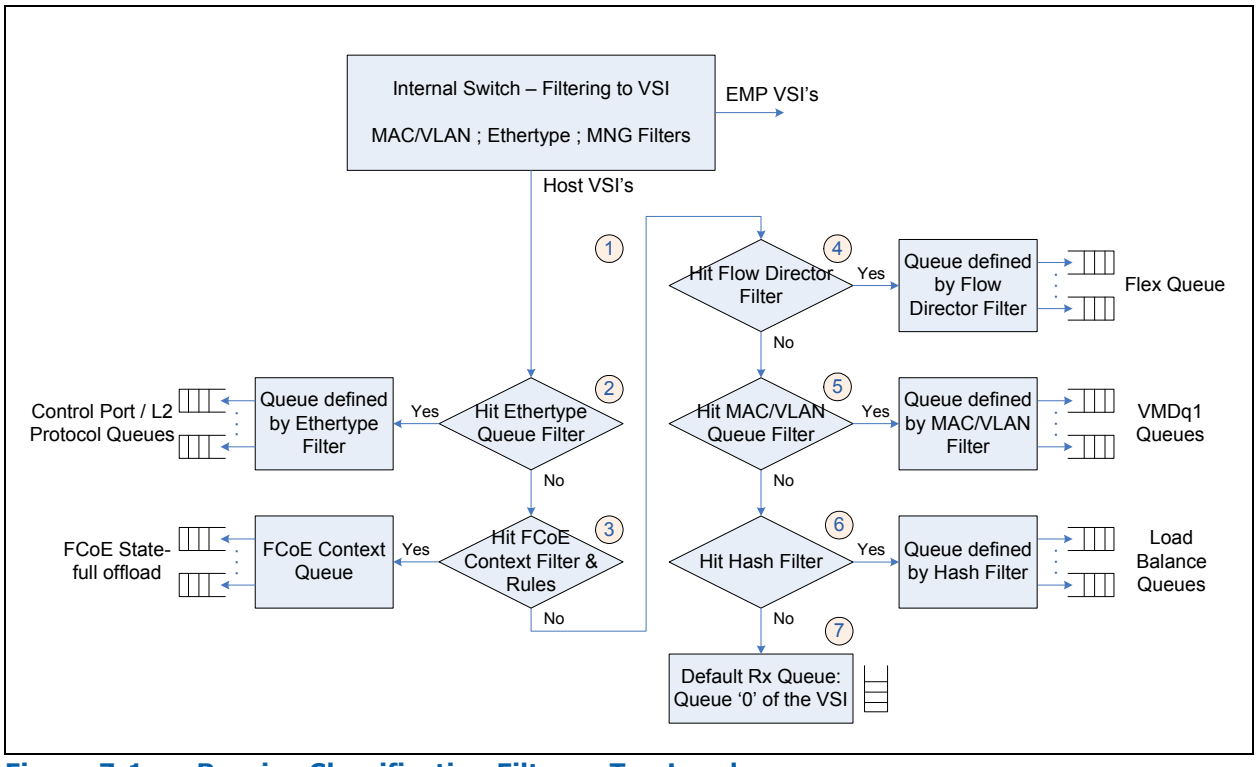


Figure 7-1. Receive Classification Filters - Top Level

The Table 7-1 below shows typical use cases of the classification filters and its programming option per function.

Table 7-1. Classification filters per PF/VF

VSI	MAC/VLAN, S-Tag and Cloud switch filters	Ethertype		FCoE Context	Flow Director	Hash
Main Usage	VMDq1	Control Port / L2 Protocols		DDP	Flex Filters ~~2~~1	Load Balance
Programming exposed to the PF	Yes	Yes		Yes	Yes	Yes
Programming exposed to the VF	No	No		No	No	Yes
1. There are two main usages for Flow Director filters: Software driver controlled filters for ATR and user/OS controlled filters.						

Once a frame matches a filter, the filter provides one or more actions to be performed on the frame. Listed below are those actions enabled by the filters while Table 7-2 indicates which of these actions are enabled per filter type:

- Accept / Reject - A decision whether the frame is dropped by the hardware or posted to the device engines for its processing and optionally posted to host memory.
- LAN / FCoE - A decision whether the frame is posted to LAN or FCoE engines.
- Receive Queue - Associate the frame to a queue index within the VSI space.
- Context ID - Match the frame to a FCoE context ID associated with the flow.
- Increment Statistical Counter - Increment the statistical counters associated with the filter.
- SW field (Filter ID) - Filter ID provided at filter programming and reported back with the matched received packets.
- Packet Bytes - Post bytes from the packet to the receive descriptor. Bytes that should be reported in the receive descriptor can be taken from the “Flexible Bytes” in the Field Vector.

Table 7-2 details the type of actions possible by each filter type.

Table 7-2. Filter actions by filter type

Action	MAC / VLAN / Ethertype / S-Tag / Cloud switch filters		FCoE Context	Flow Director	Hash
LAN/FCoE	LAN		FCoE / LAN	LAN	LAN
Receive Queue index / Context ID	Queue index within the VSI		DDP Context ID within the function space + LAN Queue index within the VSI	Queue index within the VSI	Queue index within the TC range of queues
Accept / Reject				Yes	
Increment Stat Counter (one of 512 GLQF_PCNT)				Yes	
Report Filter ID				Filter ID	Hash Signature
Report selected bytes from the receive packet				Programmable 4 or 8 bytes	

7.1.1.3 Receive Queue Index

As indicated in the previous section, one of the filter's action is the packet classification to LAN queues. The LAN queue index defined by the classification filters is a relative index within a specific VSI that is defined by the embedded switch. 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 to: VSILAN_QTABLE[n/2].QINDEX_0 for even 'n' and VSILAN_QTABLE[(n-1)/2].QINDEX_1 for odd 'n'.
- 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 4 queues in a single VSI with the following settings: VPLAN_QTABLE[0:3]=A,B,C,D (while A...D are the queue indexes in the PF space); VSILAN_QTABLE[0].QINDEX_0,QINDEX_1 = C,D and VSILAN_QTABLE[1].QINDEX_0,QINDEX_1=A,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.

The hash filter is a little bit more complex, while the queue index is a relative index within a specific range of queues (usually related to a TC) of a VSI.

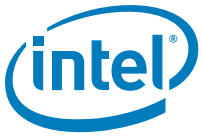
Listed below are some exception rules for the receive queue index:

- 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).
- If the packet has a MAC error (reflected by the RXE flag in the receive descriptor), the packet is directed to the default queue (queue zero of the VSI that is defined by the PRT_SBPVSI register). Note that a packet with MAC error is posted to the host only if enabled by the SBP flag in the PRT_SBPVSI register.
- The receive queue index that defines an "invalid" queue will cause dropping of the matched received packets. Such packets are counted by the GLV_REPC counter of the VSI.
- The following sub-bullets list the cases on which a queue index is considered as "invalid":
  - 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 ~~0x3FF~~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".

7.1.1.4 Initialization

This section lists all registers associated with the classification filters directly or indirectly and its required initialization. Note that some of the text in this section might not be clear, lacking a complete description of the registers. This description is given in the following sections as well as in the "Programming Interface" chapter. The initialization sequence should be executed according to the following order (described in the subsections below):

- Function Private Memory Allocation
- Queue Allocation
- Static Classification Filters Registers



- Dynamic Classification Filters Registers

7.1.1.4.1 Function Private Memory Allocation

The FCoE DDP 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 "7.9.3 Function Private Memory Space Configuration". These registers should be initialized per PF prior to any settings of the LAN queues and before any enablement of the FCoE filters.

7.1.1.4.2 Queue Allocation

LAN and FCoE DDP queues should be allocated to the functions before packets are directed to these queues. If this is not done, the packets are dropped.

7.1.1.4.3 Static Classification Filters Registers

Table 7-3. Static Classification Filters Registers Initialization

Register	Initialization Comments
PRTQF_CTL_0	Symmetric Hash enablement is loaded from the NVM
GLQF_CTL	Global Classification parameters are loaded from the NVM
VSIQF_CTL	Filtering enablement per VSI is programmed by the Add/Update VSI admin commands
PFQF_CTL_0	PF Classification parameters should be initialized before DDP contexts programming
VPQF_CTL	DDP filters parameters should be initialized after the PFQF_CTL_0 register of the PF and before DDP contexts programming
GLQF_SWAP and GLQF_HSYM	Swap registers and symmetric hash registers are loaded from the NVM
GLQF_NETKEY_PIT	The UDP network key structure is programmed by the "Set Network Key Structure" admin commands before any of the "Add Tunneling UDP" admin command are initiated
PRTQF_FLX_PIT	The field vector registers related to the flexible payload are either loaded from the NVM or programmed by the PFs at driver initialization time
PRTQF_FD_MSK	Mask Registers for the input set of the flexible payload for the FD filter is loaded from the NVM or programmed by the PFs at driver initialization time
VSIQF_TCREGION	The queue regions for 8 x TCs are programmed by the Add/Update VSI admin commands
PFQF_FDALLOC	FD filter table allocation to PFs is loaded from the NVM

7.1.1.4.4 Dynamic Classification Filters Registers

Table 7-4. Dynamic Classification Filters Registers Initialization

Register	Initialization Comments
PFQF_HENA and VFQF_HENA	PCTYPE enablement for the Hash filters are programmed by the function (PF or VF) at run time
PFQF_HREGION and VFQF_HREGION	Override the TC queue regions for specific PCTYPEs registers are programmed by the function (PF or VF) at run time
PFQF_HKEY, PFQF_HLUT, VFQF_HKEY and VFQF_HLUT	The hash key and LUT registers are programmed by the function (PF or VF) at run time

7.1.2 Packet Types and Input Set

The XL710 parses the first 480 bytes of the received packets. The hardware identifies the "protocol layers" upon which packet classification types (PCTYPE's) are defined. The supported PCTYPE's are listed in the Table 7-5 below. The fields used for the classification filters (named as "Input Set") are defined per filter per PCTYPE as shown in the Table 7-5. The table contains a shared column for the "Input Set" of the FD filter, FCoE DDP filter. The following section describes which PCTYPEs are relevant to each filter.

Table 7-5. Packet Classifier Types and its Input Sets

PCTYPE	PCTYPE Description	Hash Input Set	FD Input Set (1) FCoE DDP Input Set (2)
Reserved			
0 ... 27	Reserved for future use	-	-
[Inner] IPv4 packets			
28	Reserved		
29	Reserved		
30	Reserved		
31	NonF IPv4, UDP	IP4-S, IP4-D, UDP-S, UDP-D	IP4-S, IP4-D, UDP-S, UDP-D
32	Reserved		
32	NonF IPv4, TCP, SYN	IP4-S, IP4-D, TCP-S, TCP-D	IP4-D, TCP-D
33	NonFIPv4, TCP	IP4-S, IP4-D, TCP-S, TCP-D	IP4-S, IP4-D, TCP-S, TCP-D
34	NonFIPv4, SCTP	IP4-S, IP4-D, SCTP-Verification-Tag	IP4-S, IP4-D, SCTP-Verification-Tag
35	NonFIPv4, Other	IP4-S, IP4-D	IP4-S, IP4-D
36	Frag IPv4	IP4-S, IP4-D	IP4-S, IP4-D
37	Reserved	-	-
[Inner] IPv6 packets			
38	Reserved		
30	Reserved		
39	Reserved		
40	Reserved		
41	NonF IPv6, UDP	IP6-S, IP6-D, UDP-S, UDP-D	IP6-S, IP6-D, UDP-S, UDP-D
42	Reserved		
43	NonFIPv6, TCP	IP6-S, IP6-D, TCP-S, TCP-D	IP6-S, IP6-D, TCP-S, TCP-D
44	NonFIPv6, SCTP	IP6-S, IP6-D, SCTP-Verification-Tag	IP6-S, IP6-D, SCTP-Verification-Tag
45	NonFIPv6, Other	IP6-S, IP6-D	IP6-S, IP6-D
46	Frag IPv6	IP6-S, IP6-D	IP6-S, IP6-D
47	Reserved	-	-
FCoE			
48	FCoE OX (Data or RSP)	OX_ID	VLAN_ID; FC_D-ID; MAC-S; OX-ID; FC_TYPE
49	FCoE RX (Data or RSP)	OX_ID	VLAN_ID; FC_D-ID; MAC-S; RX-ID; FC_TYPE
50	FCoE other	OX_ID	No match

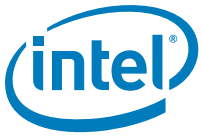


Table 7-5. Packet Classifier Types and its Input Sets

PCTYPE	PCTYPE Description	Hash Input Set	FD Input Set (1) FCoE DDP Input Set (2)
51	Reserved for future use	-	-
L2 Packet Types			
52 ... 62	Reserved for future use	-	-
63	Reserved	Reserved	Reserved
Note 1: All PCTYPE's are supported by the FD filter other than the FCoE OX and FCoE RX PCTYPE's Note 2: FCoE DDP filter support only the FCoE OX and FCoE RX PCTYPE's			

7.1.3 Filter's Input Set

The "input set" is defined per filter type as described below:

- Layer 2 filters that are used by the switch logic. These include: MAC filters; VLAN filters; S-Tag and Ethertype. The input set for these filters is defined in the "Internal Switch" chapter.
- The input set for the FD filter is flexible. Its input set for the 64 global PCTYPEs (excluding the FCoE PCTYPEs) has two components:
  - A pre-defined byte stream as shown in Table 7-5
  - A flexible input set defined by the PRTQF_FD_FLXINSET registers. A PRTQF_FD_FLXINSET register 'n' matches PCTYPE 'n' while each bit 'i' in the 8 bit INSET field in these registers enables word 'i' in the "flexible field vector". In case that only a portion of a 16 bit word is required for the input set, the non-relevant bits can be masked out by the PRTQF_FD_MSK registers.
- The input set for the Hash filter for the 64 global PCTYPEs is shown in Table 7-5.
- The input set for the FCoE context filter for the FCoE PCTYPEs is shown in Table 7-8.

7.1.4 Flexible Payload

The XL710 supports a "Flexible Payload" of 8 words (16 bytes) that are extracted from the payload of the packet. The 8 words can be extracted from up to 3 offsets within the payload. Payload is defined as follows (see also Table 1-1):

- If packet is identified as a L2 packet, payload is anything after the last EtherType. Else,
- If packet is identified as a L3 packet, payload is anything after the inner identified L3 header. Else,
- If packet is identified as a L4 packet, payload is anything after the inner identified L4 header.

The "flexible payload" is stored in a "Field Vector". This vector can be used by the FD filter as described in the following sections.

7.1.4.1 Flexible Payload

The XL710 parses the packets to protocol layers. For each protocol layer (Layers 2...4), The XL710 enables filtering on flexible payload which are defined by the PRTQF_FLX_PIT registers. The PRTQF_FLX_PIT registers can be either loaded from the NVM or programmed by the PF at driver initialization time.

7.1.4.2 PRTQF_FLX_PIT Programming

As previously indicated, the PRTQF_FLX_PIT registers map the Protocol layers Payload (Protocol layers = L2, L3, L4) to the Field Vector. During normal operation only one of the protocol layers' payload is meaningful so the entire flexible Payload in the field Vector is available for each of them.

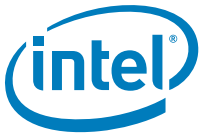
Table 7-6. PRTQF_FLX_PIT registers' fields

Field	Description
SOURCE_OFF	Source word offset in the mapped protocol layer starting from its beginning. Setting source offset rules: • The SOURCE_OFF plus FSIZE should not exceed byte 480 of a packet. • Must be programmed in ascending order: Current Offset >= previous offset + previous FSIZE. • The previous rule applies for all entries, including non-used ones.
DEST_OFF	Destination word offset in the field vector. The destination offset can be set to 50...57 matching offset 0...7 in the flexible field vector, respectively. Note that DEST_OFF plus FSIZE must not be greater than 58. For non-used registers, the DEST_OFF must be set to 0x63 (outside the range for active entries).
FSIZE	Field size defined in word units. For non-used registers, the FSIZE must be set to 0x1.

7.1.5 Layer 2 Classification Filter

The XL710 supports L2 filters (MAC / VLAN / L2 Ethertype) in the embedded switch. These filters act on the following layers: S-Tag; MAC address; VLAN and EtherType. These filters define a VSI and can define also a unique LAN queue within that VSI. The LAN queue is enabled by the ToQueue flag and the queue is defined by the "Queue Number" for each filter. These parameters are programmed by admin commands described in subsections of Section 7.4.9.5.8. The filters can be defined as the second or fifth priority filters for receive queue classification as follows:

Priority # illustrated in Figure 7-1	Filters defined by the following admin command(s).
2	Add Control Packet Filter.
5	Add MAC, VLAN pair ; Add S-tag; Add Cloud Filters.



Design limitation: Packet match to multiple L2 filters with active “ToQueue” flag that direct the packet to different queues is out of scope. Prioritization between the L2 filters in this case is defined by internal micro architecture rules that are not exposed. Therefore, software cannot rely on a specific arbitration and should avoid such cases for deterministic response.

7.1.6 Tunneling Fields

The XL710 supports tunneled packet formats including tunneled IP address, UDP Teredo, MAC in UDP, IP in GRE and MAC in GRE. These tunneling packets are supported for transmit and receive data processing offloads. When it comes to packet classification, it is based only on the encapsulated packet (exactly the same as non tunneled packets). Listed below are the supported tunneled packets:

IP tunneling — In case of non tunneled IP packet, there is only one IP header which is used for the packet classification. In case of tunneling, The inner IP header is used for packet classification while the outer IP header is used for the switch filters.

Teredo — Teredo are supported protocol layers for the classification filtering. See Section 7.1.6.1 for more details. Teredo support is mutually exclusive with MAC in UDP and GRE tunneling.

MAC in UDP — MAC in UDP packet format is illustrated in Section 8.2.3. MAC in UDP packet is identified by the destination UDP port number. The UDP ports used for MAC in UDP are programmed by the same admin commands as the Teredo UDP ports as described in Section 7.1.6.1. Classification to receive queues is based on the encapsulated packet according to its structure. Filtering to VSI for MAC in UDP packets is described in Section 7.4.4.5.3. MAC in UDP support is mutually exclusive with Teredo and GRE tunneling.

GRE Tunneling — IP in GRE and MAC in GRE packet formats are illustrated in Section 8.2.3. GRE tunneling is identified by the last Ethertype in the outer L2 header. IP in GRE or MAC in GRE are identified by the “Protocol Type” field in the GRE header. Filtering to VSI for MAC in UDP packets is described in Section 7.4.4.5.3 and Section 7.4.4.5.3. MAC in UDP support is mutually exclusive with Teredo and MAC in UDP.

7.1.6.1 Unique UDP Port Filters

There are “some” destination UDP port numbers that have unique meaning. These ports include 1588 packet identification, Teredo tunneling. XL710 supports the following port numbers:

- Two global port numbers for 1588 equal to 319 and 320.
- A “Tunneling UDP” table with 16 global port numbers for Teredo tunneling or MAC in UDP tunneling that are programmed at run time by admin commands described below. The “Tunneling UDP” port numbers should not be one of the values reserved for 1588 and should not be set to 0x0000 or 0xFFFF.

Note that XL710 does not process (offload) the checksum field in the tunneling UDP header.

The “Tunneling UDP” table is a global resource available for all PFs on a first come first served policy. This table defines the port number and the tunneling type as defined in Table 7-8. The PFs can add an entry or delete an entry by admin commands. Any add or remove command is acknowledged by a completion command that provides a status of the requested action and an updated status of the table resources.

A table entry is added by the first PF that program it. Any additional PF that programs the same UDP port number just “register” to the existing entry. Once a table entry is programmed, it affects received packets for all PFs on all LAN ports (regardless which PFs are “registered” to that UDP port number).

The admin commands used to program the tunneling headers are summarized in Table 7-7 and detailed in the following subsections:

Table 7-7. Tunneling UDP Admin commands

Name	Opcode	Ref	Admin Command Type
Add “Tunneling UDP”	0x0B00	Section 7.1.11.1	direct
Remove “Tunneling UDP”	0x0B01	Section 7.1.11.2	direct

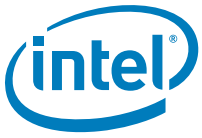
Table 7-8. Supported Tunneling Types

Tunneling Type	“Tunneling UDP Protocol Type”	Fixed Header Length [bytes]	Variable header Length	Network Key
Teredo	0x10 (next protocol is IP)	0x8	No	No
VXLAN	0x00 (next protocol is MAC)	0x10	No	Yes, see Section 7.1.6.1.1
Geneve	0x01 (next protocol is the optional Geneve header length)	0x10	Defined by bits 5:0 in byte 0x8 in the header	Yes, see Section 7.1.6.1.1
Place holder for Generic MAC in UDP	0x02 (next protocol is MAC)	Not Defined Yet	No	Yes, see Section 7.1.6.1.1
Reserved	Else	n/a	n/a	n/a

7.1.6.1.1 Network Key Structure of Tunneling header

There are 4 global network keys, shared for all LAN ports. The network keys are used to identify a cloud tenant. XL710 supports a programmable structure for these network keys that can be programmed by the PFs, using the “Program Network Key Structure” admin command. In NVGRE and Geneve headers it is defined as a 3 byte field at pre-defined offset in the header. The matched “Network Key Structure” parameters for the supported tunneling headers are shown in the table below:

Key Index	Tunneling Header	Key Offset	Key Length
0	VXLAN	12	3
1	Geneve	12	3
2	Place holder for “generic” MAC in UDP	offset 1: 8 <= Offset 1 < 32 offset 2: 8 + Offset 1 + Length 1<= Offset 2 < 64 - Length 2	0 <= Length 1, Length 2 <= 6 (*) Length 1 + Length 2 <= 10 (*) (*) Assuming word aligned keys
3	NVGRE	4	3



7.1.7 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 7-2 below. Listed below are some terms relating the hash filter.

Packet Classifier Type — The PCTYPE is the lowest index PCTYPE that is enabled for the function. The PCTYPE's are enabled for the hash filter by the PFQF_HENA registers for the PFs and by the VFQF_HENA registers for the VFs. See Table 7-5 for the PCTYPE's and its priority order.

Input Set — The input set for the hash filter are defined in the Table 7-5.

Hash Function — The 32 bit hash function is based on Toeplitz algorithm or simple XOR scheme as explained in Section 7.1.10. The hash can be calculated on the packet's fields as is or can be symmetric hash as explained in Section 7.1.10.3. The Hash signature is reported in the receive descriptor if the space is not "taken" by other filters as explained in Section 8.3.2.2. The hash is calculated only on the input set words. Masked bits within the input set words are replaced by zero's for the hash calculation.

Queue Index LUT — The queue index look up table (LUT) gets the LS bits of the hash output and provides a queue index within a "region". The LUT in each PF gets the 9 LS bits of the hash output having either 128 or 512 entries (as defined by the HASHLUTSIZE field in the PFQF_CTL_0 register). Each entry in the LUT defines receive queues in the range of 0 to 63. The LUT in each VF gets the 6 LS bits of the hash output having 64 entries while each entry defines receive queues in the range of 0 to 15. These LUTs are configured by the PFQF_HLUT and VFQF_HLUT registers for the PFs and VFs respectively.

Receive Queue Regions — The VSIs support 8 regions of receive queues that are aimed mainly for the TCs. The TC regions are defined per VSI by the VSIQF_TCREGION register. The region sizes (defined by the TC_SIZE fields) can be any of the following value: 1, 2, 4, 8, 16, 32, 64 as long as the total number of queues do not exceed the VSI allocation. These regions starts at the offset defined by the TC_OFFSET parameter. According to the region size, the 'n' LS bits of the Queue Index from the LUT are enabled. The Hash filter defines the queue index within the Queue Region of the VSI. The Queue Region is defined by the Traffic Class or the PCTYPE as programmed by the OVERRIDE_ENA and REGION fields in the PFQF_HREGION registers for the PFs and VFQF_HREGION registers for the VFs (as illustrated in Figure 7-2).

VSI Queue Index — The outcome VSI queue index is illustrated by the following expressions:

LUT_OUT + VSIQF_TCREGION.TC_OFFSET, while...

LUT_OUT = (VSIQF_TCREGION->TC_SIZE) LS bits of the QF_LUT

QF_LUT = PFQF_HLUT[9 LS bits of the packet hash] for the PF or

VFQF_HLUT[6 LS bits of the packet hash] for the VF

Outcome Queue Index

Mapping the VSI Queue index to the PF index space is done by the VSILAN_QTABLE or VSILAN_QBASE as explained in Section 8.2.1.2.

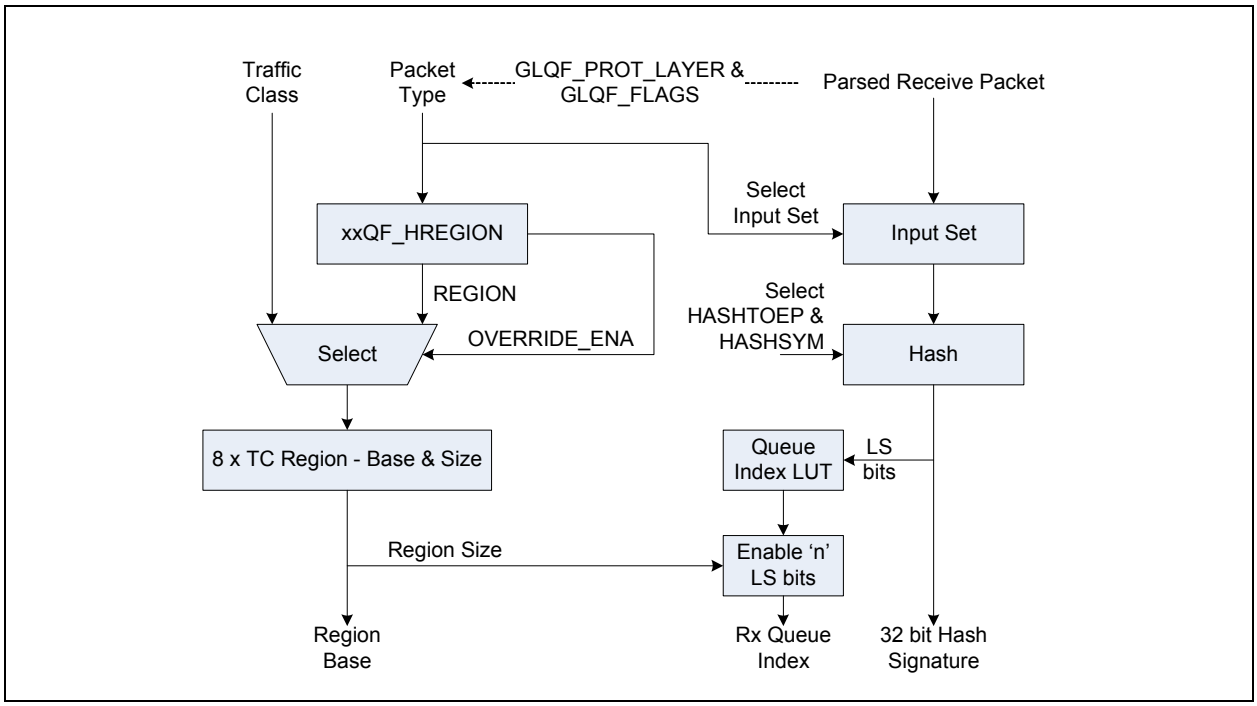


Figure 7-2. RSS Block Diagram

7.1.7.1 Comments for the Init Section

Dynamic setting options (expected at run time):

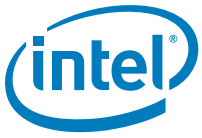
- VSIQF_TCREGION registers per VSI that define the Hash region sizes (should be dynamic b/c of the dynamic LQP allocation)
- PFQF_HLUT registers and VFQF_HLUT registers per function

7.1.8 Flow Director 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.

Programming

FD Filter programming is done by "Flow Director Programming Descriptor" described in Section 8.4.2.3 followed by packet structure that contains the filter fields as illustrated in Figure 7-3. Failed programming and removal of filter entries are reported by the "Programming Status" Descriptor described in Section 8.3.2.2.3. The packet that is used to program the filter can be optionally transmitted depending on the Dummy flag in the transmit data descriptor. Note also



that the packet used for the programming can be a single packet or part of a TSO. Removing a single filter is done by the same programming descriptor.

Filter programming is possible only from transmit queues that are enabled by the FDENA flag in the transmit queue context. Setting the FDENA flag is expected only for the PF queues. The PF can program a filter for any of its own VSIs or to VSIs that belong to its VFs. The target VSI for the matched packets is defined by the DEST_VSI parameter in the “Filter Programing Descriptor”. The PF is permitted to program a FD filter only if the FD filter is enabled by the FD_ENA flag in the PFQF_CTL_0 register. When using the PCTYPE in the FD programming descriptor, the software is permitted to set it only to those values enabled by the GLQF_FDENA registers. When the PCTYPE in the FD programming descriptor is not used (FD_AUTO_PCTYPE flag in the GLQF_CTL is set), the software is unaware of the PCTYPE. Programmed filters with unsupported PCTYPE by the GLQF_FDENA registers won't match any received packets.

The “source” and “destination” fields in the transmitted packet are presented in a reversed order with respect to the expected received packets. During filter programming time, the hardware swap these fields.

Attempt to re-program an existing filter entry updates the filters parameters.

At programming time the filter can be added on the “expense” of the “guaranteed” space or the “best effort” space of the function as explained in Section 7.1.8.1. In those cases, it is required to know up front that a filter programming is accepted by the hardware, the software should track its “guaranteed” space by reading the GUARANT_CNT in the PFQF_FDSTAT register.

In some cases the software may need to clear the FD table. Setting the CLEARFDTABLE flag in the PFQF_CTL_1 register, all entries of the PF are invalidated from the FD table. This action can be done during nominal transmission and reception. This action might take some time. Software should poll the FD filter counters of the PF in the PFQF_FDSTAT register till it is cleared as an indication that the “clear all PF entries” sequence is completed.

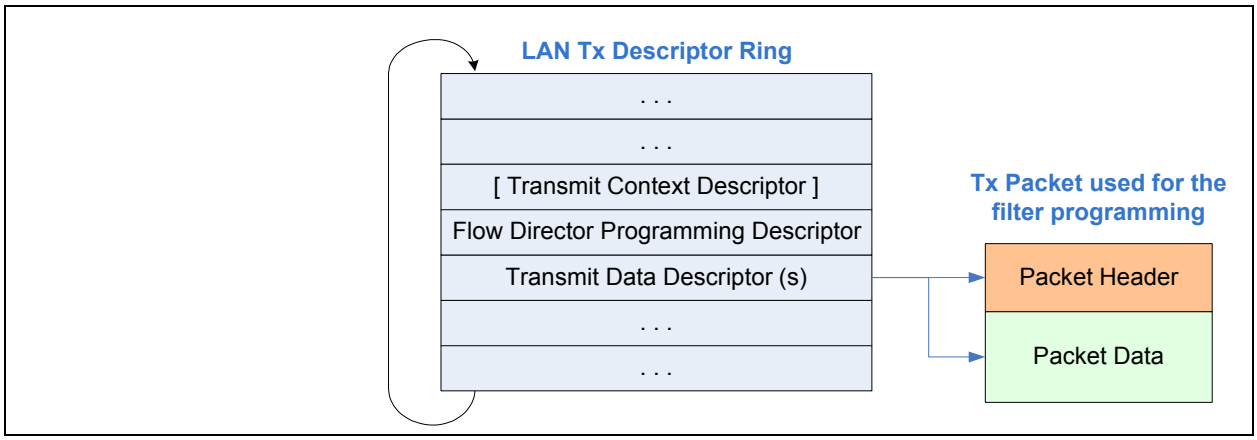


Figure 7-3. Flow Director filter Programming

Packet Classifier Type

All PCTYPE's described in Table 7-5 other than FCoE PCTYPE's are supported for the FD filter.

Input Sets

As opposed to all other filters, the input sets for the FD filter are defined per port enabling different input set for the flexible payload layers (by the PRTQF_FD_FLXINSET registers). The masking options for the flexible payload protocol layers are defined per port by the PRTQF_FD_MSK registers. These registers can be loaded from the NVM or programmed by software at init time. The Table 7-5 below shows the default setting that address the standard protocol layers.

FD filter entry context

Each filter entry is consisted of 64 bytes. As indicated above, the input set can be defined as large as 48 bytes while the other 16 bytes are reserved for the filter action, table management parameters and fields that are compared against the received packet:

— The PCTYPE and the DEST_VSI that are provided in the FD filter programming descriptor.

— The programming PF index.

— The QINDEX; FLEXOFF; STAT_CNT; FDID; DEST; FD_STATUS.

Filter match criteria

A packet matches a filter entry if the following conditions are met:

(1) The target VSI equals to the programmed DEST_VSI.

(2) The identified PCTYPE equals to the programmed PCTYPE.

(3) The received packet pattern matches the programmed one (the relevant fields for the PCTYPE as defined by the "FD input set" in Table 7-5).

Filter Action

The filter action is programmed as part of the filter programming command. The actions enabled for the FD filter are listed in Table 7-2. The filter action is taken when the packet matches the filter entry.

FD Table Size

The XL710 supports 8K x 64 byte filters. Out of these bytes, only 48 bytes can be used for the input set.

Hash and Buckets

The FD table is organized in buckets. The number of buckets equals to 16K (twice the number of supported filters). Each bucket might be: empty; include a single filter entry; or multiple filter entries. These buckets are managed autonomously by the hardware.

7.1.8.1 FD Table Allocation

The Flow Director table is a shared resource for all functions. The allocation between the PFs is loaded from the NVM to the PFQF_FDALLOC registers per PF. The FD supports 2 kinds of allocated spaces per PF: "guaranteed" space and "best effort" space. Each PF gets a private space which is a "guaranteed" number entries in the FD table (defined by the FDALLOC parameter). The sum of the "guaranteed" spaces of all PFs must be smaller than or equal to the size of the FD table. If the sum of the FDALLOC parameters is smaller than the size of the FD table, the rest of the space can be used by all function as "best effort". The FDBEST parameter in the PFQF_FDALLOC registers define the permitted use of the "best effort" space by each PF. The total size of the "best effort" space is defined by a global FDBEST in the GLQF_CTL register (loaded from the NVM). This parameter must not exceed the size of the FD table minus the sum of the FDALLOC parameters of all PFs.

Note: The maximum value of the global FDBEST parameter in the GLQF_CTL register can be set to 8K minus 32. In order to make use of all 8K filter entries, at least one of the PF's FDALLOC parameters should be larger than zero.

At filter programming the hardware tries both spaces: "guaranteed" and "best effort". Priority between usage of these spaces is set by the PROGPRIO flag in the global GLQF_CTL register. At filter invalidation the budget is gained back to either the "guaranteed" space or the "best effort" space according to a priority setting by the INVALPRIO flag in the global GLQF_CTL register.

The software can track the FD table population (best effort space as well as the guaranteed space). The GLQF_FDCNT_0 reports the total number of filter entries in the whole FD table while the PFQF_FDSTAT reports the number of filter entries used by the PF.

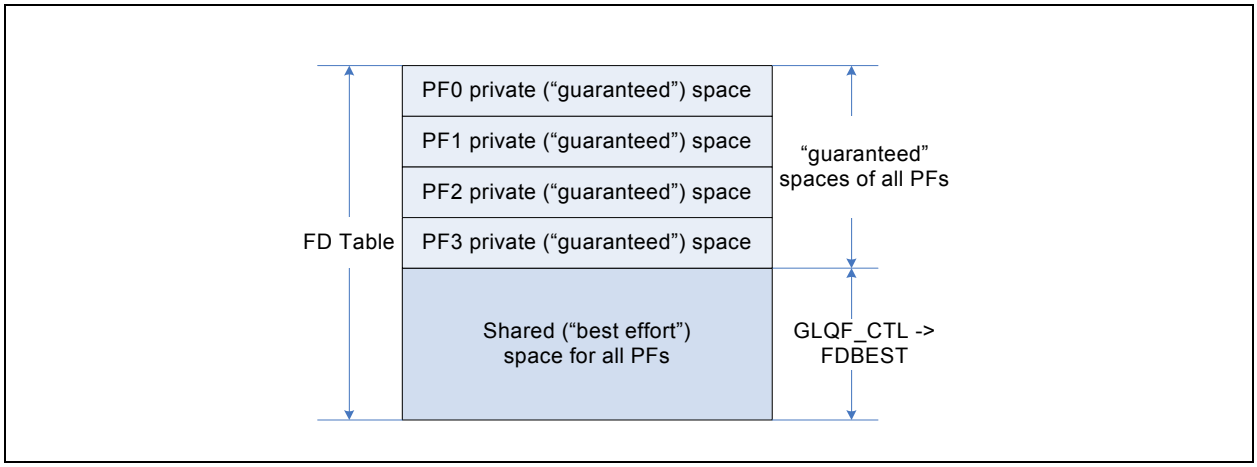


Figure 7-4. Flow Director Table Allocation (example for 4 x PFs: PF0 ... PF3)

7.1.8.2 Statistic Counters

The Flow Director filter may count packets by dedicated GLQF_PCNT counters.A filter can be associated to any of the GLQF_PCNT counters allocated to the PF by setting the CNT_ENA flag and the CNT_INDEX field in the "Filter Programming Descriptor". The statistics counters are allocated statically to the PFs according to the PCNT_ALLOC setting in the GLQF_CTL register, as follows:

PCNT_ALLOC setting	Enabled PFs	GLQF_PCNT counters per PF	GLQF_PCNT index range per PF	Physical GLQF_PCNT index in the device space
000b	Any number	Flexible	0 ... 511	All 9 bits of the index provided at filter programming. Allocation to PFs should be handled by the software. Possible ways to address it is by NVM setting readable to the software or any side band messages between the PFs which are outside the scope of this document.
100b	PF0 and PF1	256 = 512 / 2	0 ... 255	MS bit of the index is the LS bit of the PF index 8 LS bits of the index equals the 8 LS bits of the filter index provided at filter programming
101b	PF0 ... PF3	128 = 512 / 4	0 ... 127	2 MS bits of the index are the 2 LS bit of the PF index 7 LS bits of the index equals the 7 LS bits of the filter index provided at filter programming
110b	PF0 ... PF7	64 = 512 / 8	0 ... 63	3 MS bits of the index are the 3 LS bit of the PF index 6 LS bits of the index equals the 6 LS bits of the filter index provided at filter programming
111b	PF0 ... PF15	32 = 512 / 16	0 ... 31	4 MS bits of the index are the 4 LS bit of the PF index 5 LS bits of the index equals the 5 LS bits of the filter index provided at filter programming

7.1.9 FCoE Filters

This section describes the FCoE filter to LAN queues and filters to FCoE context (DDP).

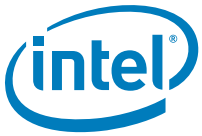
7.1.9.1 FCoE Filter to LAN Queues

The FCoE filter to LAN queues of packets that do not match an active FCoE context are aimed for load balance of FCoE packet processing among several processors. FCoE filtering to LAN queues is implemented by the hash filter.

Listed below are some comments specific for FCoE that relates to the hash filter:

Packet Classifier Type Relevant PCTYPE's are the FCoE_OX and FCoE_RX (PCTYPE's 48,49 in Table 7-5).

Input Sets The input set is the OX_ID.



7.1.9.2 FCoE Context Filter

The FCoE Context Filter is aimed to match specific FCoE flows with an active DDP context. A VSI 'n' is enabled for FCoE by the FCOE_ENA flag in the VSIQF_CTL[n] register. DDP offload is enabled only for the PFs. Therefore, the FCOE_ENA flag should not be set on VSI contexts belonging to VF and VM VSI's. The XL710 maintain some of these filter entries in internal cache while the others are stored in host memory (FPM). The FCoE context filter is composed of the following components:

Packet Classifier Type — The FCoE context filter supports 2 PCTYPEs: FCoE_OX and FCoE_RX (as defined by PCTYPEs 48,49 in Table 7-5).

Cache — Internal cache that can hold up to 1K filters.

Input Sets — The input set is defined for the FCoE PCTYPE by GLQF_FC_INSET[n] registers (while 'n' is FCoE PTYPE index shown in the Table 7-8 below).

FCoE Index	GLQF_PETYPE[Indx]	PCTYPE	Input Set and Mask
0	PCTYPE_INDEX = 48 PCTYPE_ENA = 1	FCoE_OX	VLAN_ID; FCoE D_ID; Source MAC Address; OX_ID; FC_TYPE
1	PCTYPE_INDEX = 49 PCTYPE_ENA = 1	FCoE_RX	VLAN_ID; FCoE D_ID; Source MAC Address; RX_ID; FC_TYPE

Filter match criteria — A packet matches a filter entry if the following conditions are met:

(1) The target function equals to the function that programmed the DDP context.

(2) The identified PCTYPE equals to the programmed PCTYPE.

(3) The received packet pattern matches the relevant Input Set.

Filter Action — The filter provides a hit/miss indication. In the case of hit, the packet is processed by the FCoE context while in the case of miss the packet is posted to the LAN queues. The Filter includes additional parameters of the FCoE context required for the FCoE processing. It also includes the LAN queue index which is used for some packets reported to the LAN queues as detailed in Chapter 9.0.

LAN Queue Index — Packets that match an FCoE context are handled by the FCoE context and are candidates for DDP offload. In some cases these packets or the packet's headers are posted to the LAN queue as described in the FCoE chapter. The LAN queue is defined in the FCoE context indicating an index within the VSI space (the same as the FD filter).

Context Table Size — The XL710 supports up to 4K FCoE contexts per function. Each PF can define a smaller (DMA) context table by the FCDSIZE parameters in the PFQF_CTL_0 register. At filter programming time, the software defines a unique context index. If the programmed "DDP Context Index" exceeds the FCDSIZE size, the programmed context is rejected.

Hash and Buckets — The FCoE context filter table is organized in buckets. The number of supported buckets is set per PF by the FCHSIZE in the PFQF_CTL_0 register. The bucket is defined by a hash function on the "Input Set" for the filter. Note that the software must clear (write all zero's) to the "bucket space" in host memory before enabling the table usage.

FPM Size Limitation

The total number of DDP contexts for the PF are defined by the FCDSIZE in the PFQF_CTL_0 registers. This number should not exceed the FPM allocation defined by the GLHMC_FCOEDDPCNT register of the PF.

In addition, the total number of DDP filters for the PF is defined by FCHSIZE + FCDSIZE in the PFQF_CTL_0 registers. This number should not exceed the FPM allocation defined by the GLHMC_FCOEFCNT register of the PF.

Filter Search

The PCTYPE is identified, its tuples are extracted and the hash is calculated. A filter is searched only within the matched function which is defined by the switch filters. The filter is searched in the cache. If the filter is not in cache, the hardware fetches the filter from the host memory space of the relevant function at location indicated by the calculated hash. In case of "exact" match then the process is completed and the filter is fetched to the cache.

In case of a miss, the next filter in the same bucket is fetched to the hardware repeating the above step till the end of the bucket or a matched filter is found. If the filter is not found in the whole bucket, it is considered as "miss".

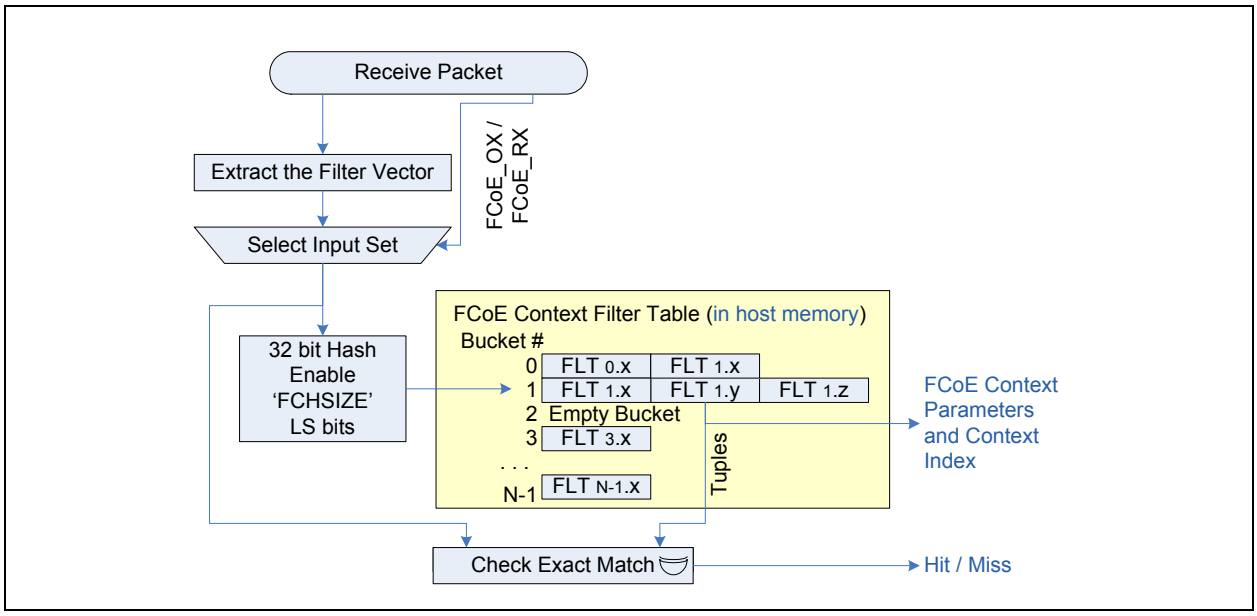
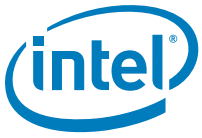


Figure 7-5. FCoE Context Filter Table

Filter Programming

A filter can be added or removed from the FCoE context filter table by any function that owns a filter. Filter programming is done as part of the DDP context programming executed from the transmit LAN queue (detailed in Section 9.4.1.1 and Section 8.4.2.4). During nominal operation the filter and the complete DDP context are removed autonomously by the device as described in Section 9.3.2.1.1 and Section 9.3.2.1.2. In some cases the software is required to remove a DDP context (including the filter). The software removes a DDP context using the same interface as the context programming. During context



programming and removing, the device distinguish between originator and responder contexts by the "Exchange Context" flag in the programming packet's header: '0' stands for originator context and '1' stands for responder context.

The packet that is used to program the filter can be optionally transmitted depending on the Dummy flag in the transmit data descriptor.

The "source" and "destination" fields in the transmitted packet are presented in a reversed order with respect to the expected received packets. During filter programming time, the hardware swap these fields.

An attempt to re-program an existing filter entry is rejected by the hardware.

Note that the PF software is expected to initialize (set to zero) the whole space in the FPM assigned for the FCoE filter table. This step should be part of the software driver init flow. The initialized FPM provides an indication to the hardware that the table is empty.

7.1.9.2.1 Comments for the Init Section

The size of the FCoE DDP context table defined by the FCDSIZE and FCHSIZE parameters are static (in the PFQF_CTL_0 register). They should be set at driver init flow before DDP is enabled by the FCOE_ENA flag in the VSIQF_CTL register.

7.1.10 Hash Functions

The XL710 supports several hash functions to be used by the various filters:

Microsoft* Toeplitz based hash vs. Simple hash. These hash methods are explained in the following subsections. Selection between the two schemes is controlled by the global GLQF_CTL register, as follows:

  - The HTOEP controls the scheme used by the hash filters for all PCTYPE's other than FCoE
  - The HTOEP_FCOE controls the scheme used by the hash filters for FCoE packets
- Symmetric hash (explained in Section 7.1.10.3)

The following notation is used to describe the hash functions:

- Ordering is little 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 (including both of them) of the incoming packet, where byte 0 is the first byte of the IP header. In other words, we consider all byte-offsets as offsets into a packet where the framing layer header has been stripped out. Therefore, the source IPv4 address is referred to as @12-15, while the destination v4 address is referred to as @16-19.
- **@x-y, @v-w** denotes concatenation of bytes x-y, followed by bytes v-w, preserving the order in which they occurred in the packet.

7.1.10.1 Microsoft Toeplitz Based Hash

The hash uses a random secret key of 416 bits (52 bytes). The key is defined per function by the PFQF_HKEY and VFQF_HKEY registers for the PFs and VFs respectively.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]

While 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 left-
most bit of K

    }

    return Result;
```

7.1.10.1.1 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 7-9. Examples of Input Fields for the Hash function

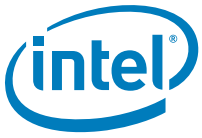
Packet Type	Hash Input	Hash Result
IPv4 with TCP or UDP	SourceAddress, DestinationAddress, SourcePort, DestinationPort: Input[12] = @12-15, @16-19, @20-21, @22-23	Result = ComputeHash(Input, 12)
IPv4 without L4	SourceAddress, DestinationAddress: Input[12] = @12-15, @16-19	Result = ComputeHash(Input, 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)

7.1.10.1.2 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
```

Table 7-10. 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 may not make sense as a tuple.

Table 7-11. IPv6

Destination Address / Port	Source Address / Port	IPv6 only	IPv6 with TCP
3ffe:2501:200:3::1 / 1766	3ffe:2501:200:1fff::7 / 2794	0x2cc18cd5	0x40207d3d
ff02::1 / 4739	3ffe:501:8::260:97ff:fe40:efab / 14230	0x0f0c461c	0xdde51bbf
fe80::200:f8ff:fe21:67cf / 38024	3ffe:1900:4545:3:200:f8ff:fe21:67cf / 44251	0x4b61e985	0x02d1feef

7.1.10.2 Simple Hash

Simple hash is an alternative hash function that trades runtime for effectiveness. It is provided for applications that recompute the hash in software and wish a lighter version of the hash function. Simple hash provides a more skewed distribution vs. the Microsoft hash. It is enabled only for the hash filter by the HTOEP flag and HTOEP_FCOE flag in the global GLQF_CTL register.

Simple hash is performed by partitioning the sequence of input bits into 32-bit entities and XOR those to generate a 32-bit output.

```
ComputeSimpleHash(input[], N)

    For input[] of length N bytes (8N bits)

    Result = 0;

    For each DWord (32 bits) in input[] {

        Result ^= (left-most 32 bits of input);

        shift input left 32 bit position;

    }

return Result;
```

The input[] vector is padded by 0 to 3 bytes of zero’s making it whole number of DWords. In those cases that the hash function is used as an address to a lookup table smaller or equal then 64K entries, the 16 MS bits of the results are XORed with the 16 LS bits creating a 16 bit output. In case that the hash function is used as an address to a lookup table smaller or equal than 256 entries, the 4 bytes of the above results are XORed together creating an 8 bit output.

7.1.10.3 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 frame between two addresses to the same queue independent of the direction of transmission.

The symmetric hash is useful for field pairs like the IP Addresses, L4 port numbers, MAC addresses, FC_IDs and FC Exchange IDs. The field pairs are defined globally per packet type by the GLQF_SWAP registers. The symmetric hash is obtained by replacing the original source and destination fields by a XOR value of these fields before calculating the hash.

Symmetric hash is enabled per PCTYPE by the GLQF_HSYM and per port by the HSYM_ENA flag in the PRTQF_CTL_0 register.

7.1.10.3.1 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.

7.1.11 Rx filter admin queue commands

7.1.11.1 Add “Tunneling UDP” (0x0B00)

Add “Tunneling UDP” admin command (shown in Table 7-12) is used to define a tunneling UDP filter. It defines the tunneling UDP port number and its header size, associating the filter to the PF that initiated the “add” command. The device response to this command is described below:

- If the requested “UDP Port” is already defined for the PF then report a completion with “EEXIST” error code.
- Else, if the requested “UDP Port” is already defined with different “Tunneling UDP Protocol Type” then report a completion with “EMODE” error code.
- Else, if the requested “UDP Port” is already defined with matched parameters for other PF(s) then update the filter entry:
  - Set the matched PF flag of the filter
  - Report back a successful programming completion
- Else, if the requested “UDP Port” does not exist then add a new filter entry:
  - Search for a free entry. If there is no free entry in the table then report completion with “ENOSPC” error code.
  - Else, add the entry to the table:
    - Set the matched PF flag of the filter
    - Report back a successful programming completion

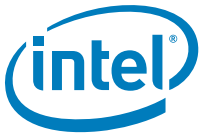


Table 7-12. Add "Tunneling UDP" Command (Opcode: 0x0B00)

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0B00	Command opcodes
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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
UDP Port	16-17		A 16-bit value of the UDP Port number to be added (while byte 17 is the MS byte which is first on the wire)
Reserved	18		Reserved
Reserved	19		Reserved
Reserved	20		Reserved
Tunneling UDP Protocol Type	21		Tunneling UDP type should be programmed as follow: 0x00 for VXLAN key 0x01 for Geneve key (next protocol is the optional Geneve header length) 0x02 Reserved option for MAC in UDP tunneling 0x10 for Teredo Else reserved
Reserved	22-31		

Table 7-13. Completion for the Add "Tunneling UDP" Command (Opcode: 0x0B00)

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0800	Command opcode
Datalen	4-5	0x00	N/A
Return value/VFID	6-7		Some comments on specific errors (see Section 7.10.9 for the errors encoding): No Error = no error EEXIST = Add Filter rejected because it already exist ENOSPC = Add Filter rejected because of no space EMODE = Add Filter rejected because the filter parameters in the Add "Tunneling UDP" Command do not match the programmed parameters in an existing filter with the same "UDP port"
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
UDP Port	16-17		A 16-bit value of the UDP Port number that was added
Filter Entry Index	18		Index of the added Tunneling UDP filter (0 to 15). This index is assigned by the device to be used by the PF to remove the filter.

Table 7-13. Completion for the Add "Tunneling UDP" Command (Opcode: 0x0B00)

Name	Bytes.Bits	Value	Remarks
Multiple PFs	19		This field equals to 0x1 when other PF(s) own the same filter entry and equals to 0x0 otherwise.
Total Filters	20		Total number of "Tunneling UDP" filters used by all PFs after the completion of the add filter command
Reserved	21-31		Reserved

7.1.11.2 Remove Tunneling UDP (0x0B01)

Remove "Tunneling UDP" admin command (shown in Table 7-15) is used to delete a tunneling UDP filter from the table. The device response to this command is described below:

- If the requested filter (defined by the "Filter Entry Index") is not owned by the PF then report an "ENOENT" error code.
- Else (the requested filter to be removed is owned by the PF), clear an internal flag that associates the filter entry with the PF and report back successful filter removal completion
  - If the filter is not associated with any PFs then set the "Entry Free" flag in the reported completion status.

Table 7-14. Remove "Tunneling UDP" Port Command (Opcode: 0x0B01)

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0B01	Command opcodes
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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-17		Reserved zero
Tunneling UDP filter Index	18.0:3		Filter Entry Index to be removed (0 to 15)
Reserved	18.4-31		reserved zero

Table 7-15. Completion for Remove "Tunneling UDP" Command (Opcode: 0x0B01)

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.2 for details.
Opcode	2-3	0x801	Command opcode
Datalen	4-5	0x00	N/A
Return value/VFID	6-7		Some comments on specific errors (see Section 7.10.9 for the errors encoding): No Error = no error ENOENT = Remove Filter rejected because no matched entry was found

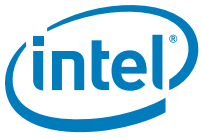
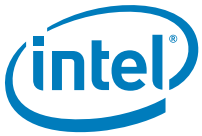


Table 7-15. Completion for Remove "Tunneling UDP" Command (Opcode: 0x0B01)

Name	Bytes.Bits	Value	Remarks
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
UDP Port	16-17		A 16-bit value of the UDP Port number
Filter Entry Index	18		The index of the requested filter entry to be removed (between 0 to 15).
Multiple PFs	19		This field equals to 0x1 when other PF(s) still own the same filter entry and equals to 0x0 otherwise.
Total Filters	20		Total number of "Tunneling UDP" filters used by all PFs after the completion of the remove filter command
Reserved	21-31		Reserved zero

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



7.2 L2 Packet Processing

7.2.1 CRC Handling

7.2.1.1 Ethernet CRC Insertion

XL710 calculates and inserts the Ethernet CRC for all packets transmitted to the network according to a per port setting configured through *Set MAC Config <CRC enable>* Admin Command.

7.2.1.2 Ethernet CRC Stripping

XL710 checks the integrity of the Ethernet CRC and possibly strip 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 FCoE DDP queues are striped from the Ethernet CRC (with no setting option).

7.2.2 Padding

Transmit packets to the network are padded with zero's (by the MAC unit) 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. Padding might be needed if the device strips off some fields from the packet on top of the CRC bytes.

Receive packets to the host are padded as well. Padding to host memory is enabled by the RX_PAD_EN flag in the GL_RDPU_CNTRL register. 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 stripped fields (like VLAN) that their remaining length is shorter than 60 bytes.
- Same rules for packets destined to the EMP.
- Packets destined to FCoE DDP queues are never padded.

7.2.3 L2 Tag Handling

7.2.3.1 Overview

In Section 7.4.6.1.3, different options to manipulate the L2 tags are described. This section describes the generic mechanism used to handle L2 tags in the XL710.

The XL710 supports up to 8 tags. In general, the text below 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 7-18. 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 the figure below.

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.

See Section 7.2.3.4 for the programming interface to describe the supported tags and the way to handle them.

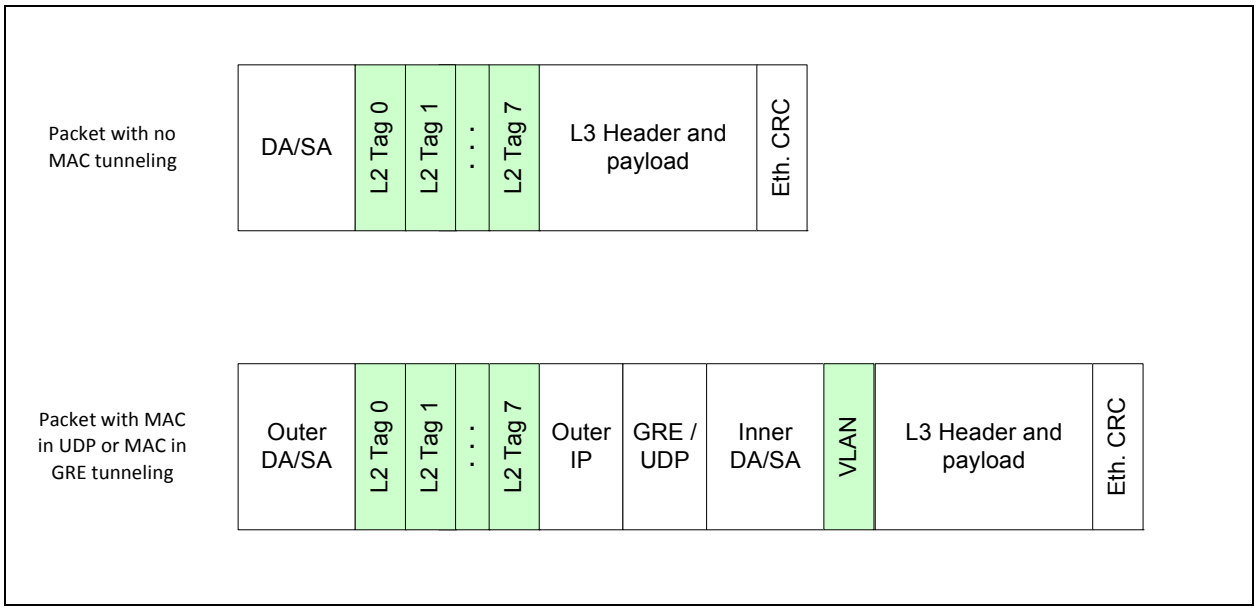


Figure 7-6. L2 tags order

7.2.3.2 Transmit Tag Handling

The transmit tag handling has three logical parts as described in the following diagram:

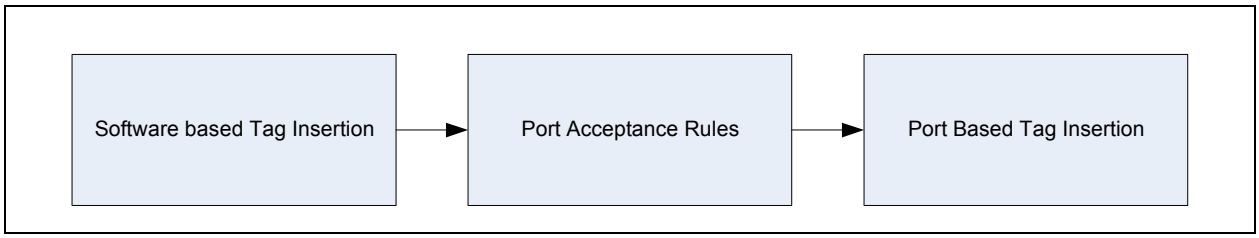
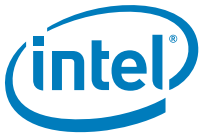


Figure 7-7. Transmit Tag Handling

The software based tag insertion is described in Section 7.2.3.2.1. The port acceptance rules and port based tag insertion are described in Section 7.2.3.2.2.



Section 7.2.3.3 describes the receive tag handling and Section 7.2.3.4 describes the software interface used to manipulate the tags.

7.2.3.2.1 Software Based Tag Insertion Rules

The software can require to send packets with different L2 tag, it may do so either directly through the data for all the tags or using the transmit descriptor as described in Section 8.4.2.1 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 2 variable bytes, then 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 7.2.3.2.2 are applied to decide if the packet can be sent.

7.2.3.2.1.1 Single Tag Handling

The following table describes the tag insertion in different cases when each tag is identified by a different Ethertype:

Table 7-16. 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: The table above relates to the configuration of a single tag. The configuration of different tags is independent.

7.2.3.2.1.2 Double Tag Handling

If two tags uses the same Ethertype (for example VLAN and double VLAN), there may be some cases where the hardware may not be able to identify which of the two tags were inserted by the driver.

In this case, if only one tag is seen in the packet and no insertion was requested by the driver, it is assumed to be the outer of the two tags (in our example, the outer VLAN). Any further action (anti spoof, UP translation, tag accept policy and port based tagging) will interpret this tag according to this (so, if anti spoof for example is applied only to inner VLAN, a packet with a single VLAN will be treated

as a packet with no VLAN for anti spoof). If a single tag is present in the buffer sent by the host, but the descriptor request insertion of one of the tags (for example an outer VLAN), then we assume 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 as described above, the handling of each tag is as described in Section 7.2.3.2.1.1.

7.2.3.2.2 Port (VSI) Based Tag Handling

7.2.3.2.2.1 Accept Rules

The previous section described how tags are inserted via the data buffer or the transmit descriptor. This section describes the rules applied per VSI to decide if the software request is acceptable. Table 7-17 summarizes the applied rule according to tag accept, tag insert and the tag presence in the packet for a specific tag.

Table 7-17. 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 ACCEPTUNTAGGED = 1	No	No	Yes	Send the packet as is
	No	Yes	Yes	Tag inserted by the VSI
	Yes	X	No	Drop packet
	VLAN ID = 0[2]	No	Yes	Send the packet as is
	VLAN ID = 0[2]	Yes	Yes	VSI's VLAN Tag overrides SW while keeping the SW priority. This mode is supported only if tag is inserted in the descriptor.
Allow tagged and untagged: VSI_TAR.ACCEPTTAGGED = 1 ACCEPTUNTAGGED = 1)	X	Yes	Unexpected VSI Setting	Undefined
	X	No	Yes	Send the packet as is
Allow tagged only: VSI_TAR.ACCEPTTAGGED = 1 ACCEPTUNTAGGED = 0	X	Yes	Unexpected VSI Setting	Undefined
	No	No	No	Drop Packet
	Yes	No	Yes	Send the packet as is

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

7.2.3.2.2.2 Port Based Tag Insertion Mechanism

After the packet sent by the software device driver was accepted, the XL710 may 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 8 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 7.2.3.2.1or as part of a port based tag insertion.

Each tag may 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* defines the location and length of the variable part. Figure 7-8 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 will use the value from the *VSI_TIR[0]* register, the second bit set, will use the value from the *VSI_TIR[1]* register, and the third bit set will use the value from the *VSI_TIR[2]* register.

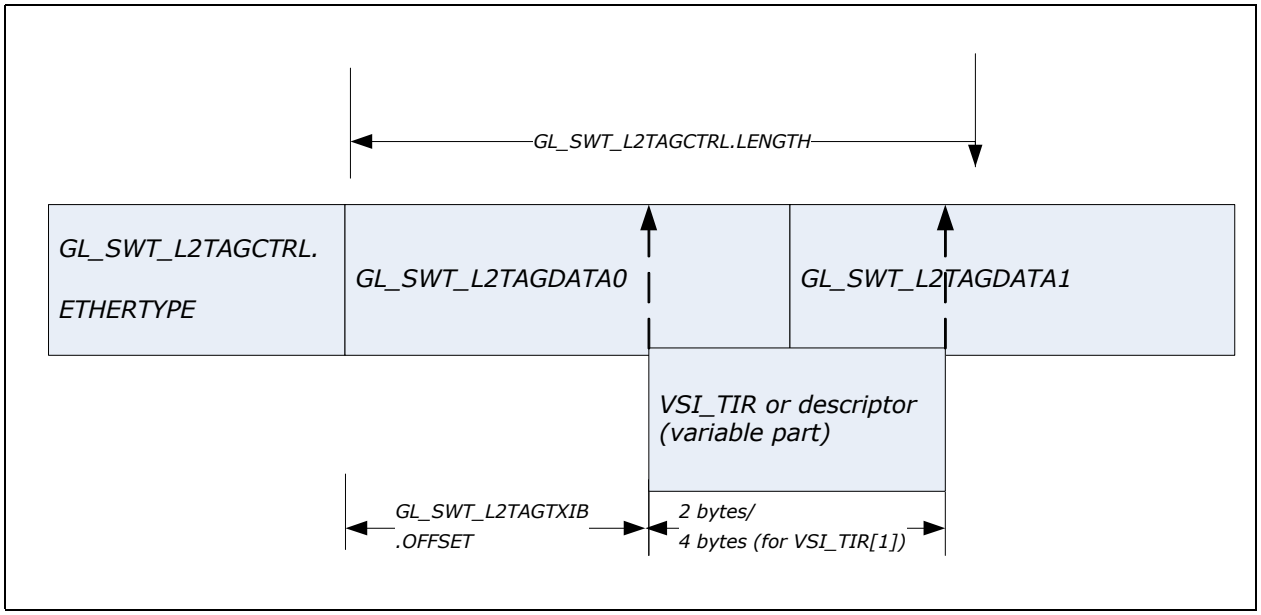


Figure 7-8. Tag insertion

7.2.3.2.2.3 Queue Based Tag Insertion

For one of the tags, the XL710 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.

This feature is used to allow different VLAN association for FCoE queues.

7.2.3.3 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.

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 3 16-bit words max). If more than one word needs to be extracted, then 32 bytes descriptors needs to be used (the *Dsize* flag in the receive queue context is set to 32B_Descriptor) otherwise 16 bytes 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* bit map and according to the L2TSEL field in the queue context. If L2TSEL bit is cleared, the first bit set will extract the value to the L2TAG1 field in the receive descriptor, the second bit set, will extract the value to the L2TAG2 (2nd) and L2TAG2 (1st) fields in the receive descriptor. If L2TSEL bit is set, the first bit set will extract the value to the L2TAG2 (2nd) and L2TAG2 (1st) field in the receive descriptor, the second bit set, will extract the value to the L2TAG1 field in the receive descriptor.

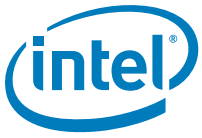
Note: The “L2TAG2 (2nd)” field is used only if the *GL_SWT_L2TAGRXEB.LENGTH* is 10b or 11b (24 or 32 bits extracted part)

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 8.3.2.2).

7.2.3.4 Tag Handling - Programming Interface

This section describes the CSR interface available to control the L2 tags handling of the XL710. 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 7.4.9.5.4.1) for the per VSI behavior.



XL710 support up to 8 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 7-18. L2 Tag control registers - global

Register	Field	Description	Register Description
GL_SWT_L2TAGCTRL [7:0]	ETHERTYPE	The Ethertype of the L2 tag	
	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, then ISVLAN should also be set. If set in multiple tags, then the Inner UP is taken from the first tag with this bit set in the packet.	
	OUTERUP	If this bit is set, then 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.	
	ENABLE	Is this Tag enabled	
	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	
GL_SWT_L2TAGTXIB [7:0]	OFFSET	Describes the offset in the header to which the variable data should be inserted (can be up to 8 bytes)	
GL_SWT_L2TAGDATA0[7:0], GL_SWT_L2TAGDATA1[7:0],	L2TAGDATA	The fixed part to insert in Transmit packets	

Each port defines which tags to expect in packets sent and received through this port using the *PRT_L2TAGSEN.ENABLE* field.

Each VSI defines which of the 8 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.

Note: In the receive descriptor, only 64 bits are allocated to the tag extraction data. So care should be taken not to enable extraction of receive tags that in total uses more than 64 bits to represent the variable part of the tags. So if a tag that has a variable part of 32 bits is used, one of the four available tags should not be used.

In the transmit descriptor, only 48 bits are allocated to the tag insertion data. So care should be taken not to enable transmit tags that in total uses more than 48 bits to represent the variable part of the tags.

Each VSI may be configured with a different behavior for each of the tags. The possible behaviors relates 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.

The following table describes the registers fields used to set the expected behavior for each tag.

Table 7-19. L2 Tag control registers - per VSI

Register	Field	Description	Register Description
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 7-20	
	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.	
	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]	
VSI_TSR	STRIPTAG[n] (n = 0..9)	Defines if the tag should be extracted from the packet.	
	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 2 of these bits should be set. If more than 2 bits are set, only the 2 first one 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. In order to allow insertion of priority bits from a descriptor and the VLAN tag value from the hardware as described in Table 7-16, 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 7.2.3.5.
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.

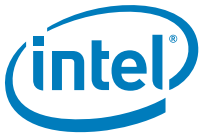
7.2.3.5 L2 Tags Configuration

This section describes the value to set in the different registers to implement the XL710 POR.

The following L2 tags are currently defined:

Table 7-20. L2 tags

Tag	Tag ID (in table)	
Reserved	0	
S-tag	1	
Outer VLAN	2	
VLAN	3	
Reserved	4-7	



The following table describes the configuration for each of the tags

Table 7-21. L2 headers support

Register/Tag	Reserved	S-tag	Outer VLAN	VLAN	Reserved	Reserved
Index	0	1	2	3	4	5-7
Global Configuration						
GL_SWT_L2TAGCTRL. ETHERTYPE		0x88A8	0x8100	0x8100		
GL_SWT_L2TAGCTRL. ISVLAN		0	0	1		
GL_SWT_L2TAGCTRL. INNERUP		0	0	1		
GL_SWT_L2TAGCTRL. OUTERUP		0/1[1]	0/1[2]	0		
GL_SWT_L2TAGCTRL. LONG		0	0	0		
GL_SWT_L2TAGCTRL. HAS_UP		1	1	1	0	
GL_SWT_L2TAGCTRL. LENGTH		2	2	2	6	
GL_SWT_L2TAGTXIB. OFFSET		0	0	0		
GL_SWT_L2TAGTXIB. LENGTH[3]		01b	01b	01b		
GL_SWT_L2TAGRXEB . OFFSET		0	0	0		
GL_SWT_L2TAGRXEB . LENGTH[3]		01b	01b	01b		
GL_SWT_L2TAGDATA 0, GL_SWT_L2TAGDATA 1	All zeros.					
Per Port Configuration Controlled by Firmware						
PRT_L2TAGSEN.ENAB LE		0/1[1]	0/1[2]	1	0	
Per VSI Configuration Controlled by *Add VSI* command						
VSI_TSR. STRIPTAG		0/1[4]	0/1[4]	0/1[5]		
VSI_TSR.SHOWTAG		0/1	0/1[7]	0/1[6]		
VSI_TSR.SHOWPRIO NLY		0	0	0/1[7]		
VSI_TAR.ACCEPTTAG GED		0/1	0/1	0/1[8]		
VSI_TAR.ACCEPTUNT AGGED		1	1	0/1[9]		

1. Set in modes where the outer tag is S-tag (when an *Add Port Virtualizer* command with an S-comp type is given).
2. Set by the *Set Port Parameters* command if double VLAN is enabled on any port.
3. 00b = 8 bits, 01b = 16bits, 10b = 24 bits, 11b = 32bits
4. Should be set.
5. Should be set if VSI is not VLAN aware or VM requested VLAN extraction offload.

6. Should be set if VSI is VLAN aware and VM requested VLAN extraction offload
7. Should be set if VSI in non VLAN aware but is DCB aware.
8. Should be set for VLAN aware VSIs.
9. Should be set if VSI must add VLAN.

7.2.4 VLAN handling

This section describes the handling of IEEE 802.1Q VLAN tags based on the features described above. Handling of other L2-tags as S-tag are described in the switching chapter (Section 7.4).

There can be up to two VLAN tags in a packet identified as inner VLAN (tag index 3) and outer VLAN (tag index 2).

Each port can be set to expect packets with outer VLAN using the *Set Port Parameters* admin command (Section 7.4.9.5.3.4). When enabled, a packet with a single VLAN will be 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 sections below describes the handling of the inner VLAN.

In MAC in UDP and MAC in GRE encapsulations, an internal VLAN may also be present as described in Figure 7-24. The handling of this VLAN is described in Section 7.2.4.3.

UP translation in inner and outer VLAN is described in Section 7.4.6.1.6.

7.2.4.1 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.

7.2.4.1.1 Tag insertion

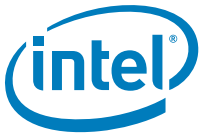
A VLAN tag can be inserted to the packets in three ways:

- 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 option are enabled if the VSI is allowed to add a VLAN tag by the *VLAN driver insertion mode* in the *Add VSI* command (Section 7.4.9.5.4.1). If a packet is sent with a VLAN tag from a VSI not allowed to add a tag, it will be 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 7.2.4.3.1 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 enable software to insert VLAN is not allowed (in this case, the VSI based VLAN tag overrides the software inserted VLAN, apart from the priority bits).

7.2.4.1.2 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 as described in Section 7.4.6.1.2.2 and dropped if the inserted VLAN is not in the list.

7.2.4.1.3 VLAN filtering

If the packet passed the previous stage, the VLAN tag is used as part of the forwarding process. It is compared to the MAC, VLAN filters added by the *Add MAC, VLAN pair* command (Section) 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 10.3 to define if it should be sent to the BMC.

7.2.4.2 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.

7.2.4.2.1 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 to the MAC, VLAN filters added by the *Add MAC, VLAN pair* command (Section) and compared to the VLAN egress filtering set by the *Add VLAN* admin command (Section 7.4.9.5.8.3) 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 10.3 to define if it should be sent to the BMC.

7.2.4.2.2 VLAN extraction

Before a packet is stored in the host memory the VLAN tag may 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 then it will show in the *L2TAG1* field and the *L2TAG1P* flag will be set.

Note: When a packet is sent to the BMC the VLAN is kept in the packet.

7.2.4.3 VLAN in tunnel packets

In MAC in UDP and MAC in GRE encapsulations, a VLAN within the tunneled MAC header may also be present as described in Figure 7-24.

7.2.4.3.1 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.

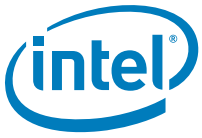
This mode is enabled irrespective of the VLAN configuration in the Add VSI command.

7.2.4.3.2 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 (1^{st}) field of the receive descriptor write-back. In this case, the second tag selected according to Section 7.2.3.3 is ignored. When the *SHOWIV* field is set, 32 bytes descriptors must be used (*DSize* field in he receive queue context must be set).

7.2.4.3.3 Tunneled VLAN in pass through traffic

Tunneled VLAN is not offloaded (not inserted nor extracted) for BMC pass through traffic.



7.2.5 S-tag handling

This section describes the handling of IEEE 802.1Qbg S-tags based on the features previously described.

There can be up to one S-tag in a packet (tag index 1). The sections that follow describe the handling of the S-tag.

7.2.5.1 Transmit flow

This section describes the handling of S-tag as part of the flow of packets sent by the host, the BMC or the EMP.

There is no anti spoofing capability for S-tags.

7.2.5.1.1 Tag insertion

An S-tag can be inserted to the packets in three ways:

1. As part of the packet buffer.
2. As part of the transmit descriptor in the L2TAG2 field if the IL2TAG2 field is set.
3. By the device from the VSI context.

The two first option are enabled if the VSI is allowed to add an S-tag by the clearing the S-tag insert enable field and setting the Accept tag from host field in the Add VSI command (Section 7.4.9.5.4.1). If a packet is sent with an S-tag from a VSI not allowed to add a tag, it will be dropped.

The third option is enabled by setting the S-tag insert enable in the same command. The tag to insert is defined in the S-tag field of the command.

Note: Setting both S-tag insert enable and accept tag from the host to enables software to insert VLAN is not allowed (in this case, the VSI based S tag overrides the software inserted S-tag, apart from the priority bits).

7.2.5.2 Receive flow

This section describes the handling of S-tag as part of the flow of packets received by the host, the BMC or the EMP.

7.2.5.2.1 S-tag extraction

Before a packet is stored in the host memory the S-tag can be stripped and optionally stored in the receive descriptor. The action done is defined per VSI in the S/E-tag extract mode field in the Add VSI command. The possible actions are:

- Show S-tag and UP in descriptor (legacy behavior) - used for cascaded port virtualizer with offload.
- Do nothing (leave S-tag in packet) - used for cascaded port virtualizer without offload

If the S-Tag is exposed in the descriptor then it shows in the L2TAG2 field and the L2TAG2P flag is set. In this case, 32-byte descriptors must be used (*DSize* field in the receive queue context must be set).

Note: If the SHOWIV field is set in the queue context, then the L2TAG2 field is allocated for the inner VLAN of tunnel packet. In this case, the S-tag extraction setting is ignored.

7.3 Frame Formats

This section describes the packet formats supported by the device. It is structured by layers (L2, L3, and L4), providing the details per each layer. Packets that do not conform with the described formats are handled as L2 packets.

- The following restrictions apply on the lengths of parsed packets: The device parses headers in the first 480B of the packet only. If the header (as defined in this section) extends beyond 510B, the packet is handled as an L2 packet with a packet type of "abort" (0xFF).
- Each single header is limited to 255B (unless restricted further in the text). If the header (as defined in this section) extends beyond 255B, the packet is handled as an L2 packet with a packet type of "abort" (0xFF).

7.3.1 L2 Packet Formats

IEEE 802.3 and 802.3 SNAP packets are considered and treated as Layer 2 packets. The XL710 does not identify such packets explicitly in any way.

7.3.1.1 Standard L2 Packet Format

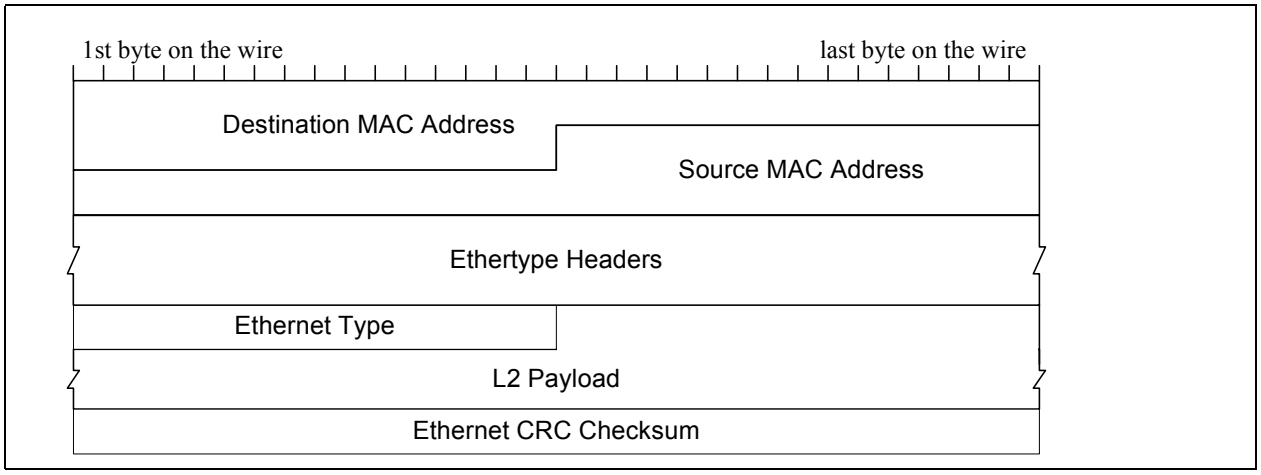


Figure 7-9. Standard L2 Frame format

7.3.1.2 Serial Ethertypes (L2-tags)

Table 7-22 lists the supported Ethertype headers. The order of Ethertype headers in the L2 header is per Figure .

Table 7-22. List of Serial Ethertype headers

Ethertype header	Value	Length[1]	Defined in
802.1BR E-tag	0x893F	8 words	IEEE 802.1BR clause 7
802.1Q S-tag	0x88A8	2 words	IEEE 802.1Q clause 9
Outer VLAN	Loaded from NVM (e.g. 0x8100, 0x9100, 0x9200)	2 words	IEEE 802.1Q clause 9
VLAN	0x8100	2 words	IEEE 802.1Q clause 9

1. Ethertype headers are an integer number of words. Length includes the Type field

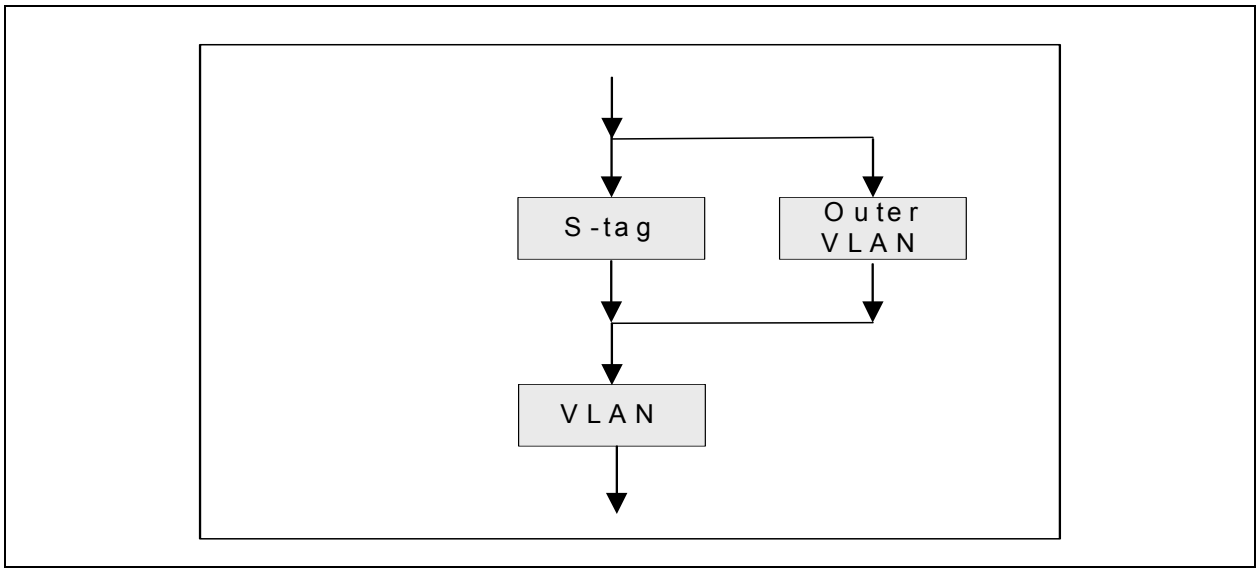


Figure 7-10. Order of L2 Ethertype headers

7.3.1.3 Last Ethertype (Upper Layer Protocol)

The upper-layer protocol (ULP) following the L2 header is identified by the Ethernet Type field (see Figure 7-9).

The following Table lists the Ethertypes supported by the device along with their index value. A value of 00-00 means the entry is reserved.

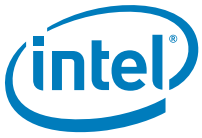


Table 7-23. List of Ethernet Types

Index	Value	Purpose	Format of ULP
0	88-F7	IEEE 1588 (IEEE 802.1AS)	See Section 8.5
1	89-14	FIP (FCoE Initialization Protocol)	See Section 7.3.5.2
2	00-00	Reserved	
3	00-00	Reserved	
4	88-CC	IEEE 802.1ab (LLDP)	IEEE P802.1AB spec
5	89-40	ECP	IEEE 802.1Qbg spec
6	00-00	Reserved	
7	00-00	Reserved	
8	88-8E	IEEE 802.1X (Network Access Control)	N/A
9	08-06	ARP	See Section 7.3.2.6
10	00-00	Reserved	
11	00-00	Reserved	
12	08-00	IPv4	See Section 7.3.2.1
13	86-DD	IPv6	See Section 7.3.2.2
14	89-06	FCoE	See Section 7.3.5
15	00-00	Reserved	

7.3.1.4 LLDP frame format

The general structure of an LLDP packet is depicted in Table 7-24.

Table 7-24. Structure of LLDP frame

LLDP Header	
LLDP Ethertype	LLDPDU

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 Table 7-25.

Table 7-25. Structure of LLDP PDU

Mandatory	Mandatory	Mandatory				Mandatory
Chassis ID TLV	Port ID TLV	Time to Live TLV	Optional TLV	...	Optional TLV	End of LLDPDU TLV

Each TLV has the structure depicted in Figure 7-26

Table 7-26. Structure of LLDP TLV

TLV Header		
7 bits	9 bits	0 <= n <= 511 octets
TLV type	TLV information string length	TLV information string

The basic format for Organizationally Specific TLVs is shown in Table 7-27

Table 7-27. Structure of LLDP Organizationally Specific TLV

TLV Header		TLV information string - 0 <= n <= 511 octets		
7 bits	9 bits	3 octets	1 octet	0 <= n <= 507 octets
TLV type	TLV information string length	Organizationally Unique Identifier (OUI)	Organizationally defined subtype	

The following table lists the applicable Organizationally Specific TLVs:

Table 7-28. List of Organizationally Specific TLVs

TLV	OUI value	Subtype value
DCBx	00-80-C2	09, 0A, 0B, 0C
EEE	00-12-0F	05

7.3.1.5 Other L2 packets

Magic packets are described in Section 5.4.1

Link control packets are described in Section 3.2.1.5.1. Such packets do not include any L2 tags other than those described in Section 3.2.1.5.1.

7.3.2 L3 Packet Formats

7.3.2.1 IPv4 Datagram

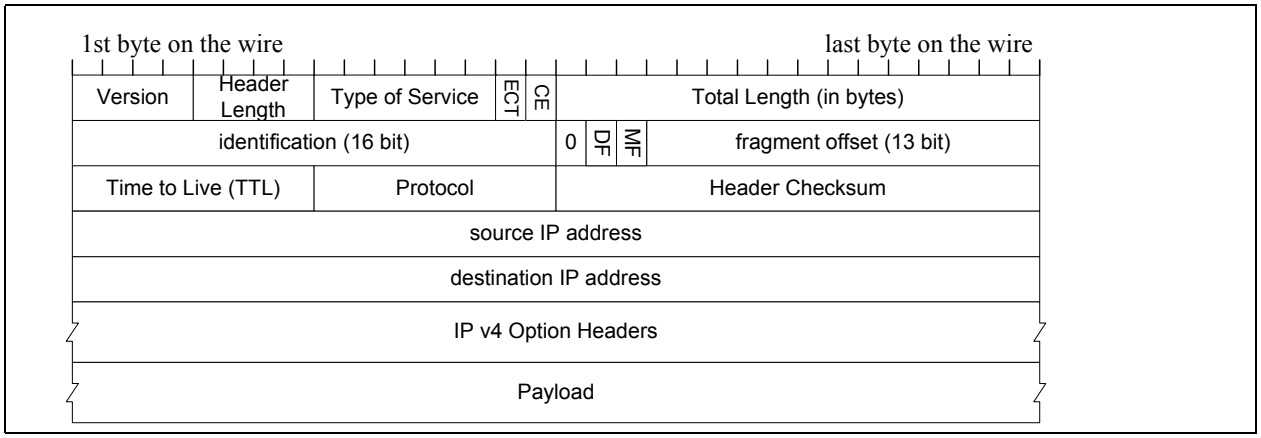


Figure 7-11. IPv4 Datagram

Version: The Version field is set to 0x4 for IPv4 header.

Header Length: The length of the IP header defined in DWord units.

Type of Service - TOS (6 bits): The Type Of Service field is used to indicate the quality of service with which this datagram is to be delivered by the internetwork routers.

IP Congestion Flags: CE, ECT.

Total Length: The Total Length is the size of the IP datagram (IP header and payload) in byte units.

Identification - ID: 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: Fragment offset, More Fragment(s) flag and Disable Fragmentation flag.

Time to Live - TTL: The TTL field indicates the number of links that this IP datagram can travel before an IP router discards it.

Protocol (Next Header): The Protocol field indicates the next protocol encapsulated within the IP layer. See Section 7.3.2.3 for the list those protocols that can be offloaded by the XL710.

Header Checksum: The 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.

7.3.2.2 IPv6 Datagram

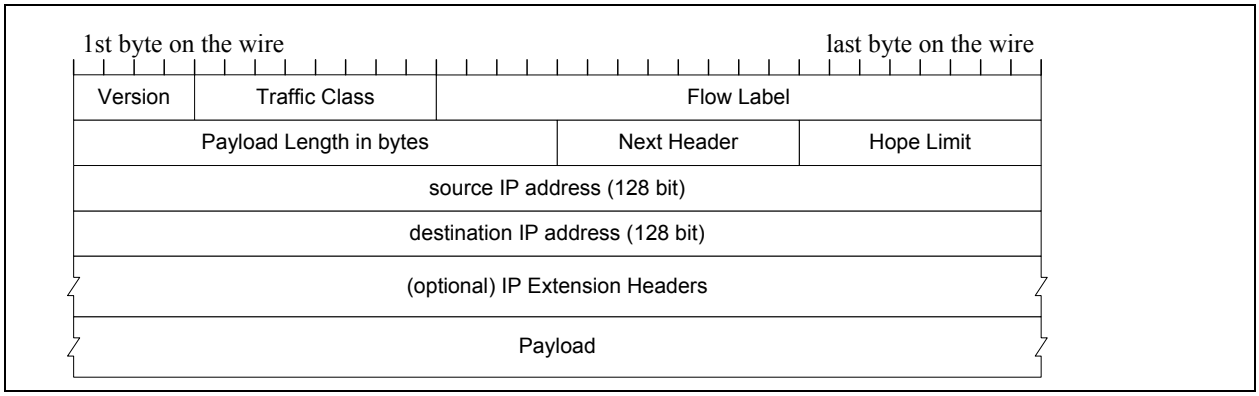


Figure 7-12. IPv6 Datagram

- Version: The Version field is set to 0x6 for IPv6 header.
- Traffic Class and Flow Label: The Traffic class and Flow Label are used for QOS support.
- Payload Length: The length of the IPv6 payload, i.e., 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): The Next Header field indicates the next protocol encapsulated within the IP layer. See Section 7.3.2.3 for the list those protocols that can be offloaded by the XL710.
- Hope Limit (TTL): The Hope 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.

7.3.2.2.1 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. The Table 7-29 below lists those most used IPv6 extension and their recommended ordering in the packet. Then these extension headers are illustrated in the following figures. The XL710 identifies these headers and skip them parsing the rest of the packet for its processing (unless specified differently in the table below).

Table 7-29. 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.
Destination Options	60	Variable length field defined in 8 byte units excluding the first 8 bytes. Can be located either at this location or before the mobility header.
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 XL710 does not continue to parse the rest of the packet.
Authentication header	51	Length Field is at Byte 1, define the header length minus 2 in 4 bytes units. When this header is found, The XL710 does not continue to parse the rest of the packet. Applicable for IPv4 as well.

Table 7-29. IPv6 Extension headers and their recommended ordering

Header (Protocol)	Next Header Value	Header Length and Header Length Field Offset
Encapsulating Security Payload	50	Length is 8 bytes plus variable length of Initial Value plus a trailer. When this header is found, The XL710 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. Can be located either at this location or before the routing header.
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

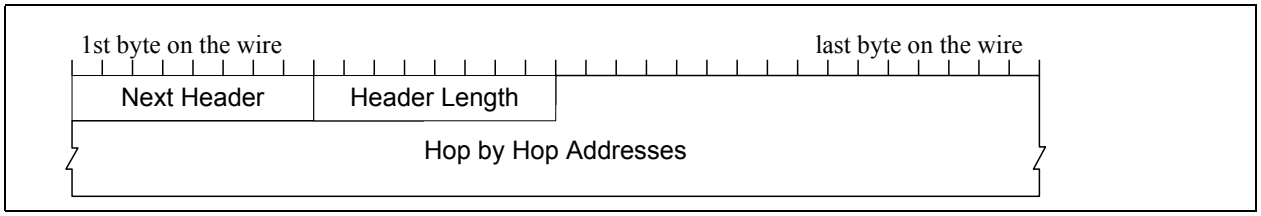


Figure 7-13. IPv6 Hop by Hop extension header

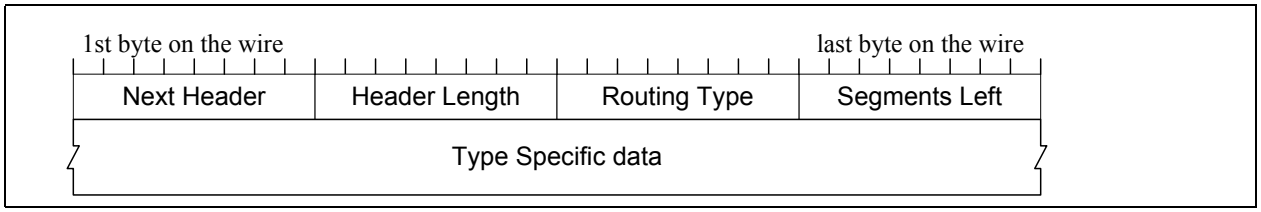


Figure 7-14. IPv6 Routing header

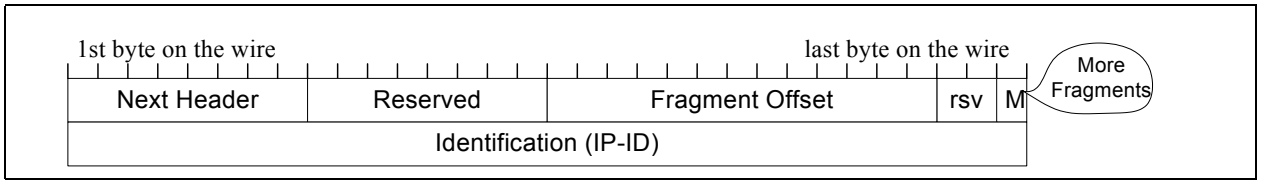


Figure 7-15. IPv6 Fragment header

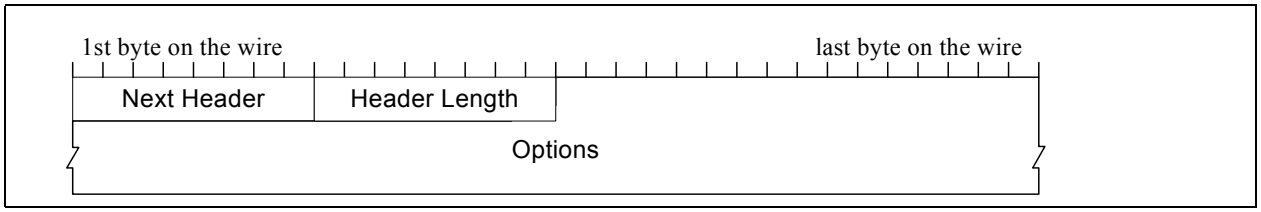


Figure 7-16. IPv6 Destination Options

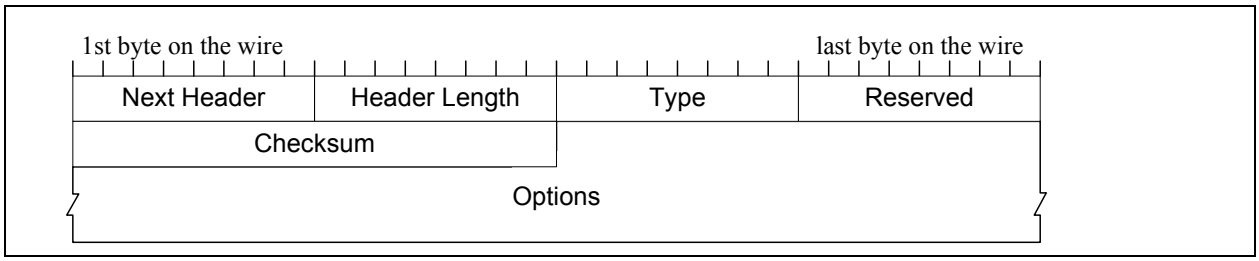


Figure 7-17. IPv6 Mobility Header

7.3.2.3 Protocol Headers (Next Header)

The Table 7-30 below lists the encoding of the Next Header Type field and information on determining each header type's length recognized by the XL710 according to the recommended ordering.

Table 7-30. 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 bytes units
IPv6	41	Length is always 40 bytes
Authentication	51	Length Field is at Byte 1, define the header length minus 2 in 4 bytes units
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 bytes 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
No Next Header	59	When no next header type is found, the rest of the packet is not processed

7.3.2.4 ICMP Datagram

ICMP packets are used as part of the manageability filtering.

Table 7-31. ICMP Packet Format

	msb..... . . . .lsb 31 24 First byte on the wire	msb 23	lsb 16	msb. lsb 15 8	msb.. ... . . lsb 7 0 Last byte on the wire
0	Type	Code		ICMP Header checksum	
4	Payload				

The following table describes the processing done to identify ICMP packets:

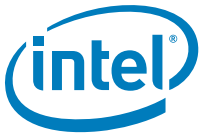


Table 7-32. IPv4 Packet Structure and Processing

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)	Possible S-TAG		Skip	
12 + T	S=(0/4)	Possible VLAN Tag		Compare	Processed by main address filter.
12 + T + S	2	Type	0x0800	Compare	IPv4
14 + T + S	1	Version/ HDR length	0x4X	Compare	Check IPv4
15 + T + S	1	Type of Service	-	Ignore	
16 + T + S	2	Packet Length	-	Ignore	
18 + T + S	2	Identification	-	Ignore	
20 + T + S	2	Fragment Info	-	Ignore	
22 + T + S	1	Time to live	-	Ignore	
23 + T + S	1	Protocol	0x1	Compare	ICMP
24 + T + S	2	Header Checksum	-	Ignore	
26 + T + S	4	Source IP Address	-	Ignore	
30 + T + S	4	Destination IP Address	-	Ignore	
34 + T + S	1	ICMP type	-	Ignore	
35 + T + S	1	ICMP Code	-	Ignore	
36 + T + S	2	ICMP Header Checksum	-	Ignore	
38 + T + S	F	ICMP Payload	-	Ignore	

The XL710 does not parse the ICMP header and only detects the IP Next protocol as ICMP.

7.3.2.5 ICMPv6 Datagram

ICMP packets are used as part of the manageability filtering and as part of proxying capabilities

Table 7-33. ICMPv6 Packet Format

	msb..... . . . .lsb 31 24 First byte on the wire	msb 23	lsb 16	msb. 15	lsb 8	msb.. 7	... . . lsb 0 Last byte on the wire
0	Type	Code		ICMPv6 Header checksum			
4	Payload						

The XL710 supports filtering of the following ICMPv6 packets.

Neighbor Discovery packets:

1. 0x86 (134d) - Router Advertisement.

2. 0x87 (135d) - Neighbor Solicitation.
3. 0x88 (136d) - Neighbor Advertisement.
4. 0x89 (137d) - Redirect.

MLD packets:

1. 0x82 (130d) - MLD Query
2. 0x83 (131d) - MLDv1 Report
3. 0x84 (132d) - MLD Done
4. 0x8F (143d) - MLDv2 Report

The following table describes the processing done to identify ICMPv6 packets:

Table 7-34. ICMPv6 Packet Structure and Processing

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 S-TAG		Skip	
12 + T	S=(0/4)	Possible VLAN Tag		Compare	Processed by main address filter.
12 + T + S +	2	Type	0x86DD	Compare	IPv6
14 + T + S	1	Version/ Priority	0x6X	Compare	Check IPv6
15 + T + S	3	Flow Label		Ignore	
18 + T + S	2	Payload Length		Ignore	
20 + T + S	1	Next Header	IPv6 next header types or 0x3A	Compare	0x3A - ICMPv6 header type
21 + T + S	1	Hop Limit	0x1	Ignore	
22 + T + S	16	Source IP Address		Ignore	
38 + T + S	16	Destination IP Address		Ignore	
54 + T+ S	N	Possible IPv6 Next Headers		Ignore	
ICMPv6 header					
54 + T + S + N	1	Type	0x82, 0x83, ox86, 0x87,0x88, 0x89 or 0x8F	Compare	Multicast Listener Discovery (MLD) or Neighbor Discovery types.
55 + T + S + N	1	Code	0x0	Ignore	
56 + T + S + N	2	Checksum		Check	
58 + T +S + N	F	Message Body		Ignore	

7.3.2.6 ARP packets

ARP packets are used as part of the manageability filtering.

The following Figure 7-18 and Table 7-35 describe the ARP packets format and the processing of these packets:

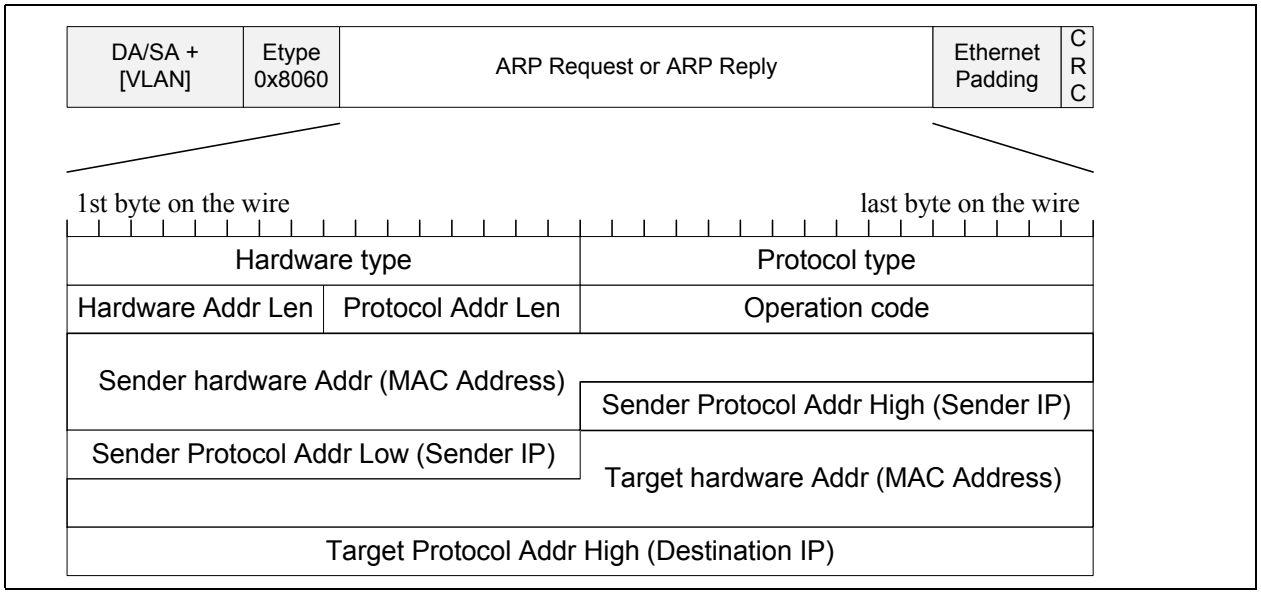


Figure 7-18. Arp packet format

Table 7-35. ARP Packet Structure and Processing

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 S-TAG		Skip	
12 + T	S=(0/4)	Possible VLAN Tag		Compare	Processed by main address filter.
12+ T + S	2	Type	0x0806	Compare	ARP
14 + T + S	2	Hardware Type	0x0001	Compare	Ethernet Hardware type
16 + T + S	2	Protocol Type	0x0800	Compare	IPv4 Protocol
18 + T + S	1	Hardware Size	0x06	Compare	MAC address size in bytes
19 + T + S	1	Protocol Address Length	0x04	Compare	IPv4 address size in bytes
20 + T + S	2	Operation	0x0001/ 0x0002	Compare	0x0001 = Request; 0x0002 = Response
22 + T + S	6	Sender HW Address	-	Ignore	
28+ T + S	4	Sender IP Address	-	Ignore	
32 + T + S	6	Target HW Address	-	Ignore	
38 + T + S	4	Target IP Address	IP4AT	Compare	Use to decide of an IP match of ARP packets.

7.3.3 L4 Packet Formats

7.3.3.1 UDP Datagram

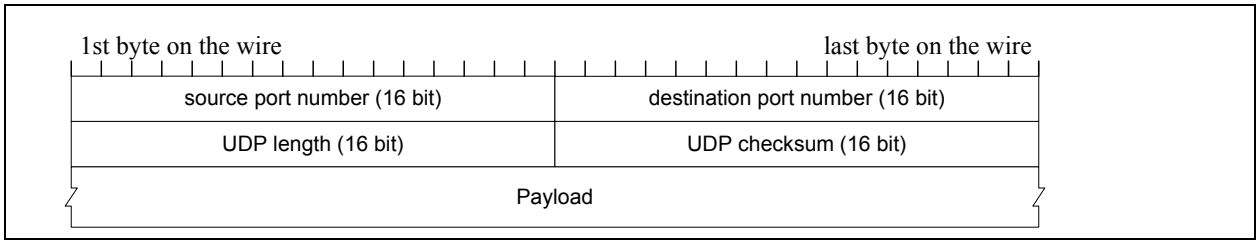


Figure 7-19. UDP Packet Format

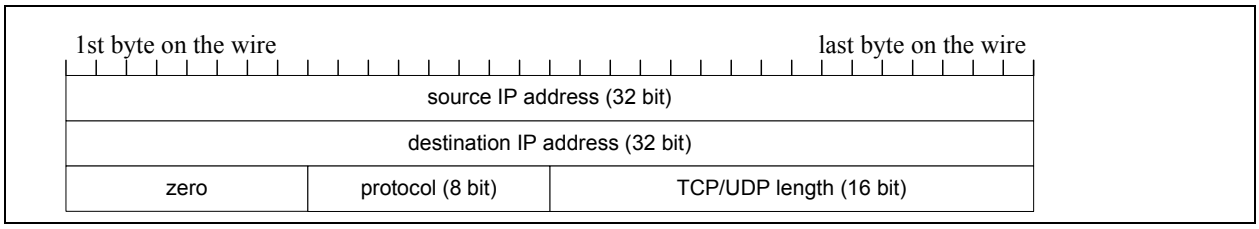


Figure 7-20. IPv4 Pseudo Header for TCP/UDP checksum

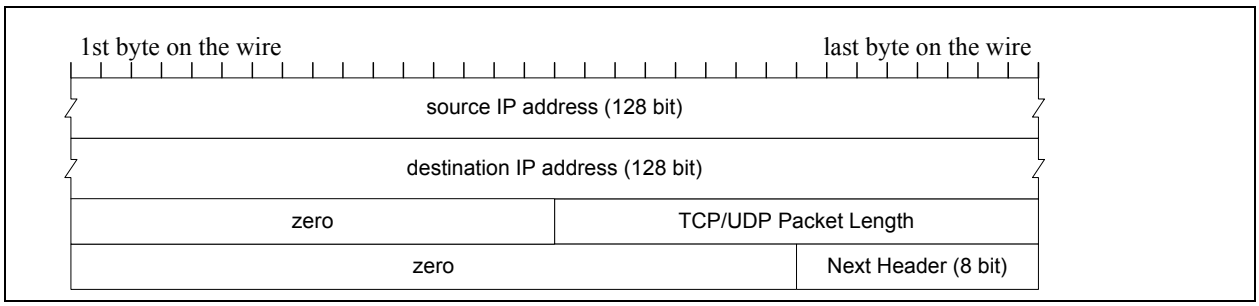


Figure 7-21. IPv6 Pseudo Header for TCP/UDP checksum

- Source and Destination Port Number: 16-bit port numbers
- UDP Length: The length of the entire UDP datagram, including both header and Data fields defined in byte units.
- UDP Checksum: 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 Figure 7-20 and Figure 7-21 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.

7.3.3.2 TCP Datagram

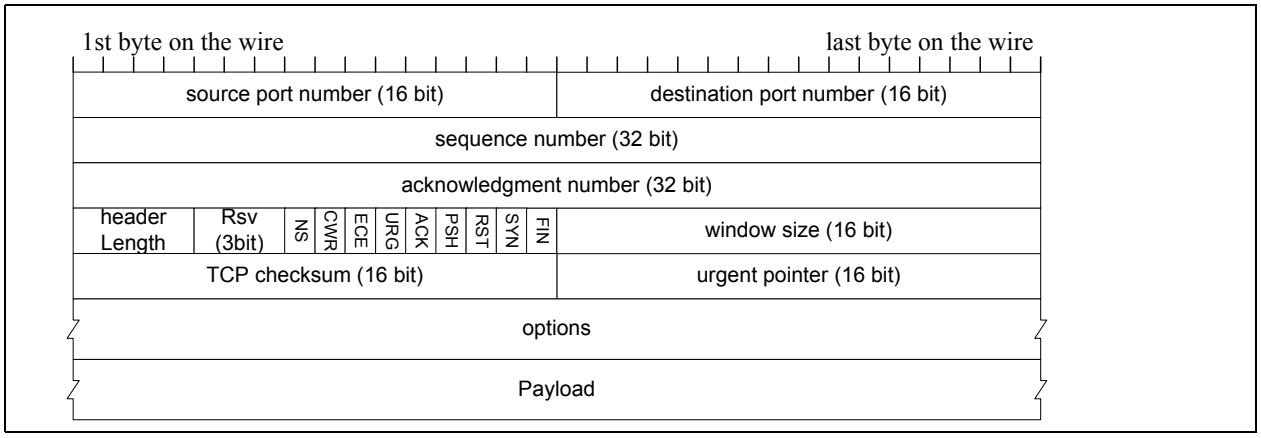


Figure 7-22. TCP Packet Format

- Source and Destination Port Number: 16-bit port numbers
- Sequence Number: 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.
- Acknowledge Sequence Number: 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: 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 (4bit): 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: 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: The checksum field a 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 Figure 7-20 and Figure 7-21 above. 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.

7.3.3.3 SCTP Datagram

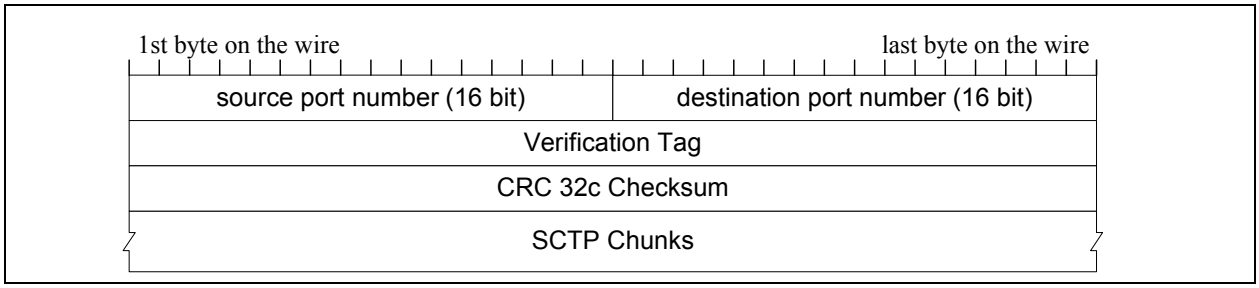


Figure 7-23. SCTP Packet Format

- Source and Destination Port Number: 16-bit port numbers
- Verification Tag: 32-bit random value selected by each endpoint in an association during setup. It is used to discriminates between two successive associations as well as 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 + x2 + x4 + x5 + x7 + x8 + x10 + x11 + x12 + x16 + x22 + x23 + x26 + x32. The CRC bytes are transmitted on the network in big endian ordering while the MS bytes is first on the wire.

7.3.4 Supported Tunneled Packet Formats

Figure 7-24 below illustrate the supported tunneled packet formats:

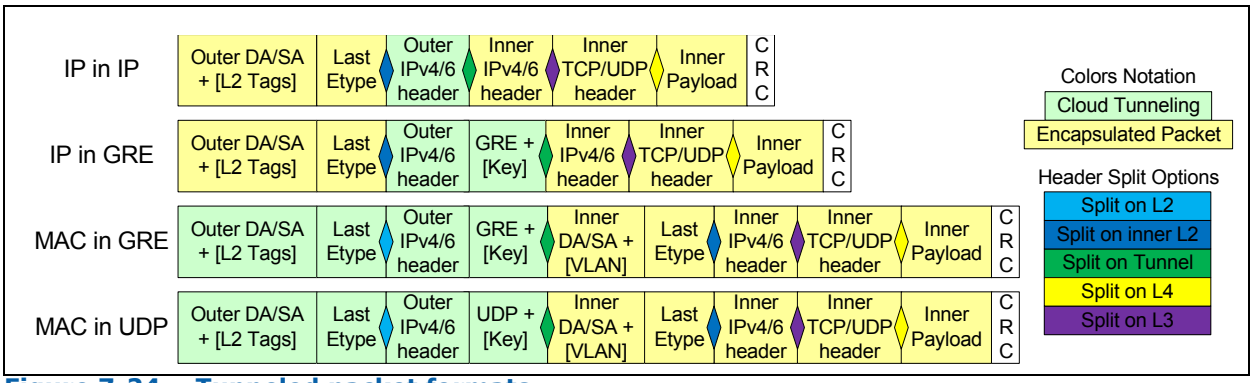


Figure 7-24. Tunneled packet formats

Processing of these packets is described in the LAN Engine chapter, Internal Switch chapter and Receive Classification Filters chapter.

7.3.4.1 IP and UDP Tunneling

The XL710 supports the following tunneled packets:

- IPv4 - IPv4
- IPv4 - IPv6
- IPv6 - IPv4
- IPv6 - IPv6
- IP - UDP teredo - IP// The IP can be IPv4 or IPv6 as described above

The IP headers may have options (IPV4) or extended headers (IPv6) described in Section 7.3.2.2.1.

UDP Teredo header is identified by its destination port number equals to one of 16 values as detailed in Section 7.1.6.1. The XL710 does not provide checksum offload for the Teredo header with no reporting.

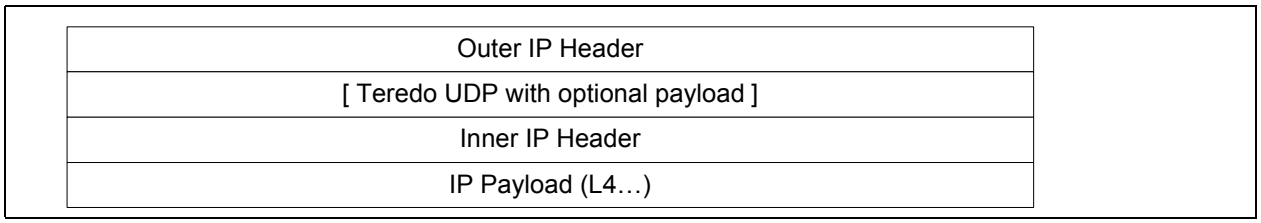


Figure 7-25. IP / Teredo Tunneling

7.3.4.2 GRE Header Version 0 (NVGRE)

1st byte on the wire								**last byte on the wire**
C	R	K	S	s	Recur	Flags	Version	Protocol Type
Checksum (Optional)								Offset (Optional)
Key (Optional) - Tenant Network ID (TNI)								Reserved
Sequence Number (Optional)								
Routing ::: (Optional)								

The GRE header is indicated by IP protocol equals to 47 (0x2F). The GRE headers supported by XL710 can be 1, 2, 3 or 4 Dwords depending on the flags in the first byte.

- 'C' (Checksum Present): 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): GRE header with routing header is not supported by XL710. If this flag is found active the header is not recognized by the device.
- 'K' (Key Present): If set then the Key field is present and contains valid information.
- 'S' (Sequence Number): If set then the Sequence Number field is present and contains valid information. XL710 ignores this flag other than an indication for the length of this header.

- 's' (Strict Source Route): It is recommended that this bit only be set if all of the Routing Information consists of Strict Source Routes. XL710 ignores this flag.
- 'Recur' (Recursion Control) - 3 bits: Contains the number of additional encapsulations which are permitted. XL710 supports only GRE header with no recursion headers (Recursion Control equals to zero).
- 'Version' - 3 bits: GRE protocol version. Zero value identifies a GRE header version zero. XL710 supports only zero value. (Note that version 1 is used for PPP protocol)
- 'Protocol' - 16 bits: Contains the protocol type of the payload packet. In general, the value will be the Ethernet protocol type field for the packet. The following values are supported by XL710. Packet parsing that might have other protocol values is undefined.
  - 0x0800 ..... IPv4 header (indicates an IPv4 in GRE packet format)
  - 0x86DD ..... IPv6 header (indicates an IPv6 in GRE packet format)
  - 0x6558 ..... MAC header (indicates a MAC in GRE packet format)
- 'Checksum' - 16 bits: Contains the IP (one's complement) checksum of the GRE header and the payload packet. This field is not processed by XL710.
- '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 XL710.
- 'Key' - 24 bits: Contains a number which was inserted by the encapsulator. It may 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 which is inserted by the encapsulator. It may 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 XL710.
- 'Routing' - Variable length. This field is a list of SREs and is not supported by XL710.

7.3.4.3 UDP plus its Optional Private Header

UDP Header	Optional UDP Private Header

- UDP Header - The UDP Destination Port number equals to one of 16 Teredo port numbers (also used for MAC in UDP tunneling)
- Private Header - The private UDP header includes any optional private content that could include a networking key. the length of the private header and the optional networking key are recognized by the device according to the UDP port number as programmed by the “Add Teredo Port” admin command described in Section 7.1.6.1.

7.3.4.3.1 VXLAN Header in UDP format

VXLAN draft provides the structure of the optional “private header” described above used for VXLAN encapsulation as shown in Figure 7-26 and Table 7-36.

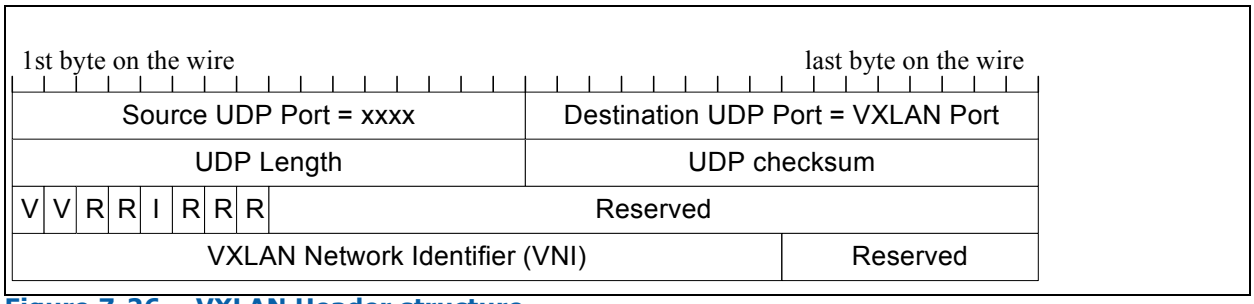


Figure 7-26. VXLAN Header structure

Table 7-36. VXLAN Header structure

Offset (from the UDP header)	Size [byte]	Field	Value	Action	Comment
0	2	Source UDP Port	XXXX	Ignore	Source UDP port number could be any value
2	2	Destination UDP Port	VXLAN Port	Compare	The reserved VXLAN port number is programmed by the Add "Tunneling UDP" admin command.
4	2	UDP Length	XXXX	Ignore	Length of the entire datagram including the UDP header
6	2	UDP checksum	XXXX	Ignore	Provide an indication if the checksum equals to zero (i.e., no checksum)
8	1	Flags	0x08	Ignore	VV = version (expect zero) I = VNI valid indication
9	3	Reserved	XXXX	Ignore	Reserved
12	3	VXLAN Network Identifier	XXXX	Compare	Unique value per tenant
15	1	Reserved	XX	Ignore	Reserved

7.3.4.3.2 Geneve Header in UDP format

Geneve draft (Geneve: Generic Network Virtualization Encapsulation draft-gross-geneve-00) provides the structure of the optional "private header" described above used for Geneve encapsulation as shown in Table 7-37 and Table 7-38.

Table 7-37. Geneve Header structure

1st byte on the wire					last byte on the wire
Source UDP Port = xxxx					Destination UDP Port = 0x6081
UDP Length					UDP Checksum
V	Opt Len	O	C	Reserved	Next Protocol
Virtual Network Identifier (VNI)					Reserved
Variable Length Options (defined by "Opt Len")					

Table 7-38. Geneve Header structure

Offset (from the UDP header)	Size [byte]	Field	Value	Action	Comment
0	2	Source UDP Port	XXXX	Ignore	Source UDP port number could be any value
2	2	Destination UDP Port	0x6081	Compare	The reserved Geneve port number is programmed by the Add "Tunneling UDP" admin command.
4	2	UDP Length	XXXX	Ignore	Length of the entire datagram including the UDP header
6	2	UDP checksum	XXXX	Ignore	Provide an indication if the checksum equals to zero (i.e., no checksum)
8.0 9.0	2 bits 1 bit	V V (Version) O (OAM frame)	00b 0b	Check	If not zero then packet is posted to Rx queue zero of the default VSI of the LAN port (not identified as ~~NGE~~a Geneve packet)
8.2	6 bit	Opt Len (Option Length)	Variable	Check	Defines the length of the options fields in 4 byte units. XL710 supports only 5 bits length. If the MS bit of the field is set, the packet is posted to Rx queue zero of the default VSI of the LAN port (not identified as a Geneve packet)
9.1	1 bit	C (Critical Options Present)	X	Ignore	Expected to be processed by the software stack
9.2	6 bit	Reserved	X	Ignore	
10	2	Next Protocol		Check	Next protocol that follows the Geneve header. XL710 supports the following values: 0x6558 MAC header (MAC in Geneve) 0x0800 IPv4 header (IPv4 in Geneve) 0x86DD IPv6 header (IPv6 in Geneve) If other values are found, the packet is identified as "Geneve, Other"
12	3	VNI (Network Identifier)	Variable	Compare	Unique value per tenant
15	1	Reserved	X	Ignore	
16	4 x "Opt Len"	Variable Length Options	XXXX	Ignore	The length of the Options field is defined by the "Opt Len" field in this header

7.3.5 FCoE packet formats

Related Standards

- FC-FS-2: FRAMING AND SIGNALING-2 (FC-FS-2) Rev 1.00
- FCoE: Fibre Channel over Ethernet Draft Presented at the T11 on May 2007

FC over Ethernet packets encapsulate FC frames as shown in Table 7-39 and Table 7-40.

Table 7-39. Ethernet Encapsulation to FC frames

Ethernet MAC Addresses	VLAN Tag	FCoE Header	FC Frame					EOF	Ethernet CRC
			[FC Extended Header]	FC Header (24 bytes)	[FC Optional Header]	Optional FC Data 0 to 2112 bytes)	FC CRC		

Table 7-40. Detailed FCoE Packet Structure without Extended and Optional headers

	msb..... . . . .lsb 31 24 First byte on the wire	msb 23	lsb 16	msb. 15	. 12	 11	lsb 8	msb.. 7	... . . lsb 0 Last byte on the wire
0	Destination Ethernet MAC Address								
4									
8	Source Ethernet MAC Address								
12	Optional Ethertype Headers (T bytes while T equals 4 or 8)								
12+T	802.1Q Tag (VLAN + Priority)								
16+T	FCoE Ethernet Type = 0x8906			Ver			Reserved		
20+T	Reserved								
24+T	Reserved								
28+T	Reserved							SOF	
32+T	Optional Extended VFT Header (S bytes while S equals to 8)								
32+T+S	Routing Control (R_CTL)		Destination Identification (D_ID)						
36+T+S	Class Specific Control (CS_CTL)		Source Identification (S_ID)						
40+T+S	TYPE		Frame Control (F_CTL)						
44+T+S	Sequence ID (SEQ_ID)		Data Field Control (DF_CTL)		Sequence Count (SEQ_CNT)				
48+T+S	Originator Exchange ID (OX_ID)				Responder Exchange ID (RX_ID)				
52+T+S	Parameter (PARAM)								
0…N	Optional FC Data… always 4 byte align (including optional FC padding)								
56+N+T+S	Fibre Channel CRC (FC_CRC)								
60+N+T+S	EOF		Reserved						
64+N+T+S	Ethernet CRC								

7.3.5.1 Hardware Operation on the FCoE packet fields

7.3.5.1.1 Packet Field Processing at Packet Reception

7.3.5.2 FCoE Encapsulation - Header and Trailer

Table 7-41. FCoE Header checks

Field	Hardware checks at packet reception
Destination Ethernet MAC Address	Filtering to VSI
Source Ethernet MAC Address	FCoE context match parameter
802.1Q Tag (VLAN + Priority)	VLAN ID is FCoE context match parameter VLAN Priority selects a traffic class
FCoE Ethernet Type = 0x8906	FCoE packet identification
Version	FCoE packet identification. Version larger than FCOEVER field in the GLFCOE_FCTL register is not recognized as FCoE packet.
SOF & EOF	checked by DDP / DWO context rules
Optional Extended VFT Header	Skip header
Routing Control (R_CTL)	Identify FCoE header type: VFT/FC headers; Data (0x01), RSP (0x07), RDY (0x05) or Other
Destination Identification (D_ID)	FCoE context match parameter
Class Specific Control (CS_CTL)	Skip
Source Identification (S_ID)	Skip
TYPE	FCoE packet identification. Value other than 0x08 is reported as “FCoE protocol Error” in the receive descriptor
Frame Control (F_CTL)	checked by DDP / DWO context rules and Hash filter. See complete description in Figure 7-45
Sequence ID (SEQ_ID)	checked by DDP / DWO context rules
Data Field Control (DF_CTL)	must be zero for DDP / DWO
Sequence Count (SEQ_CNT)	checked by DDP / DWO context rules (in order reception)
Originator Exchange ID (OX_ID)	FCoE context match parameter and hash filter
Responder Exchange ID (RX_ID)	FCoE context match parameter and hash filter
Parameter (PARAM)	checked by DDP / DWO context rules (in order reception)
Fibre Channel CRC (FC_CRC)	Integrity check

The FCoE header is composed of a new FCoE Ethernet type, FCoE version tag and the Start Of Frame tag (SOF).

- FCoE Ethernet Type: Equals 0x8906
- Version (Ver): A 4 bit field that indicates the FCoE protocol version number. The XL710 supports FCoE version as defined by FCOEVER field in the FCRXCTRL register.
- Start of Frame (SOF): The FCoE Start of frame is a subset of the FC-FS-2 SOF codes as defined in the Table 7-42 below:

Table 7-42. E_SOF Mapping

FC SOF	FCoE SOF Code	FC Traffic Class	Description
SOFf	0x28	F	Fabric start of frame. Not expected in an FCoE.
SOFi2	0x2D	2	used in the first frames in a sequence

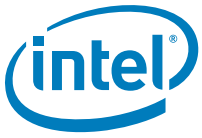


Table 7-42. E_SOF Mapping

FC SOF	FCoE SOF Code	FC Traffic Class	Description
SOFn2	0x35	2	used in all but first frame in a sequence
SOFi3	0x2E	3	used in the first frames in a sequence
SOFn3	0x36	3	used in all but first frame in a sequence

- End of Frame (EOF): The FCoE End of frame maps the FC-FS-2 EOF codes as defined in the Table 7-43 below:

Table 7-43. EOF Mapping

FC EOF	FCoE EOF Code	FC Traffic Class	Comment
EOFn	0x41	2, 3, 4, F	normal EOF
EOFt	0x42	2, 3, 4, F	EOF Terminate used to close a sequence
EOFni	0x49	2, 3, 4, F	EOF Invalid indicating that the frame content is invalid.
EOFa	0x50	2, 3, 4, F	EOF Abort

7.3.5.3 FC Frame Format

The FC frame as defined in FC-FS-2 specification is shown in the Table 7-44 below while relevant fields for the hardware are detailed in this section.

Table 7-44. FC Frame Format

SOF	Extended Header	FC Header	Optional Headers	FC Payload: FC Data & optional padding (0 to 2112 bytes)	FC CRC	EOF

7.3.5.3.1 FC SOF and EOF

- FC SOF and EOF: FC Start of frame (SOF) delimiter and End of frame (EOF) delimiter. In FCoE frames, the SOF and EOF fields in the FC frame are extracted and reflected in the FCoE encapsulation. The SOF and EOF codes that are reflected in the FCoE framing are shown in Table • and Table 7-43.
- FC CRC: The Cyclic Redundancy Check (CRC) is a four byte field that follows the Data Field. It enables end to end integrity checking on the whole FC frame. The CRC polynomial is the same one as the one used in FDDI and Ethernet while the CRC bytes are transmitted in reversed byte ordering relative to Ethernet: X32+X26+X23+X22+X16+X12+X11+X10+X8+X7+X5+X4+X2+X+1. The HW offloads the FC CRC calculation as described in the FCoE Engine chapter - Section 9.0.

7.3.5.3.2 FC Header

The FC header includes fields that are modified by the HW as part of transmit offloads: TSO and ETSO. These fields are indicated in this section as “Dynamic” while fields that are not modified by the HW are indicated as “Static”.

- Routing Control (R_CTL): The R_CTL is a one-byte Static field during TSO that contains routing and information bits to categorize the frame function. During ETSO the R_CTL is set to a different value for the RSP packet (see RSP packet format in this chapter).
- Class Specific Control (CS_CTL): This is a 1 byte Static field that defines either the Class specific control or priority according to bit 17 in the Frame control (F_CTL) field.
- Destination Identification (D_ID): The D_ID is a 3 byte Static field that defines the FC destination address.
- Source Identification (S_ID): The S_ID is a 3 byte Static field that defines the FC source address.
- Data Structure Type (TYPE): The TYPE is a 1 byte Static field that identifies the protocol of the frame content for data frames. When the Routing bits in R_CTL indicate FC-4 Device_Data or FC-4 Link_Data the TYPE field is expected to be 0x08 for Fibre Channel Protocol.
- Data Field Control (DF_CTL): The DF_CTL is a 1 byte Static field that specifies the presence of optional headers at the beginning of the Data_Field. The XL710 does not support TSO, ETSO, DDP and DWO offloads for packets with optional headers. Software should not use these offloads in case optional headers are expected.
- Sequence ID (SEQ_ID): During TSO the SEQ_ID is a 1 byte Static number associated with a sequence. A sender must assign SEQ_ID numbers so that the recipient would always be able to distinguish between consecutive sequences. SEQ_ID do not have to be sequential and do not have to be unique even within the same IO exchange as long as it is guaranteed that the recipient would be able to distinguish between them. In ETSO the SEQ_ID is Dynamic number while it is incremented by one on the RSP packet.
- Sequence Count (SEQ_CNT): The SEQ_CNT is a 2 byte Dynamic field that indicates the sequential order of Data frame transmission within a single Sequence or multiple consecutive Sequences for the same Exchange. The SEQ_CNT of the first Data frame of the first Sequence of the Exchange transmitted by either the Originator or Responder is '0'. The SEQ_CNT of subsequent Data frames in the Sequence is incremented by one for each data frame. The SEQ_CNT of the first Data frame in each sequence other than the first one can start at '0' or be incremented by 1 from the last used SEQ_CNT.
- Originator Exchange ID (OX_ID): The OX_ID is a two-byte Static field that identifies the Exchange_ID assigned by the Originator. If the Originator is enforcing uniqueness via the OX_ID mechanism, it shall set a unique value for OX_ID other than FFFFh. A value of FFFFh indicates that the OX_ID is unassigned and that the Originator is not enforcing uniqueness via the OX_ID. The XL710 supports large receive and direct data placement only if OX_ID is used to identify uniqueness.
- Responder Exchange ID (RX_ID): The RX_ID is a two byte Static field assigned by the Responder that shall provide a unique, locally meaningful exchange identifier at the Responder. The Responder of the Exchange shall set a unique value for RX_ID other than FFFFh.
- Parameter (PARAM): The Parameter field is a Dynamic field which is based on frame type. For Data frames with the relative offset present bit set to 1, the Parameter field specifies relative offset. The offset defines the relative displacement of the first byte of the Payload of the frame from the base address as specified by the ULP. Relative offset is expressed in terms of bytes.
- Frame Control (F_CTL): The F_CTL is a 3 byte Dynamic field that contains control information relating to the frame content. The F_CTL field is further detailed in the Table 7-45 below. Some of the F_CTL fields are meaningful only in some conditions (detailed in the table). If these conditions are not met then the value of these fields should be ignored.

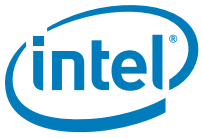


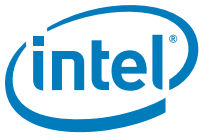
Table 7-45. F_CTL field

Control Field	bit	Description
Exchange Context	23	0 = Originator of Exchange; 1 = Responder of Exchange. Receive handling: Checks FCoE context match by the OX_ID when "Exchange Context" equals to '1' and RX_ID when "Exchange Context" equals to '0'. TSO, ETSO, WDO transmit handling: Taken from the THeader
Sequence Context	22	0 = Sequence Initiator; 1 = Sequence Recipient. Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Taken from the THeader
First Sequence	21	0 = Sequence other than first of Exchange; 1 = first Sequence of Exchange. Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Always zero
Last Sequence	20	0 = Sequence other than last of Exchange 1 = last Sequence of Exchange Must be set on the last data frame of the last sequence. However it can be set on any preceding frames. Once it is set, it must be set on all frames till the last one of that exchange. The hardware validate that this flag is active only in RSP packet. TSO, ETSO, WDO transmit handling: Always zero other than RSP packet in ETSO
End Sequence	19	0 = Data frame other than last of Sequence; 1 = last Data frame of Sequence The hardware identifies the last packet in a sequence by this flag. The EOF is checked to be EOFt according to the "End Sequence" flag. TSO = Zero in all packets but the last one on which it is taken from the THeader ETSO = Zero in all packets but the last data packet and the RSP on which it is set to one WDO = Zero in all packets but the last packet on each sequence on which it is set to one
End Connection	18	Relevant for Class 1 or 6 Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Always zero
CS CTL / Priority Enable	17	0 = Word 1, Bits 31-24 = CS_CTL; 1 = Word 1, Bits 31-24 = Priority Defines the meaning of the CS_CTL byte in the FC header to be either CS_CTL or Priority indication. Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Taken from the THeader
Sequence Initiative	16	0 = hold Sequence initiative; 1 = transfer Sequence initiative This bit is used to transfer initiative from a sender to a recipient. This bit is meaningful only on the last frame of a sequence (when bit 19 - "End Sequence" is set). The hardware expect that this flag is set in RSP packet. It can also be set on the last packet in a sequence of DDP context. In this case the context is invalidated and the packet header is posted to the LAN queue. TSO = Zero in all packets but the last one on which it is taken from the THeader ETSO = Zero in all packets but the RSP on which it is set to one WDO = Zero in all packets but the last packet on each sequence on which it is set to one
X_ID reassigned	15	Obsolete. Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Taken from the THeader
Invalidate X_ID	14	Obsolete. Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Taken from the THeader
ACK Form	13:12	00b = No assistance provided ; 10b = reserved 01b = Ack_1 Required ; 11b = Ack_0 Required ACK Form is meaningful on all Class 1, Class 2, or Class 6 Data frames of a Sequence and on all connect request frames. ACK_Form is not meaningful on Class 1, Class 2, or Class 6 Link_Control frames, or any Class 3 frames. Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Taken from the THeader

Table 7-45. F_CTL field

Control Field	bit	Description
Data Compression	11	Obsolete. Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Taken from the THeader
Data Encryption	10	Obsolete. Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Taken from the THeader
Retransmitted Sequence	9	0 = Original Sequence transmission; 1 = Sequence retransmission Meaningful in Class 1 and 6 and only. The hardware enables processing by the DDP and DWO contexts only if the flag is cleared. TSO, ETSO, WDO transmit handling: Taken from the THeader
Unidirectional Transmit	8	Relevant for Class 1 Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Taken from the THeader
Continue Sequence Condition	7:6	Last Data frame - Sequence Initiator 00b = No information; 01b = Sequence to follow-immediately 10b = Sequence to follow-soon; 11b = Sequence to follow-delayed It is meaningful only when the "End Sequence" is set to '1' and the "Sequence Initiative" bit is cleared '0'. Receive handling: This flag is ignored. TSO = Zero in all packets but the last one on which it is taken from the THeader ETSO and DWO = Zero in all packets
Abort Sequence Condition	5:4	ACK frame - Sequence Recipient 00b = Continue sequence; 01b = Abort Sequence, Perform ABTS 10b = Stop Sequence; 11b = Immediate Sequence re-XMT requested Data frame (1st of Exchange) - The Abort Sequence shall be set to a value by the Sequence Initiator on the first Data frame of an Exchange to indicate that the Originator is requiring a specific error policy for the Exchange. 00b = Abort, Discard multiple Sequences; 01b = Abort, Discard a single Sequence 10b = Process policy with infinite buffers; 11b = Discard multiple Seq with immediate re-XMT Receive handling: DDP/DWO context is invalidated and packet header is posted to the LAN queue if this field s non-zero. TSO = Zero in all packets but the last one on which it is taken from the THeader. The "Abort Sequence Condition" might be set to non-zero value in TSO only if the software uses TSO also for a single packet transmission. ETSO and DWO = Zero in all packets.
Relative offset present	3	0 = Parameter field defined for some frames; 1 = Parameter Field = relative offset Receive handling: Relates to the PARAM field only if this flag is set. TSO, ETSO, WDO transmit handling: Taken from the THeader
Exchange reassembly	2	Reserved for Exchange reassembly Receive handling: This flag is ignored. TSO, ETSO, WDO transmit handling: Taken from the THeader
Fill Bytes	1:0	Defines the number of FC padding bytes to fill in the last frame in the sequence. The value of these bytes is 0x00. When FCoE offload is enabled (TUCMD.FCoE bit is set in the transmit context descriptor), the hardware can pad the FC payload according to the buffer size indicated by software (by the PAYLEN field in the transmit data descriptor). Receive handling: If this field is non zero, the packet header is posted to the LAN queue. TSO = DWO = Zero in all packets but the last one on which it is taken from the THeader. ETSO =Zero in all packets but the last data packet on which it is taken from the THeader.

7.3.5.3.3 FC Extended Headers



The diagrams in Table 7-46, Table 7-48 and Table 7-49 show the Fibre Channel frame structure with Extended headers and lists the existing headers. Extended headers are identified by the R_CTL value as shown in the diagrams below. In FCoE traffic only the VFT header is expected. The XL710 does not relate to the content of the extended headers other than skipping it in order to parse the rest of the packet.

Table 7-46. FC Frame format with Extended Headers

FC Frame				
Extended Headers [Optional]	FC Header	Optional FC Header	[optional] FC Data (including optional padding)	FC CRC

Table 7-47. Extended Header Format

	31.....24 First bytes on the wire	23.....16	15.....8	7.....0 Last bytes on the wire
0	R_CTL	Extended Header Specific Fields		
4...				

Table 7-48. FC Extended Headers

R_CTL	Extended Header	Length	Supported Offloads
0x50	VFT_Header (Virtual Fabric Tagging Header)	8 Bytes	All offloads are supported
0x51	IFR_Header (Inter-Fabric Routing Header)	8 Bytes	FC CRC integrity, EOF and SOF tags offload
0x52	Enc_Header (Encapsulation Header)	24 Bytes	
0x53...0x5F	Reserved	-	

Table 7-49. VFT Extended Header Format

Bits Word	First byte on the wire															Last byte on the wire		
	3				2	2	2	2	1	1	1	1	1	1				
	1				4	3	2	1	8	7	6	5	3	2		7	1	0
0	R_CTL				Ver		Type		R		Priority		VF_ID					R
1	HopCt				R = Reserved													

7.3.5.3.4 FC Optional Headers

FCoE frames may include FC Optional headers. These headers (if they exist) would always show in the first frame in a sequence. While FC implementation may include the FC Optional headers only on the first frame in a sequence. The XL710 does not support TSO, ETSO, DDP and DWO offloads for packets with optional headers. Software should not use these offloads in case optional headers are expected. The Table 7-50 lists the known FC Optional headers.

Table 7-50. FC Optional Headers

DF_CTL	Optional Header	Length	Appears in FC packets
bit 6	ESP Header / ESP Trailer	Variable	All frames of the exchange
bit 5	Network Header	16 Bytes	first Data frame of a Sequence
bit 4	Association Header (obsolete)	32 Byte	first Data frame of a Sequence
bits 1:0	Device Header	0, 16, 32, or 64 Bytes	first Data frame or in all Data frames of a Sequence

7.3.5.4 FC Data Frame Format

FC Data Frame is identified by the R_CTL field. See also Section 9.3.2.4.3 for further Data packets verifications rules by the DDP offload logic.

- R_CTL-->Information (least significant four bits) equals 0x1 (solicited data)
- R_CTL-->Routing (most significant four bits) equals 0x0 (device data)

7.3.5.5 FC RDY Frame Format

FC RDY Frame is identified by the R_CTL field. The FC RDY packet is shown in the diagram below (excluding the Ethernet and FCoE encapsulation).

- R_CTL-->Information (least significant four bits) equals 0x5 (data descriptor)
- R_CTL-->Routing (most significant four bits) equals 0x0 (device data)

Table 7-51. FCP RDY Packet format (special case with no extended and optional headers)

Field Size [bytes]	Description
24	FC Header: R_CTL... PARAM
4	FCP: Data Read Offset (processed by the hardware)
4	FCP: Burst Length (processed by the hardware)
8	FCP: Additional Parameters (N bytes, most likely N = 8)
4	Fibre Channel CRC (FC_CRC)

7.3.5.6 FC RSP Frame Format

RSP Frame is identified by the R_CTL field. See also Section 9.3.2.4.2 for further RSP packet verification rules by the DDP offload logic and also Section 9.2.3 for RSP packet generation by the ETSO offload logic including the FC payload.

- R_CTL-->Information (least significant four bits) equals 0x7 (command status)
- R_CTL-->Routing (most significant four bits) equals 0x0 (device data)
- TYPE equals to 0x08 // FCP
- F_CTL is Exchange Responder, Last Sequence, Last Frame in a Sequence (and the first one as well), Transfer Sequence Initiative, PARAM is not used as an offset

7.3.5.7 FC RD Request / WR Request Frame Format

Read request and Write request Frames are identified by the R_CTL field equals to unsolicited command. This frame consist a whole sequence sent from the originator to the responder with all matched flags in the F_CTL.

- R_CTL-->Information (least significant four bits) equals 0x6 (unsolicited command)
- R_CTL-->Routing (most significant four bits) equals 0x0 (device data)
- TYPE equals to 0x08 // FCP
- F_CTL is Exchange Context = 0 (Originator), First Sequence = 1
- SOF = SOFi, EOF = EOFt

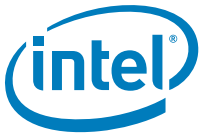
7.3.5.8 FIP Packet Formats

FCoE Initialization Protocol (FIP) packet format is identified by its Ethertype equals to 0x8914. No offloads other than Ethernet CRC and identification as FIP packet.

Table 7-52. Detailed FCoE Packet Structure without Extended and Optional headers

	msb 31 lsb 24	msb 23 lsb 16	msb 15 . 1 lsb 8	msb 7 lsb 0
	First byte on the wire			Last byte on the wire
0	Destination Ethernet MAC Address			
4				
8	Source Ethernet MAC Address			
12	Optional Ethertype Headers (T bytes while T equals 4 or 8)			
12+T	802.1Q Tag (VLAN + Priority)			
16+T	FIP Ethernet Type = 0x8914		Ver 2 11	Reserved
20+T	FIP Operation Code		Reserved	FIP SubCode
24+T+S	Descriptor List Length		Flags	
28+T+S	Descriptor List (N bytes)			
28+T+S+N	PAD, if needed to minimum length or Max Receive Size (M bytes)			
28+T+S+N+M	Ethernet CRC			

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



7.4 Internal Switch

7.4.1 Switch Concept

The XL710 includes a number of switch elements. Part of these elements are generic L2 switching elements and part are dedicated elements that can implement a specific functionality.

These elements are compatible to the IEEE P802.1Qbg specification. Section 7.4.2 describes the possible usage models of the switch as defined in these specifications. Section 7.4.3 describes the management protocol used to configure the switch.

The different elements are described in Section 7.4.4. Each switch can be connected to different ingress and egress ports in different configurations as described in Section 7.4.5. The algorithm that defines which switch routes which packet is described in Section 7.4.8.

The switches can have Virtual Station Interfaces connected through the PCIe interface, a physical port connected via the Ethernet interface and management ports either internal (Embedded Processor - EMP) or connected to a PCIe function. Details about the ingress and egress ports of the switch can be found in Section 7.4.5. The internal memory of the switch is described in Section 7.7.1.1.5.

Virtual Station Interfaces contain a set of queues or flows and are characterized by an L2 identifier. After the packet is forwarded to a specific ingress port, a internal destination inside the port is selected.

In addition to forwarding services, the switch supports also some additional services as described in Section 7.4.6.

The switch is not a learning switch and it is managed by the host, The programming interface exposed to the Operating system is that of a managed switch. Each switch element may be configured either via a driver running on a PF or via the EMP. When a driver programs the switch it uses admin commands processed by the EMP as described in Section 7.4.9.

7.4.2 Switches configurations

XL710 embedded switch can be used to offload the data plane functions of the virtual switching capability supported by many VMMs (bridging and switching are used synonymously in this document).

This section describes the logical models that can be represented by the switching elements. The actual switching elements definition can be found in Section 7.4.4.

XL710 switch can be configured to operate in different modes as described below:

- Internal switching configurations — Virtual embedded bridging:
    - Software based switching in VMM
    - Hardware based switching for direct connected Virtual machines
    - Local switching between virtual machines
- External switching configurations:
    - Virtual Port aggregator where VM to VM switching is performed by an adjacent bridge
    - Port Virtualizer configuration where the VM to VM switching is performed by a controlling bridge that could be in the network edge or aggregation layers.

Note: A Device may work either in Port aggregator or in Port Virtualizer mode. The mode of operation should be the same for all the ports.

- Combination of Internal and external switching configurations:
  - Internal switching modes, SW switching model or direct connect model, can coexist with one of the external switching modes, Port aggregator or Port Virtualizer.

7.4.2.1 Internal Switch Configurations

In the internal switching mode, the XL710 switching element can be configured to offload data plane functions of the VMM based SW switches or can handle data plane switching functions for direct connected VMs (through SR-IOV or Multiple PCI functions).

7.4.2.1.1 Hardware Offload of SW Virtual Bridges (VEB)

Figure 7-27 illustrates the hardware offload of SW based switches.

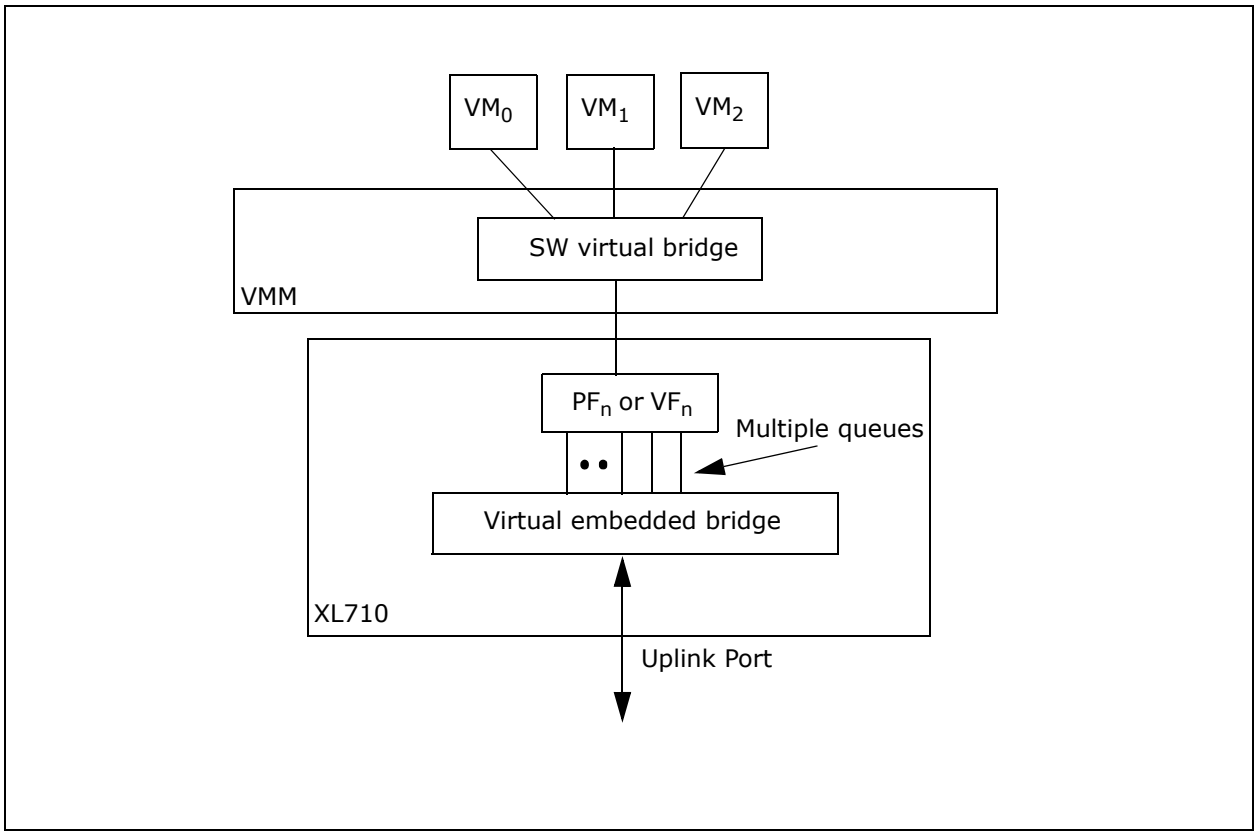


Figure 7-27. Internal switching configuration — SW Virtual embedded bridging

When the XL710 switching element is configured to operate in this mode, data plane functions such as the filtering and forwarding functions are offloaded to the hardware. The switching element can also perform HW replication of multicast and broadcast frames. In addition the switch can perform VLAN offload functions and security functions such as MAC address spoofing. In this mode multiple queues

are assigned to the PCI function (PF or VF in case of SR-IOV). Each set of queues represents a VSI (Virtual Station Interface) in the switching element. The uplink of the switch is connected to one of the external Ethernet MAC ports.

The entire control plane functionality is handled by the software switch in the VMM via a PF. The switch hardware is programmed by a set of switch APIs. The switch control plane software manages the switching elements forwarding database, VLAN membership, bandwidth assignment or other virtual port characteristics. Since the forwarding path is still through the software intermediary layer in the VMM, it is possible for the software switch to examine the packets or perform additional functions on the frames, for example handling new protocol headers, collecting additional statistics etc.

7.4.2.1.2 Virtual Embedded Bridge, for Direct Connected VMs

Figure 7-28 illustrates the HW switching of direct connected virtual machines.

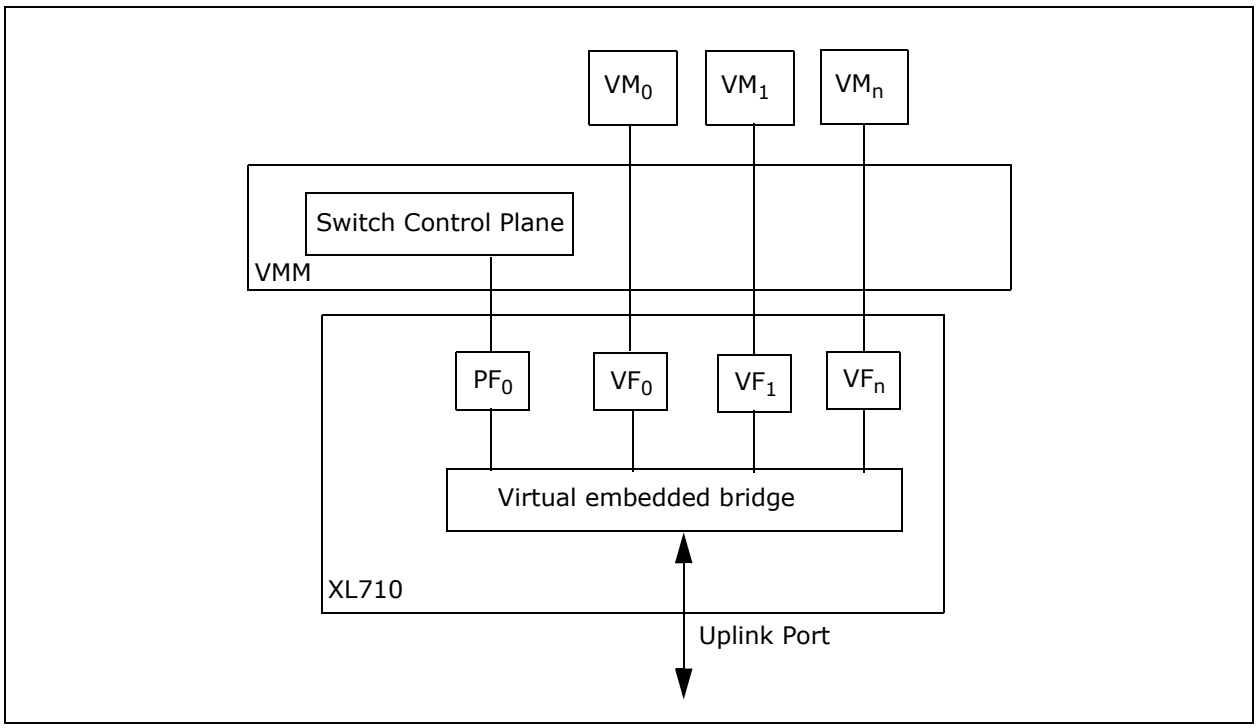


Figure 7-28. Internal switching configuration — HW Virtual embedded bridging

When the XL710 switching element is configured to operate in this mode, each of the Virtual Station Interfaces are directly connected to PCIe functions (PFs or VFs) that are assigned to the VMs. The software layer is eliminated in the forwarding path hence the entire data plane functionality is handled by the HW embedded switch. All the virtual switching functions such as VLAN membership, filtering and forwarding, Bandwidth management, statistics etc., are handled in HW.

The entire control plane functionality is handled by the switch management agent in the VMM. Control packets are forwarded to the control plane agent through a management port. The switch hardware is programmed by a set of switch APIs. The switch management software manages the switching elements forwarding database, VLAN membership, bandwidth assignment or other VSI characteristics. The switch management can monitor the traffic by accessing the statistics counters or can also monitor data traffic to VM by enabling mirroring functionality in the embedded switch.

Multiple virtual embedded switching elements can be instantiated in the XL710. There can be only one virtual switching element per uplink port. If there are multiple switching elements on a single uplink port, then a multi-channel capable S-component (See Section 7.4.4.2.1) needs to be configured on the uplink. However a virtual switching element can be configured without an uplink port for local switching between VMs. This is used for configuring internal networks.

7.4.2.2 External Switch Configurations

XL710 supports external switching modes. In the external switching modes the virtual switching functionality including VM to VM, and VM to uplink forwarding is handled by external switches. This enables a single network management domain to manage both enterprise switches and virtual switching. The entire data plane functionality and the control plane functionality is handled by the external switch.

Two protocols are being defined by IEEE to enable virtual bridging in external switches, IEEE P802.1Qbg and IEEE P802.1BR. XL710 supports both the models. The virtual port aggregator model is very similar to the embedded virtual switching model.

7.4.2.2.1 Virtual Port Aggregator Model

Figure 7-29 illustrates a Port aggregator configuration.

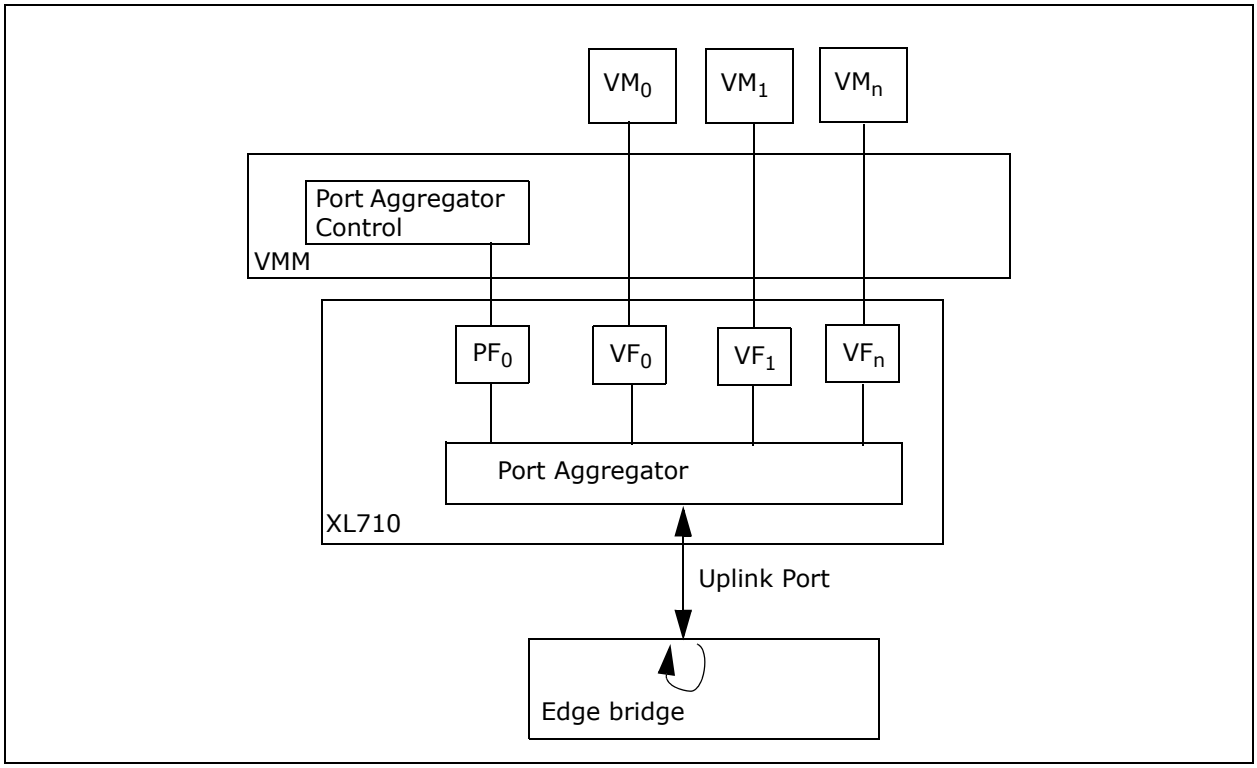
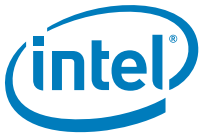


Figure 7-29. External switching configuration — Virtual Port Aggregator



In a simple port aggregator mode, there is no change to the Ethernet frame format. During the ingress direction, the operation is similar to virtual embedded bridging, the packets are forwarded to the VMs by querying into the forwarding database in the embedded switch. In the egress direction all traffic including the VM to VM traffic is forwarded the external switch. The external switch performs lookup on the packets and turns-around the packet back to the end node (called reflexive relay service or hairpin forwarding) if the packet has to go to another VM residing in the same end station. Since all packets flow through the external switch, all advanced features of the external switch (e.g. advanced ACLs and flow monitoring features) are available for VM to VM traffic. However the bandwidth of the uplink is shared between the VM to VM traffic and the uplink traffic; also there is additional latency for externally switched VM to VM traffic.

In the Port Aggregator mode, the HW performs broadcast and multicast replication functions. During multicast and broadcast, the packet is forwarded by the external switch once to the end station and the Port aggregator in the XL710 replicates the packet multiple times to all the Multicast member VMs attached to the Port aggregator.

The Virtual Station Interfaces (VSIs) of a Port Aggregator can be directly connected to the VMs through PCIe functions or VFs in SR-IOV mode. A cascaded Port Aggregator configuration is also allowed where a software based Port Aggregator in VMM can be connected through one of the virtual ports. In this case the port is designated as a cascaded port (See Section 7.4.2.4).

The discovery and configuration of the Port Aggregator mode is handled by a software Port Aggregator control agent residing in the VMM. The HW switch element is managed through a set of APIs accessible to the Port Aggregator control agent.

7.4.2.2.2 Port Virtualizer model

There are two ways to divide the link to channels:

- A Port Mapping S-VLAN component (a.k.a S-comp) defined in the IEEE 802.1Qbg specification. The S-comp uses s-tags (Ethertype = 0x88A8) to define s-channels that do not support replication. Section 7.4.2.2.2.1 and .Section 7.4.4.2.1.
- A port mapping based on MAC address. This mode is used when a function is dedicated to storage function and uses a small and exclusive set of MAC addresses. This mode supports two channels only. A storage channel and a default channel receive all the other traffic. See Section 7.4.3.

Figure 7-30 illustrates the HW switch element in the XL710 configured to operate in S-comp configuration.

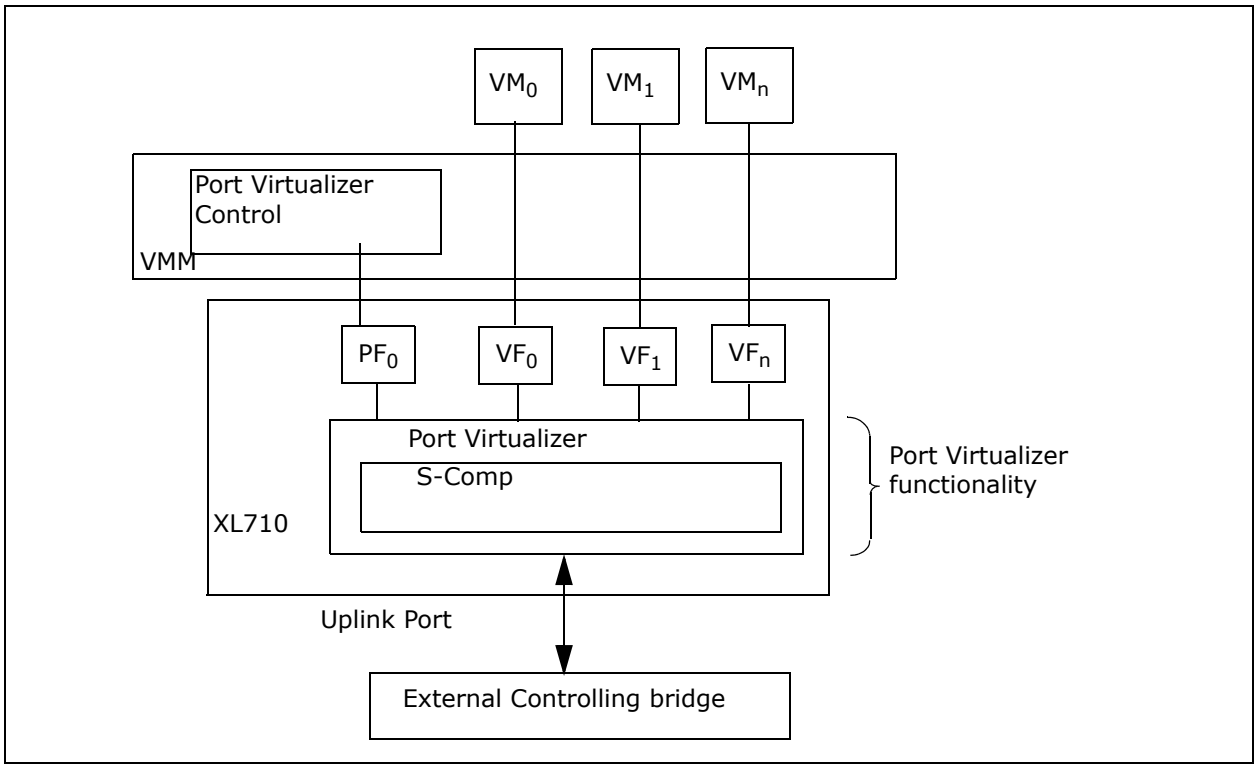


Figure 7-30. External switching configuration — Port Virtualizer

7.4.2.2.2.1 S-comp model

In an S-comp Port Virtualizer configuration, the controlling bridge's ports are logically extended to the virtual stations (or VMs). A physical port in the controlling bridge is configured to have multiple logical channels (or logical ports), typically one per VM. Correspondingly the physical port in the end node is also configured to have same number of logical channels (or logical ports) that are associated with VMs. The packets need to explicitly carry a Tag to identify the logical channels (or logical ports). An S-component (service VLAN component) is configured to provide this multi-channel capability. The S-Tag indicates a logical channel or logical port on the controlling bridge. In this mode of operation, each Virtual Station Interface in the HW switching element is associated with a logical channel in the S-component.

7.4.2.2.2.2 Connection to the host

The Virtual Station Interfaces of a Port Virtualizer can be directly connected to the VMs through PCIe functions or VFs in SR-IOV mode. A cascaded Port Virtualizer configuration is also allowed where a software based Port Virtualizer in VMM can be connected through one of the virtual ports. In this case the port is designated as a cascaded port (See Section 7.4.2.4). The discovery and configuration of the Port Virtualizer mode is handled by a software Port Virtualizer control agent residing in the VMM. The HW switch element is managed through a set of APIs accessible to the Port Virtualizer control agent in VMM or IOVM.

7.4.2.2.3 SAN and LAN MFP Forwarding

Figure 7-31 illustrates the switch structure for an MFP configuration of two functions, one for LAN and one for SAN. The switch provides the following functionality:

- The S-comp selection of S-channels is based on MAC address (see Section 7.4.4.2.3).
- No forwarding of transmit traffic between functions. A VEB may still forward transmit traffic within the LAN function.
- Receive unicast packets are forwarded to either LAN or SAN, based on destination MAC address (exact match).
- Receive multicast packets are forwarded to either LAN or SAN, based on destination MAC address (exact match).
- Receive broadcast packets are forwarded to LAN.
- Receive packets that do not match any MAC address are forwarded to the LAN.
- LAN L2 filtering may then decide to drop such packet.
- VEB/VEPA support is expected only on the LAN function.
- FCoE traffic is over SAN functions only. No FCoE traffic is expected over LAN function.
- Sx operation (see Section 7.4.4.8) is associated with the LAN function. Thus WoL is supported only over LAN function.

During the Cold Reset sequence, the device configures a Port Virtualizer with two SEIDs (one per each PCI function). The LAN PF may then proceed and configure a VEB on top of its Switch Port.

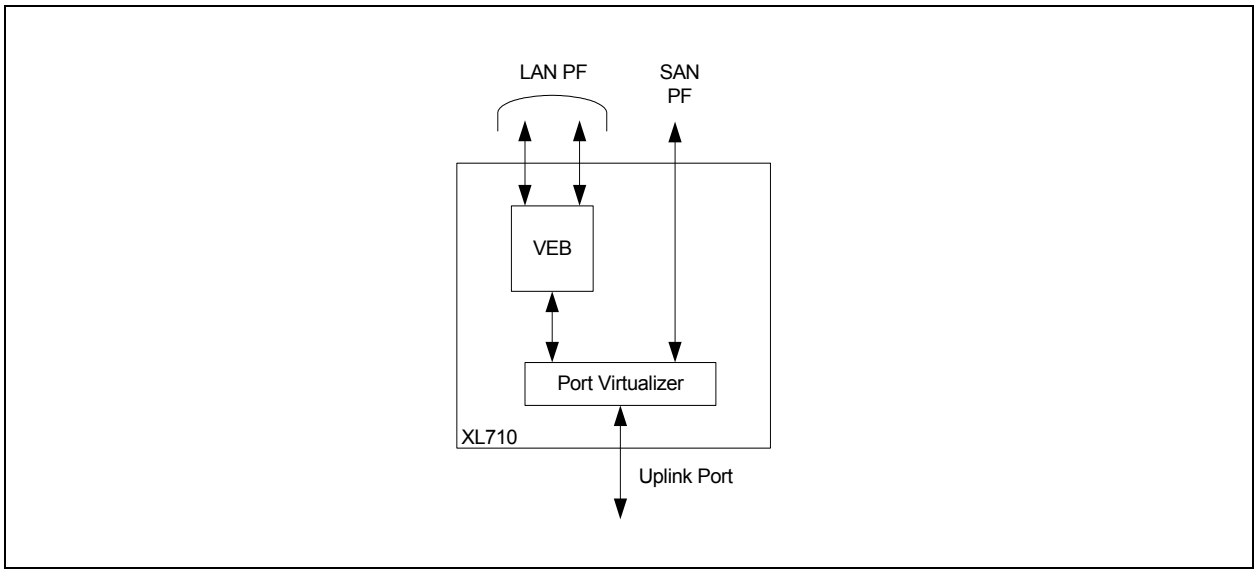


Figure 7-31. Internal Switch Configuration - LAN & SAN PFs

7.4.2.3 Coexistence of Internal/External Switch Configurations

Both internal and external switching configurations can coexist within an end station accessible through the same uplink port. Legacy VMs or applications that need higher performance for VM to VM traffic will require virtual embedded bridging functionality to perform internal switching. In such usage models, the virtual embedded bridge will coexist with either a Port Aggregator or Port Virtualizer in the same end station.

Figure 7-32 illustrates a combination of Virtual embedded bridges with a Port Aggregator.

In this configuration multiple switching elements are instantiated in the XL710 on the same uplink port. Each instance is accessible through a multi-channel S-Component. The S-Component is used to mux/demux multiple logical channels (S-Channels) from the same physical port. The S-Tag identifies the logical channels (S-Channels) or logical ports that are extended from the adjacent bridge.

In this example configuration, the legacy VMs are connected through a SW based virtual embedded bridge connected through one of the S-channel ports. A second VM (that hosts a virtual appliance) is directly connected to the external switch through the second S-Channel port. In addition, a Port Aggregator is configured through the third S-channel port that provides external switching functionality to VMs. Certain higher performance applications that require heavy VM to VM traffic (for example co-located front end and mid tier application servers) are configured through directly connected VFs to an embedded bridge. This embedded bridge is accessible through fourth S-channel port.

Discovery and configuration of individual switching components is performed through appropriate control agents in the VMM or IOVM.

The configuration of the switch is done using the CDCP protocol as defined in 802.1Qbg specification.

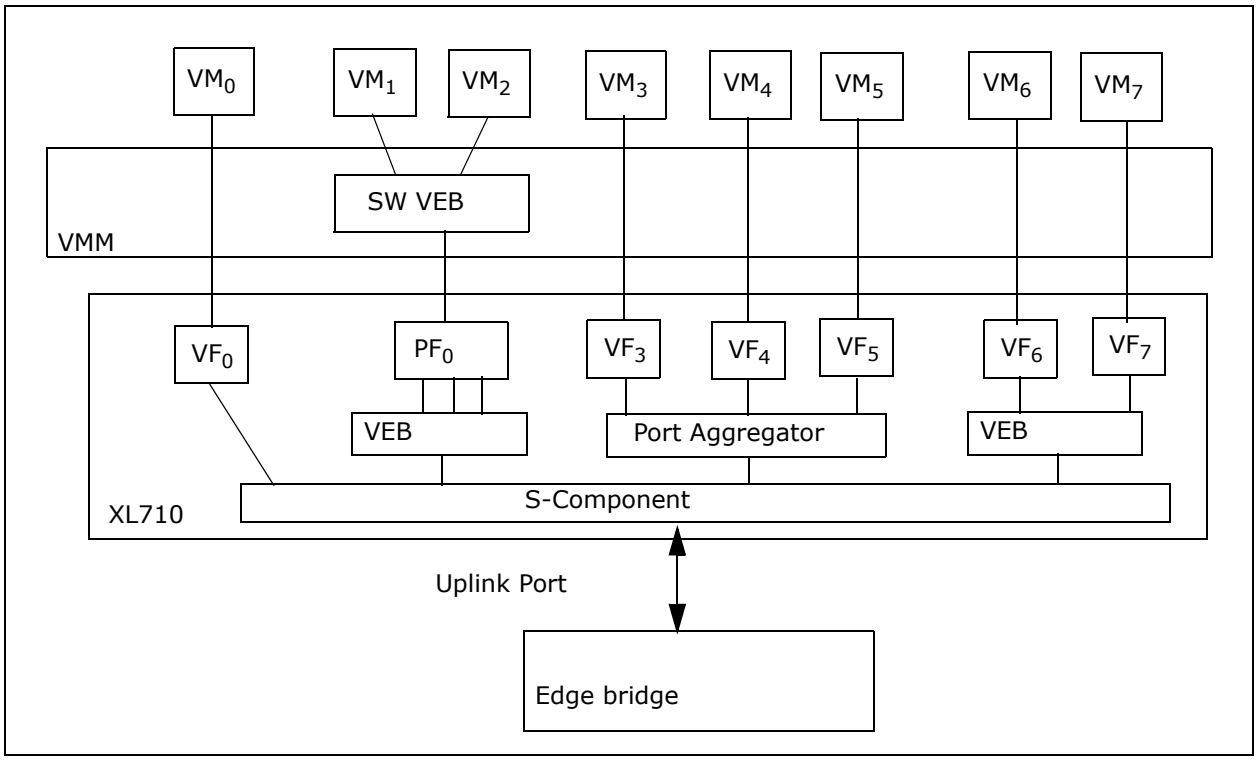


Figure 7-32. Cascaded Virtual Ethernet Bridge

7.4.2.4 Cascading of Software VEBs, Port Aggregators and Port Virtualizers

Software based VEBs, Port Aggregators and Port virtualizers could be cascaded over hardware based VEBs, PVs and PAs in the NIC. This section provides illustration of typical cascaded configurations allowed by the XL710.

7.4.2.4.1 Cascaded HW and SW VEBs

Figure 7-33 provides illustration of a SW VEB cascaded over a HW VEB.

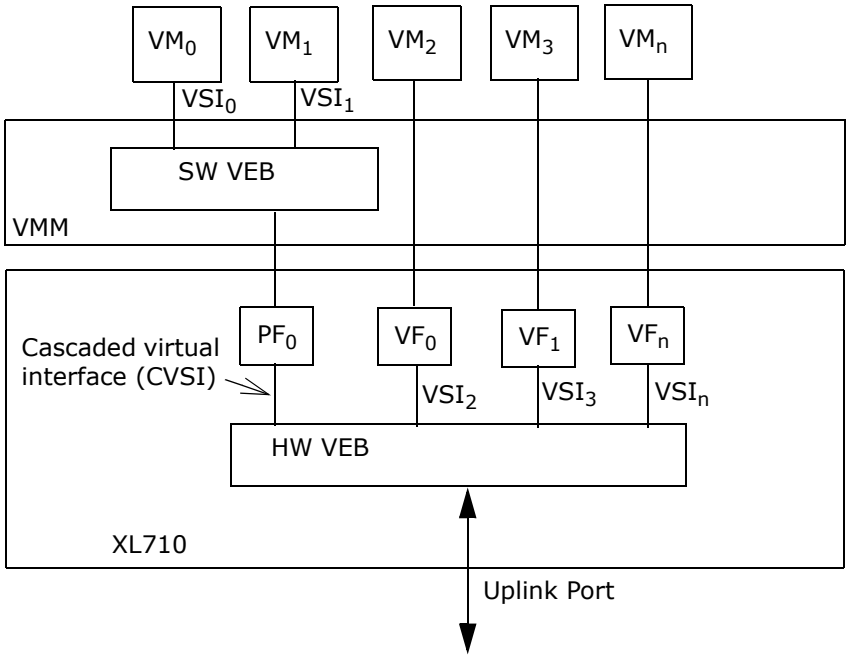


Figure 7-33. Cascaded Virtual Ethernet Bridge

In this configuration a HW VEB is instantiated in the XL710 controller. The VEB is configured with multiple Virtual Station Interfaces (VSIs). A VSI refers to a connection between the VEB and a virtual end station and its associated resources such as MAC address, VLAN, BW/QoS attributes etc. Typically a VSI is configured with a single MAC address and VLAN. A software VEB is configured in the VMM and the connection between the SW VEB and a port in the HW VEB is referred to as cascaded Virtual station interface (or simply a downlink port). The cascaded interface is a special VSI that is used as a downlink to a software VEB. This type of port is referred to as trunk port in typical Ethernet switches. The downlink ports are typically configured with multiple MAC addresses and multiple VLANs serviced by the SW VEB. Security features such as MAC anti spoofing and VLAN ingress checks (See Section 7.4.6.1.2) should not be configured on the downlink ports since those functions are performed at the ingress VSIs of SW VEB.

XL710 can be configured to perform VLAN offload functions on the downlink VSIs. In this case, the VLAN information is carried in the transmit descriptor and the XL710 will insert the VLAN in the transmit direction. XL710 can strip the VLAN in the receive direction and store the VLAN tag information in the receive descriptor.

7.4.2.4.2 Cascaded HW and SW Port Aggregators

Figure 7-34 provides illustration of a SW VEPA cascaded over a HW VEPA.

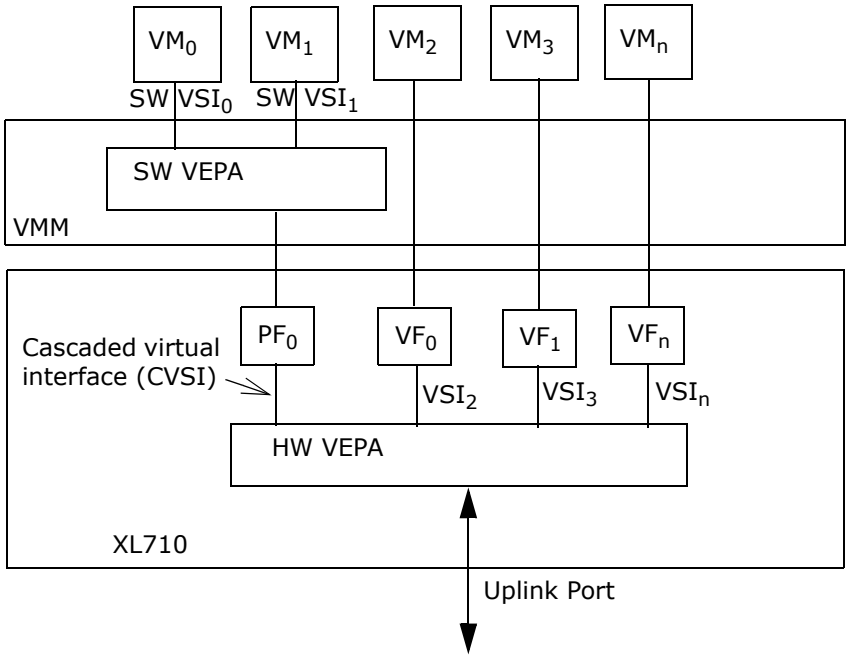


Figure 7-34. Cascaded Virtual Ethernet Port Aggregator (VEPA)

In this configuration a HW VEPA is instantiated in the XL710 controller. The HW VEPA is configured with multiple Virtual Station Interfaces (VSIs). A software VEPA is configured in the VMM and the connection between the SW VEPA and a port in the HW VEPA is referred to as Cascaded Virtual station interface (or simply a downlink port). The cascaded interface in this case is a special VSI that is used as a downlink to a software VEPA. The only cascaded configuration allowed over a HW VEPA is a SW VEPA. A SW VEB cannot be cascaded over a HW VEPA, this configuration is not allowed by the IEEE P802.3bg specification.

The downlink ports in a VEPA are similar to a VEB trunk port (see Section 7.4.2.4.1) except that Multicast source address pruning functionality specified by IEEE P802.3bg is not configured on a downlink port to a SW VEPA; this functionality is performed at the egress of the software VEPA VSIs connected to the VMs.

Security features such as MAC anti spoofing and VLAN ingress checks (See Section 7.4.6.1.2) should not be configured on the downlink ports since those functions are also performed at the ingress VSIs of SW VEPA.

XL710 can be configured to perform VLAN offload functions on the downlink VSIs. In this case, the VLAN information is carried in the transmit descriptor and the XL710 will insert the VLAN in the transmit direction. XL710 can strip the VLAN in the receive direction and optionally store the VLAN tag information in the receive descriptor.

7.4.2.4.3 Cascaded VEB and Port Virtualizers

Figure 7-35 provides illustration of a SW VEB, HW VEB and a Software Port Virtualizer cascaded over a Hardware Port Virtualizer. A Port Virtualizer logically extends the ports of an external bridge to virtual end stations as explained in Section 7.4.2.2.2. XL710 supports hardware based Port Virtualizer for directly connecting virtual end stations to extended bridge ports using VFs in SR-IOV configuration.

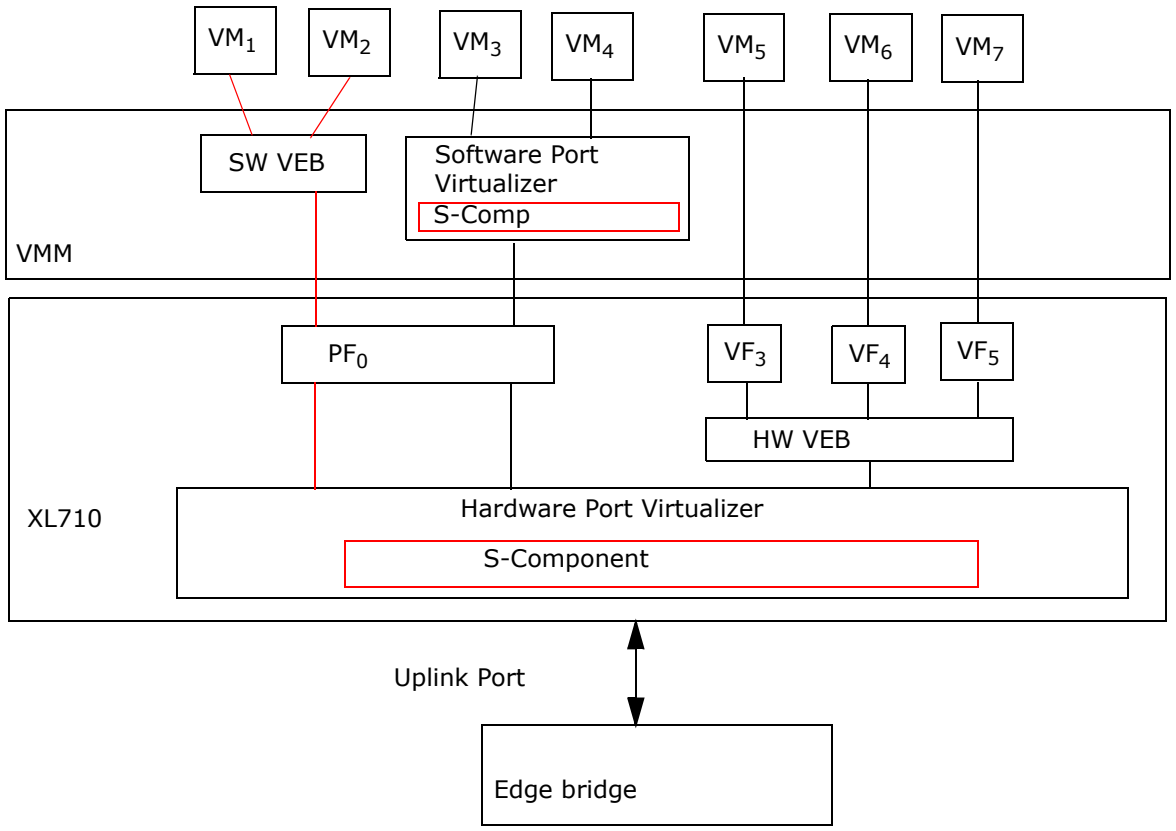
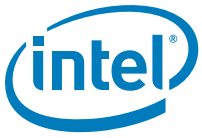


Figure 7-35. Cascaded VEB and Port Virtualizer

7.4.2.4.3.1 Cascaded SW port Virtualizer over a HW port Virtualizer

XL710 supports cascading a software based Port Virtualizer in VMM to the HW based Port Virtualizer in the network controller. A software Port Virtualizer is configured in the VMM and the connection between the Software Port Virtualizer and a port in the Hardware Port Virtualizer is referred to as Cascaded Virtual station interface (or simply a downlink port). The cascaded interface in this case is a special VSI that is used as a downlink to a software Port Virtualizer. A VSI connection between a VM and a Port Virtualizer uses single S-Tag to identify the logical bridge port. However, a cascaded VSI (downlink port) connected to a Software based Port Virtualizer carries traffic to multiple Software based VSIs serviced by the software Port Virtualizer. So multiple S-Tags need to be configured on this downlink VSI. Receive Multicast replication functionality and source pruning functionality should be disabled on these downlink ports.

XL710 supports VLAN and S-Tag offload functions for Software based Port Virtualizers connected to the downlink VSI (See Section 7.4.5.2.1.2). XL710 performs S-Tag and VLAN tag insertion functionality on transmit direction and can optionally strip the VLAN tag and S-tags in the receive direction. In this case,



the VLAN tag and S-Tag information is carried in the transmit descriptor and the XL710 will insert the VLAN tag and S-Tag in the transmit direction. XL710 can strip the VLAN tag and S-Tag in the receive direction and store the VLAN and S-tag information in the receive descriptor.

XL710 can also classify ingress frames based on S-Tag and forward to selected queues within the downlink VSI connected to Software Port Virtualizers (similar to VMDq1 mode). The XL710 offload functions as described above enhance the performance of SW based port virtualizers. See Section 7.4.6.3 for additional details on SW port Virtualizer offload functionality.

Note: A VSI used as a cascaded port virtualizer should not activate L2 filters on the aggregated S-channels. The VSI should be defined with a *Connection Type* of *Default Port*.

7.4.2.4.3.2 Cascaded SW/HW VEB Over a HW Port extender (or Multichannel S-comp)

XL710 supports cascading of Software VEBs to Hardware Port Virtualizer . In this configuration, a Software VEB can be configured in the VMM and the connection between the SW VEB and a port in the Hardware Port Virtualizer is referred to as Cascaded Virtual Station interface (or simply downlink port). XL710 supports VLAN offload functions for SW VEBs. The SW VEB is not aware of the S-Tag functionality. The Hardware Port Virtualizer (or multichannel S-Component) performs the insertion and stripping of S-Tag to the cascaded VSI (downlink port) connected to the VEB. XL710 can classify the ingress frames based on {VLAN, MAC address} pair and forward to selected queues within the downlink VSI connected to Software VEBs (VMDq1 mode). See Section 7.4.5.2.1.2 for additional details on offload functions available for software VEBs.

XL710 supports cascading of HW VEBs to Hardware Port Virtualizer. HW VEB is configured in the XL710 to support direct connected VMs in SR-IOV mode of operation. In this configuration a HW Port Virtualizer (or multichannel S-component) and a HW VEB can be instantiated in the XL710. The HW VEB can be connected through a cascaded (or downlink) port in the HW Port Virtualizer. Logically the hardware VEB's trunk port is connected to the external controlling bridge through a logically extended bridge port identified by an S-Tag. The hardware Port Virtualizer (or multichannel S-Component) performs insertion and stripping of S-Tags to the cascaded downlink port connected to the HW VEB.

7.4.3 Edge Virtual Bridging Management Protocol

This section describes the control and management protocol for discovery and configuration of Edge Virtual bridging (EVB) components between the end stations and external edge switches. The current IEEE Edge Virtual bridging (EVB) supports multiple different switching elements: Virtual Ethernet bridges, Virtual Ethernet Port Aggregators and Port Extenders for virtual networking. A control protocol is required to discover the capabilities of the end stations and external switches and configure the appropriate switching elements at both ends.

LLDP protocol is used for initial discovery of EVB capabilities between the switch and the end station. New TLVs (type, length, values) are defined that allows both end station and edge switch to advertise the capabilities, such as:

- The type of switching element
- Whether VEB, VEPA, or S-compare supported
- The number of switching elements supported etc.

DCBX protocol over LLDP is used to configure ETS, PFC on the physical link.

In case of EVB, the LLDP protocol with additional TLVs allows advertisement of whether multi channel capabilities (S-Component) are supported, the number of channels need to be enabled, etc. The TLV exchange is expected to be similar to DCBX protocol that is used for DCB parameter exchange.

In case of Port Virtualizer, the LLDP protocol with additional TLVs will allow advertisement of Port Virtualizer that includes an S-Component and M-Component, number of extended logical bridge ports (S-channels), and the number of multicast table entries etc.

The LLDP packet is the standard Ethernet frame defined with LLDP Ethertype of one of the group multicast MAC address used by LLDP as defined in IEEE Std 802.1AB. The LLDP frame format is illustrated in Figure 7-53.

Table 7-53. LLDP frame format

0	15 16	18 19 20
88-cc (LLDP Ethertype)	LLDPDU	

The end station LLDP agent is expected to validate the LLDP frame by examining both the DA MAC address and the LLDP Ethertype. The group MAC address used is 01-80-C2-00-00-0E (nearest bridge address) or 01-80-C2-00-00-00 (nearest customer bridge address). The first address above is used for LLDP communication with S-components and the second address is used to forward the LLDP frame to an adjacent C-VLAN bridge (VEB or VEPA) separated by an S-VLAN component.

Once the EVB switching elements are discovered and configured after the LLDP protocol exchange the virtual station interfaces can be configured on the respective switching elements.

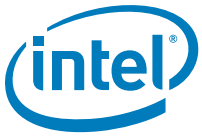
A new protocol (VSI discovery protocol or “VDP”) is defined by IEEE. The “VDP” protocol is transported over ECP (Edge Control Protocol) that allows exchange of VSI specific TLVs between the edge switch and the control plane of appropriate switching element - VEB, VEPA or Port Virtualizer. The TLV transfer protocol is expected to use the same group multicast destination address(es) reserved for LLDP as described above with a new Ethertype (to be defined). See Figure 7-54 for the frame format for the new TLV transfer protocol. A new Ethertype enables forwarding the protocol frames to the appropriate control plane of the respective switching element.

Table 7-54. New TLV transfer protocol frame format for VDP exchange

0	15 16	18 19 20
0x8940 (ECP Ethertype)	VDPDU	

The VDP protocol will use TLVs to discover and configure the VSIs in the end station. The VDP protocol will be used for establishing a new VSI and exchanging VSI attributes such as VSI Id, MAC, VLAN, QoS parameters, etc. The VDP protocol may also be used for exchanging VSI statistics from end station to edge switches. In Port Extender configuration, the protocol may also include mechanisms for the edge switch to push the multicast forwarding table to the Port Extender in the end station.

XL710 has the ability to filter for appropriate Edge Virtual bridge (EVB) control frames and forward to appropriate management ports (VSIs or VSI queues) of the respective switching elements. The control frames (for example LLDP or VDP) are expected to be exchanged by the respective control plane agent in the VMM (or IOVM) and the external switch. LLDP is a link level protocol, so a virtual machine is not expected to directly exchange LLDP control frames with an external switch (i.e. the control frames cannot cross the virtual bridge). If a VM is configured to exchange control frames, it will do so with the



appropriate virtual bridge control agent in the VMM. XL710 has a mechanism to either filter and/or forward the control frames from the VSIs to the management port of the switching element to which the VM is connected.

When a VEB or VEPA is instantiated over a Port Virtualizer, this is the equivalent of a VEB or VEPA uplink directly connected to a logical port in the external bridge. So the external bridge can exchange bridge control frames over the logical link to the VEB or VEPA. In this case the external bridge can send or receive LLDP frames to VEB or VEPA in the end station. These frames are identified using the S-Tags that correspond to the extended bridge ports. Any of the bridge control frames can pass through this S-channel (e.g. LACP, new TLV transfer protocol/VDP). The bridge control frames over the S-Channels are not expected to be VLAN tagged, so these control packets will be untagged when delivered to the VEB or VEPA by the S-component. XL710 can forward those control frames to the respective VEB or VEPA control ports (VSIs).

So in a multi-channel configuration there could be multiple LLDP agents (peer-to-peer) established between the external bridge through extended bridge ports (S-channels) and the corresponding VEB, VEPA or Port Virtualizer components in the end station.

In the Port Virtualizer mode, once the Port Virtualizer is discovered and configured, one of the S-Channels could be designated to carry management frames to the Port Virtualizer control agent in the end station. XL710 can forward those control frames to the designated Port Virtualizer control port. A default port will be defined for the S-Component, so any control frames that do not match the control frame filtering rules will be forwarded to this default control port.

Note: After a link down/link up event, the switch configuration protocol should be run again, as the partner (and its capabilities) may have changed.

7.4.4 Switch Elements

Figure 7-36 describes the possible switching elements on one of the XL710 ports and the connections between them. Every switching element is optional. In the minimal case, the Ethernet port may be connected to a PCI function (“Virtual Station Interface”) with no switching capabilities at all. In this minimal case, an L2 filter is used as in regular NICs.

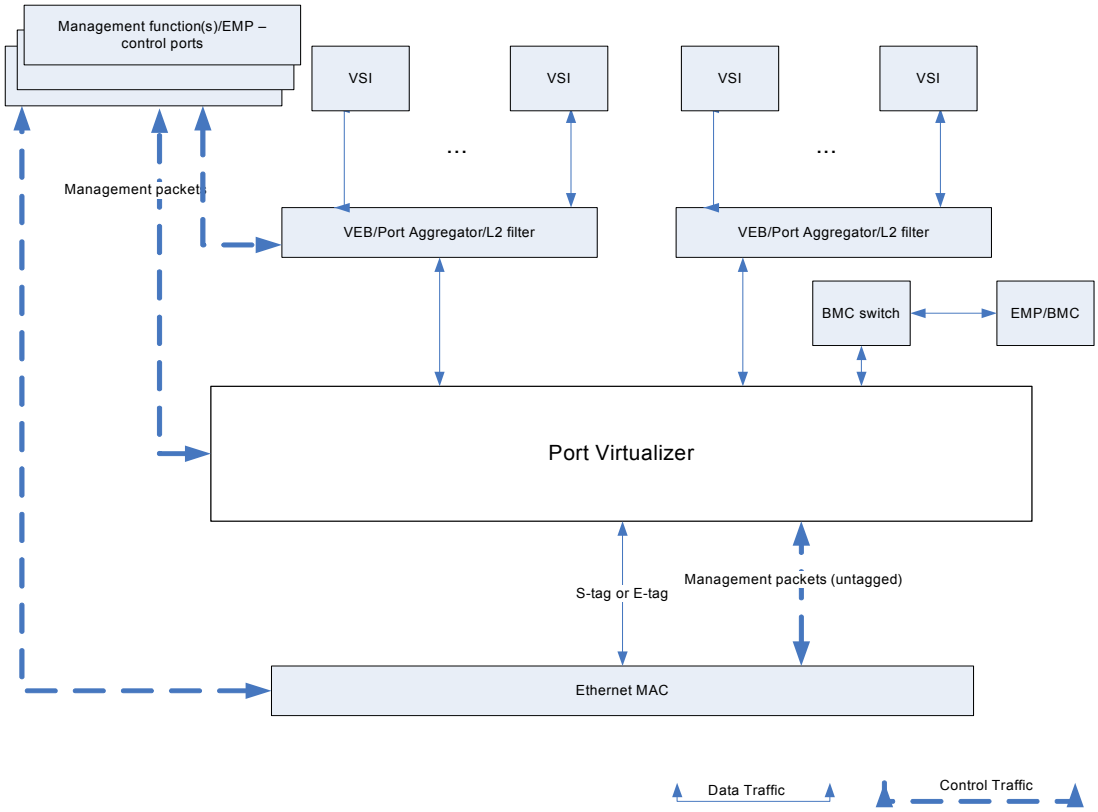


Figure 7-36. Switching elements per port

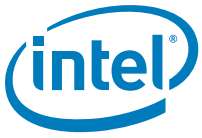
There can be up to one port virtualizer per physical port. There can be up to 16 concurrent VEB or Port Multicasting defined in the device. There can be up to one L2 filter per Virtual Station Interface.

7.4.4.1 MAC

The functionality of the MAC (CRC handling, error detection, etc.) is described in Section 3.2.1. This section only describes the switching decisions of the MAC.

The MAC switching behavior is set according to the following parameters:

- Port Enable
- Link Status
- Save Bad Packets (Stored in the *PRT_SBPVSI.SBP* field)
- Default Port VSI (Stored in the *PRT_SWT_DEFPORTS.DEFAULT_VSI* field)
- Local MAC address (per LAN port)



- One or more control ports of the MAC. If a Port Virtualizer or a VEB is directly connected to this MAC, they may share the same control ports. At init time, a single control port is connected to the EMP. At a later time, if requested by the driver, an additional control port on a VSI of one of the physical function may be added or may replace the EMP control port.

As a rule, the MAC forwards all the packets to the next stage, apart from the following packets:

- Packets with L2 error - these packets are dropped, unless the *Save Bad Packets* attribute of the MAC is set. In this case, the MAC must define a control port and all the error packets are forwarded to this port. The errors included in this category are:
  - CRC errors
  - Alignment errors
  - Length errors- either too big or too small
- Packets with the MAC local MAC address. If a *Station MAC address* is defined, packets with this address may be forwarded to the one of the control ports of the MAC in combination with some Ethertype filters. Each control port may require forwarding of these packets using the *Add Control Packet Filter* or the *Add MAC, VLAN pair* admin commands.
- Link Local packets without S-tag:
  - Flow control packets (both 802.3x link flow control and 802.3bd priority flow control frames) are handled by the MAC and are not forwarded.
  - Any packet defined in a control VSI forwarding rules (see Section 7.4.5.2.2.2)

7.4.4.2 Port Virtualizer

An Ethernet port can be connected to a Port Virtualizer.

A Port Virtualizer can be either an S-comp (Section 7.4.4.2.1) or a Port Extender (Section). Each port virtualizer should be connected to a control port that will receive the control protocol (CDCP) packets.

There can be one Port Virtualizer per physical port. These two elements are mutually exclusive and in given device only one type of port virtualizer can be initiated on all ports.

A Port Virtualizer can be created using the *Add Port Virtualizer* admin command described in Section 7.4.9.5.5.1.

If a Port Virtualizer is not implemented on a port, S-tagged packets will be dropped.

7.4.4.2.1 Service-VLAN Component (S-comp)

An S-component is basically a mux/demux component that divide the Ethernet port to a set of S-channels.

Each channel in the S-component can be attached as a physical port to one of the switches. If no switch is needed, anS-comp is used to directly connect S-channels to VSIs.

7.4.4.2.1.1 S-tag Format

The S-tag format is the format defined for S-tags in the IEEE 802.1ad spec as described below:

Table 7-55. S-tag Format

0	15	16	18	19	20	31
88-a8		PCP		DEI	S-VID	

- 88-a8: The S-tag Ethertype
- PCP: Priority Code Point - the priority bits of the S-tag. These bit can be derived from the VLAN priority bits.
- DEI: Drop Eligible Indicator - this bit is ignored.
- S-VID - the S-tag ID

7.4.4.2.1.2 S-comp Receive Flow

The S-comp acts only on packets with an S-tag received from the uplink. These packets are forwarded to one of multiple S-comp channels according to the S-tag forwarding table, thus forwarding them directly to VEB switches or VSIs.

If an S-component is defined in the system, packets without S-tag or with an unrecognized S-tag not forwarded to one of the control ports are sent to the default port. If such a port is not defined, they are dropped.

The S-tag is usually removed by the S-component and is not forwarded to the host or to the sideband interface. This functionality is controlled by the *SS-tag extract mode* parameter of the *Add VSI* admin command. For cascaded S-comp ports that represents multiple S-channels, the S-tag can be either left in the packet or removed and stored in the *L2TAG2 (1st)* field of the receive descriptor. The *L2TAG2 (1st)* field of the receive descriptor contains the S-tag if the *S-tag extract mode* parameter of the VSI is set to 10b and is valid if the *L2TAG2P* flag in the descriptor is set. If S-tag extraction to descriptor is required the VSI must use 32 bytes receive descriptors.

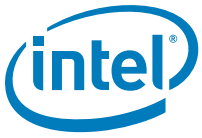
The algorithm used by an S-comp is described in Section 7.4.8.2.

7.4.4.2.1.3 S-comp Transmit Flow

The S-comp adds the S-tag matching to the S-channel from which the transmit packet is received and forwards it to the physical port. It also forwards packets received from the control ports to the physical port.

Switch based insertion of the S-tag can be disabled using the *S-tag insert enable* parameter of the *Add VSI* admin command. For cascaded S-comp ports that represents multiple S-channels. the S-tag insertion should be disabled. In this case, the software may either send untagged packet, ask for S-tag insertion via the transmit descriptor or add it inline in the packet. This functionality is enabled by the *Accept Tag from host* parameter in the *Add VSI* admin command.

An S-comp only transfers packets from the S-channels to an ethernet link or vice versa. It does not replicate packets or forward packets between virtual ports or physical ports. A VEB connected on top of an S-channel can forward packets between VSIs contained by this VEB.



7.4.4.2.2 Port Virtualizer (S-comp) parameters

A port virtualizer is defined by the following parameter:

1. The Port Virtualizer connectivity:
   a. The Uplink connected.
   b. Control ports that will receive link local packets. The control ports may be VSIs or the internal embedded controller or both.
2. A Demux rule - what is the rule used to demux ingress packets. The usual rule is to use an S-tag as described above.
3. A default port. This port will receive all the traffic not identified as control packets and not forwarded to any of the S-channels. If this port is not defined, such traffic will be dropped.
4. A set of channels - Each channel is defined by the following parameters:
   a. The connected VSI or next level switch.
   b. An S-tag (S-VID) - see Section 7.4.4.2.1.1.
   c. A maximal and a minimal rate.
   d. S-tag strip enable.
   e. S-tag insert enable.
   f. Report S-tag.
   g. Accept tag from host.

7.4.4.2.3 LAN/SAN S-comp

The LAN/SAN Port Extender divides the Uplink Port between LAN and SAN traffic. The LAN channel may be attached to a switch (for example a VEB).

This S-comp differs from the standard 802.1Qbg port extender in that it uses the MAC address as the channel classifier. The MAC address behaves the same way as the S-tag, and it can translate to a single Switch ID.

7.4.4.2.3.1 Receive Flow

The S-comp filters on unicast and multicast addresses associated with the SAN function. A packet is forwarded to the SAN channel if it matches one of the MAC addresses (unicast or multicast) associated with SAN. Else, it is forwarded to the LAN channel.

Note: The MAC address is used twice in the switch process, first to define the S-channel and second to define the VSI within this S-channel.

Contrary to other Port Extenders, the MAC address is never stripped and it is always forwarded to the host.

The algorithm used by an M-comp is described in Section 7.4.8.1.

7.4.4.2.3.2 Transmit Flow

The S-comp only transfers packets from the S-channels to an Ethernet link or vice versa. It does not replicate packets or forward packets between virtual ports or physical ports.

No tag is added to the transmitted packet.

7.4.4.2.4 Port Virtualizer Control Ports

The Port Virtualizer may forward control packets to the associated control port.

Port Virtualizer control packets are untagged LLDP or enhanced LLDP packets combined with specific MAC addresses as described in Section 7.4.3. Each control port may require forwarding of specific control packets forwarding using the *Add Control Packet Filter* admin command.

In MFP mode, this control port is owned by the EMP. In SFP mode, it is may be owned by the PF driver or by the internal Firmware.

7.4.4.2.5 Port Virtualizer Flow Diagram

The following describes the high level flow used to define the switch ID associated with a packet. The details can be found in the algorithm sections (Section 7.4.8).

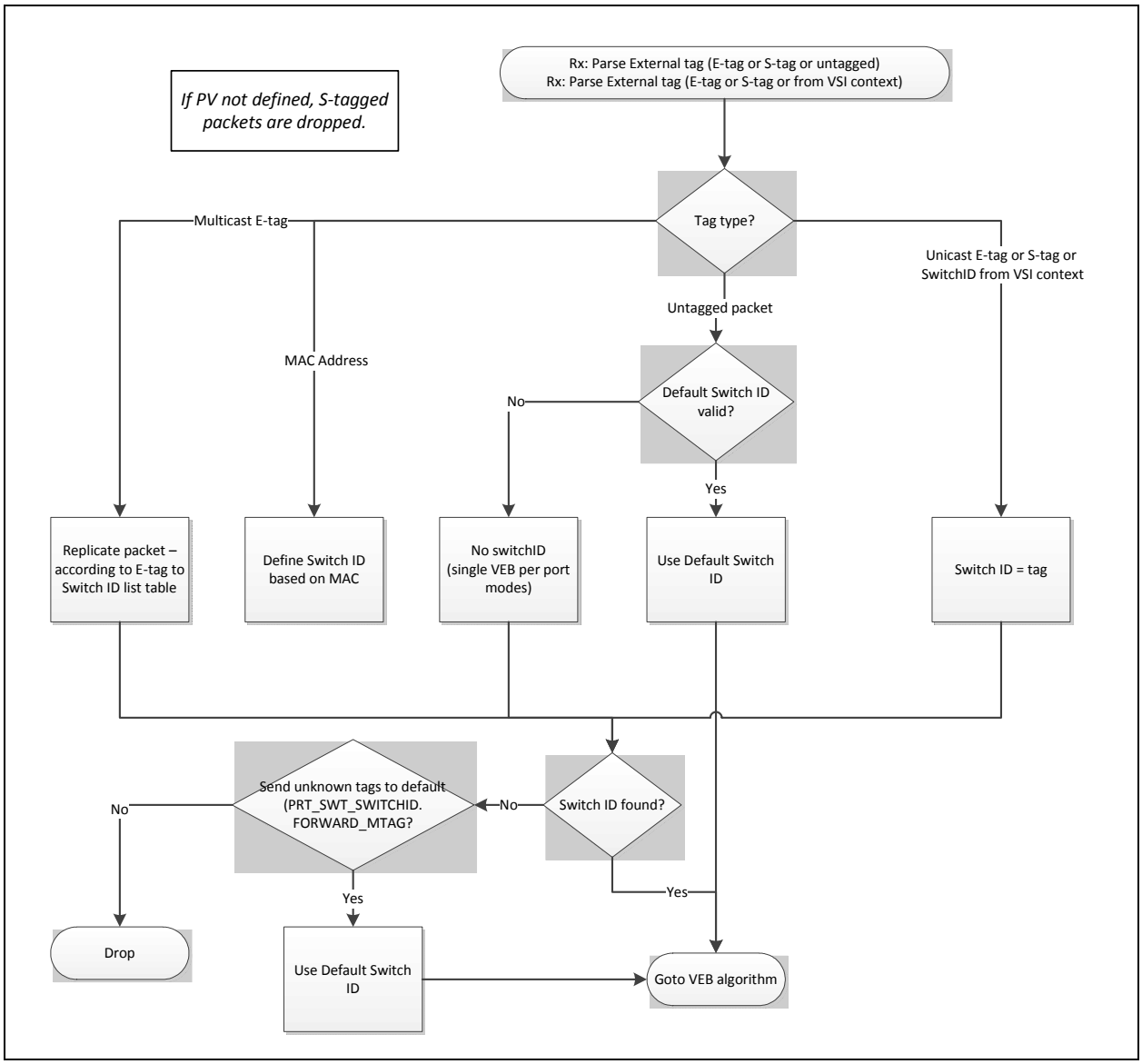


Figure 7-37. Port Virtualizer Flow

7.4.4.3 L2 Filters

A VSI directly connected to a physical port, to a port extender or to an S-comp, still needs the ability to filter incoming traffic as done in traditional NICs. The L2 filtering element provides this ability. An L2 filtering element is relevant only for receive or sideband traffic and does not apply to transmit traffic. An L2 element has no control port and is controlled directly by the function.

An L2 filter element is not represented as part of the topology and is implicitly created by adding L2 filters to existing VSIs.

This section describes L2 filters used when no VEB or VEPA element is initiated. If a VEB or VEPA element is added, the L2 filtering is done as part of the VEB forwarding as described in Section 7.4.4.4 and following sections.

An L2 filtering element includes a set of filters used to define whether a packet is received by the function or is dropped. There are two type of filters: exclusive filters that identify the packet as exclusive to this function and non exclusive filters that may pass also packets that may be forwarded to other ports or functions.

Non exclusive traffic may be further defined as perfect or imperfect, where perfect means that a packet that passed these filters was requested by the host, and imperfect filters may pass packets the host did not request.

In addition, a packet may be filtered out if it is larger than the maximum size defined by the port.

Note: Even if a packet passed the length filter of the port, it may yet be filtered by the length limitation of the receive queues.

An L2 filter is applicable only to receive traffic and does not refer to transmit traffic. Thus, if there is only an L2 filtering element, transmit traffic is forwarded to the network. A packet sent by this connection will be loopback to the sending VSI.

Note: The cloud filtering includes an L2 filter as part of its VEB or VEPA. This L2 filter.

7.4.4.3.1 Exclusive Filters

The following exclusive filters are available as part of an L2 filtering element:

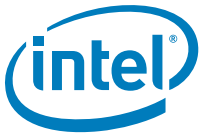
- {MAC:VLAN} Unicast table: Used to forward to a port packets matching both the MAC and VLAN pair. A VLAN value of zero includes untagged packets. Only unicast addresses are considered as exclusive
- {MAC} Unicast table: Used to forward to a port packets matching MAC addresses ignoring the VLAN tag.
- {VLAN} table: Used to define the VLAN to which the port belongs. A VLAN value of zero indicates the port can receive untagged packets.
- {Ethertype} table: Used to forward to a port packets matching an Ethertype ignoring the MAC address and VLAN tag.

7.4.4.3.2 Non Exclusive Filters

The following non exclusive perfect filters are available as part of an L2 filtering element:

- {MAC:VLAN} Multicast table: Used to forward to a port packets matching both the MAC and VLAN pair. A VLAN value of zero includes untagged packets. Only unicast addresses are considered as exclusive.
- {MAC} Multicast table: Used to forward to a port packets matching MAC addresses ignoring the VLAN tag.
- {UPE}: Promiscuous Unicast
- {MPE}: Promiscuous multicast
- {BAM}: Accept broadcast packets

The following non exclusive imperfect filters are available as part of an L2 filtering element:



- {HashMAC, VLAN, Address Type}: Used to accept packets whose Multicast/Unicast MAC address match a given MAC address and their VLAN match the VLAN tag in the pair. A VLAN value of zero includes untagged packets.
- {HashMAC, Address Type}: Used to accept packets whose Multicast/Unicast MAC address match a given MAC address.

The algorithm used by an L2 filter is described in Section 7.4.8.3.

7.4.4.4 Virtual Ethernet Bridge (VEB)

A VEB is used to switch packets between a set of virtual ports, a Physical port or an S-channel and a control port. The VEB behaves like a managed 802.1d transparent switch. All the configuration of the switch is done by a software agent. There are no auto configuration capabilities in the VEB (for example, there is no auto-learning of MAC addresses).

A VEB can forward packets received from any of its ports to a list of ports (including, optionally the originating port. The VEB uses the rules described in Section 7.4.4.4.2 to define the egress ports of each packet.

A packet is associated with a VEB using the SwitchID associated to it. A switchID can be either {0,S-tag} or a switch ID based on the SwitchID VSI parameter or the Port SWID parameter (*PRT_SWT_SWITCHID.SWID*).

There can be up to 16 VEBs or Port Aggregators in the XL710. They can be assigned dynamically to Physical functions. All the virtual ports associated with a VEB must belong to a single PF or its VFs.

Note: A VEB can be added only by a PF and can not be added by a VF or the EMP.

A VEB can be created using the *Add VEB* admin command described in Section 7.4.9.5.6.1.

7.4.4.4.1 VEB Parameters

A VEB is defined by the following parameters:

1. A control port that will receive link local packets (e.g. 802.1X packets or LLDP packets).
2. An optional default port that may receive packets received from the physical port and not forwarded to any virtual port.
3. Definitions of local and Private VLANs.
   a. Local VLANs: A list of VLAN tags defined as local that do not include the physical port.
   b. Private VLANs: A list of VLAN tags defined as private. For each private VLAN tag, define the promiscuous ports list, the community ports list, and the isolated ports list.

Note: More details on special VLANs can be found in Section 7.4.6.1.4.

4. A set of VSIs - Each VSI is defined by the following parameters:
   a. The connected PCIe function.
   b. A transmit enable and receive enable.
   c. A set of forwarding tables and parameters rules used to forward packets to the port as described in Section 7.4.4.4.2.1.
   d. Security features
   — Port based VLAN insertion - see Section 7.4.6.1.3 for more details.
   — Anti spoofing enables - see Section 7.4.6.1.2 for more details.

e. Optionally, a maximal and a minimal rate.

The full list of VSI parameters is described in Section 7.4.5.2.5.

7.4.4.4.2 VEB Switching Rules

The following parameters are used to define which egress ports (VSIs and LAN) will receive a packet received by the VEB.

7.4.4.4.2.1 Forwarding Tables and Parameters

- {MAC, VLAN} table: Used to forward to VSI(s) packets matching both the MAC and VLAN pair.
- {MAC} table: Used to forward to a VSI(s) packets matching MAC addresses ignoring the VLAN tag.
- {HashMAC, packet type}: Used to forward to port(s) packets whose Multicast or Unicast MAC address match a given MAC address. Packet Type is unicast or multicast.
- {HashMAC, VLAN, Packet type}: Used to forward to port(s) packets whose Multicast or Unicast MAC address match a given MAC address and their VLAN match the VLAN tag in the pair. Packet Type is unicast or multicast.
- {Ethertype} table: Used to forward to port(s) packets matching an Ethertype ignoring the MAC address (usually used for the control VSI).
- {MAC, Ethertype} table: Used to forward to port(s) packets whose MAC address match a given MAC address and Ethertype match an Ethertype (usually used for the control VSI).
- {VLAN} table: Used to define the VLANs to which a VSI(s) belongs for egress and ingress checks. This table can point to two lists - one list that is used for ingress VLAN filtering and one used for egress VLAN filtering.

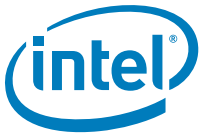
Note: In all above tables, a VLAN value of zero includes untagged packets

Note: A port in these tables means either a VSI or the LAN port for transmit packets and a VSI for receive packets.

- Promiscuous modes per VSI:
  - {UPE}: Used to forward to a VSI, all unicast packets.
  - {MPE}: Used to forward to a VSI, all multicast packets.
  - {BAM}: Used to forward to a VSI, broadcast packets.
- {VPE}: Used to forward to a VSI, packets with any VLAN tag.
- Loopback rules:
  - {ALLOWLOOPBACK} Should this port be allowed to send packets to other virtual ports?
  - {PruneEnable} Should this port receive packets it sent? This mode is useful to allow offload of a software switch connected to this virtual port.
- Anti Spoofing parameters. These parameters defines if a packet should be allowed to enter the switch.
  - MAC Anti spoofing enable
  - VLAN anti spoofing enable
  - FCoE Enable - allows forwarding of FCoE packets

The following commands are used to program these tables:

- Add MAC, VLAN pair (Section 7.4.9.5.8.1).
- Remove MAC, VLAN pair (Section 7.4.9.5.8.2).



- Add VLAN (Section 7.4.9.5.8.3).
- Remove VLAN (Section 7.4.9.5.8.4).
- Set VSI promiscuous modes (Section 7.4.9.5.8.5).
- Add Control Packet Filter (Section 7.4.9.5.8.9).
- Remove Control Packet Filter (Section 7.4.9.5.8.10).

Note: The *Add MAC, VLAN pair* command allows filtering based on a MAC address (any VLAN) or on a MAC, VLAN pair. The *Add VLAN* command is used to define the VLAN membership of ports and is not used to add destination VSIs. Using this method, the VEB can allow MAC based filtering and promiscuous modes within specific VLANs.

7.4.4.4.3 VEB Flow

Figure 7-38 describes the generic flow of the VEB element. The exact algorithm implementing this flow is described in Section 7.4.8.4.

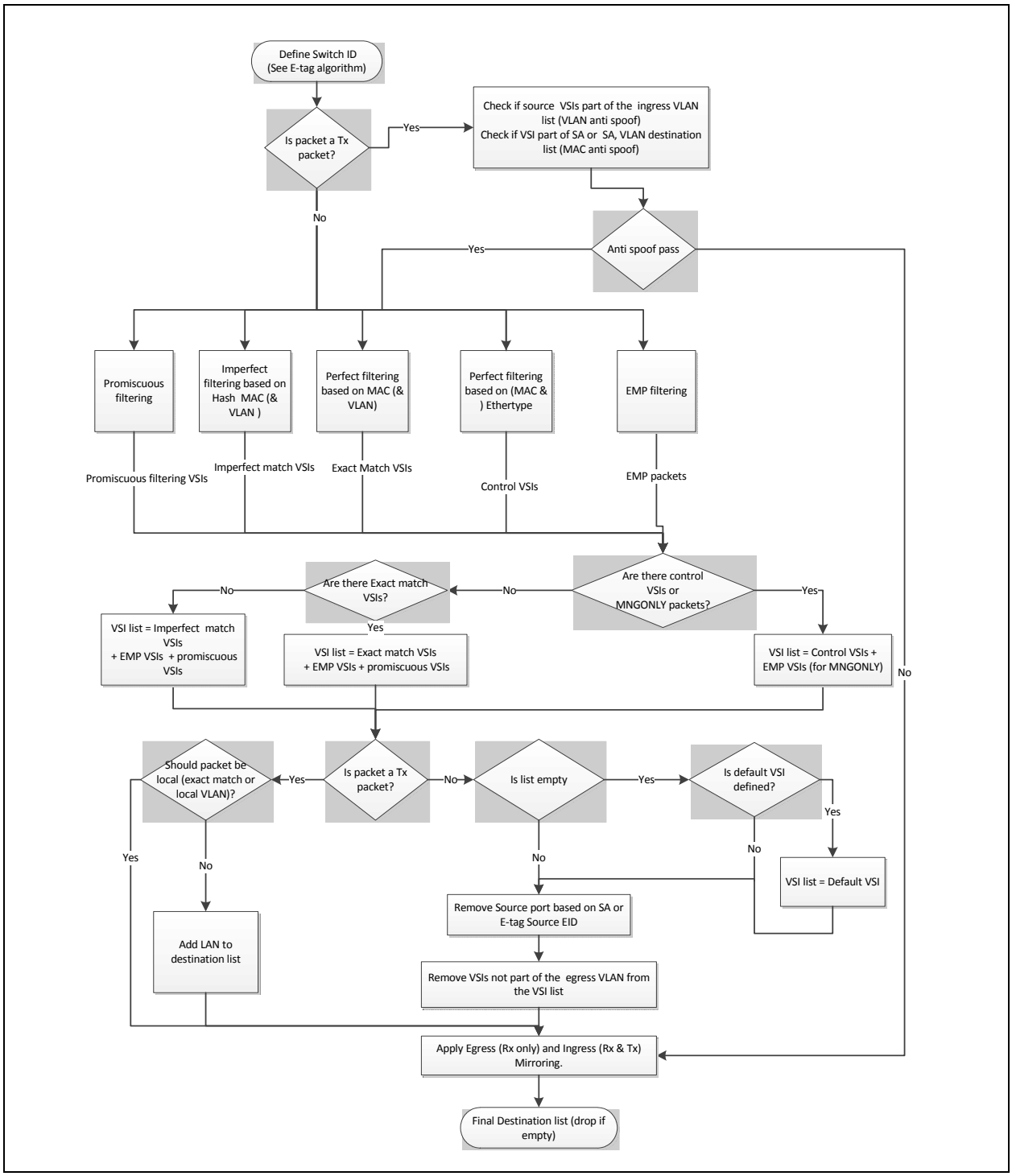
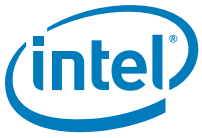


Figure 7-38. VEB flow diagram



7.4.4.5 Virtual Ethernet Bridge with Cloud Support (Cloud VEB)

A VEB is used to switch packets between a set of virtual ports, a Physical port or an S-channel and a control port. The VEB behaves like a managed 802.1d transparent switch. All the configuration of the switch is done by a software agent. There are no auto configuration capabilities in the VEB (for example, there is no auto-learning of MAC addresses).

A VEB can forward packets received from any of its ports to a list of ports (including, optionally the originating port. The VEB uses the rules described in Section 7.4.4.5.2 to define the egress ports of each packet.

A packet is associated with a VEB using the SwitchID associated to it. A switchID can be either {0,S-tag} or a switch ID based on the SwitchID VSI parameter or the Port SWID parameter (*PRT_SWT_SWITCHID.SWID*).

There can be up to 16 VEBs or Port Aggregators in the XL710. They can be assigned dynamically to Physical functions. All the virtual ports associated with a VEB must belong to a single PF or its VFs.

Note: A VEB can be added only by a PF and can not be added by a VF or the EMP.

A VEB can be created using the *Add VEB* admin command described in Section 7.4.9.5.6.1.

A cloud VEB can also be configured to VEPA mode. The exact algorithm used by a cloud VEB is described in Section 7.4.8.6.

The usage of cloud VEB is similar to the usage or a regular VEB except from the following points:

1. An adequate cloud NVM image should be used (Cloud or UDP cloud).
2. When working in cloud mode, only a single VEB can be instantiated per port. There is no support for floating VEBs or for port virtualizers.
3. The *Add Cloud filters* (Section 7.4.9.5.8.11) and *Delete Cloud filters* (Section 7.4.9.5.8.12) commands are available to add cloud based filters. These filters are available only for VSIs defined as Cloud VSIs in the *Add VSI* command (Flags.Cloud VSI = 1).

7.4.4.5.1 VEB with Cloud support Parameters

A VEB is defined by the following parameters:

1. A control port that will receive link local packets (e.g. 802.1X packets or LLDP packets).
2. An optional default port that may receive packets received from the physical port and not forwarded to any virtual port.
3. Definitions of local and Private VLANs.
   a. Local VLANs: A list of VLAN tags defined as local that do not include the physical port.
   b. Private VLANs: A list of VLAN tags defined as private. For each private VLAN tag, define the promiscuous ports list, the community ports list, and the isolated ports list.

Note: More details on special VLANs can be found in Section 7.4.6.1.4.

4. A set of VSIs - Each VSI is defined by the following parameters:
   a. The connected PCIe function.
   b. A transmit enable and receive enable.
   c. A set of forwarding tables and parameters rules used to forward packets to the port as described in Section 7.4.4.5.2.

d. Security features
   — Port based VLAN insertion - see Section 7.4.6.1.3 for more details.
   — Anti spoofing enables - see Section 7.4.6.1.2 for more details.
e. Optionally, a maximal and a minimal rate.

The full list of VSI parameters is described in Section 7.4.5.2.5.

7.4.4.5.2 Cloud VEB Switching Rules

The following parameters are used to define which egress ports (VSIs and LAN) will receive a packet received by the VEB.

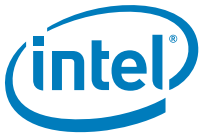
7.4.4.5.2.1 Forwarding Tables and Parameters

- Priority 1 filters (Control filters):
  — {Ethertype} table: Used to forward to port(s) packets matching an Ethertype ignoring the MAC address (usually used for the control VSI).
  — {MAC, Ethertype} table: Used to forward to port(s) packets whose MAC address match a given MAC address and Ethertype match an Ethertype (usually used for the control VSI).
- Priority 2 filters (Cloud Filters):
  — Outer IP (for GRE packets or IP in IP packets)
  — {Inner MAC, Inner VLAN} (for NVGRE, VXLANor Geneve packets)
  — {Inner MAC, Inner VLAN, Tenant ID} (for NVGRE, VXLANor Geneve packets)
  — {Inner MAC, Tenant ID} (NVGRE packet or VXLANGeneve packets).
  — Inner MAC filter
  — {Outer MAC, Tenant ID, Inner MAC} filter
  — Inner IP filter
- Priority 2 filters (Cloud Filters) available with “Cloud” NVM image:
  — Outer IP (for GRE packets or IP in IP packets)
  — Outer IP, GRE Key (for GRE packets)
  — Inner MAC, Inner VLAN, GRE Key (for NVGRE packets)
  — Inner MAC, Inner VLAN, Outer IP (for NVGRE packets)
  — Inner MAC, Inner VLAN (for NVGRE packets)
  — Inner MAC filter
- Priority 2 filters (Cloud Filters) available with “UDP Cloud” NVM image:
  — Inner MAC, Inner VLAN (for NVGRE or VXLAN packets)
  — {Inner MAC, VNI} (VXLAN packets).
  — {Inner MAC, Inner VLAN, VNI} (for VXLAN packets)
  — Inner MAC filter

Note: The Network Key is extracted from the GRE or UDP headers in MAC in GRE or MAC in UDP packets.

GRE Key is a 4 byte field enabled by the 'k' flag in the GRE header. The key is extracted only if the GRE header includes a key as indicated by active 'K' flag in the header.

The UDP port used for MAC in UDP tunneling is defined by a shared 16 entry table with Teredo port numbers. The key is extracted if the UDP port number contains a key as defined by the “UDP Protocol Index” which is programmed by the *Add Teredo Port Command*.



- Priority 3 filters (L2 Filters):
  - {MAC, VLAN} table: Used to forward to VSI(s) packets matching both the MAC and VLAN pair.
  - {MAC} table: Used to forward to a VSI(s) packets matching MAC addresses ignoring the VLAN tag.
  - {HashMAC, packet type}: Used to forward to port(s) packets whose Multicast or Unicast MAC address match a given MAC address. Packet Type is unicast or multicast.
  - {HashMAC, VLAN, Packet type}: Used to forward to port(s) packets whose Multicast or Unicast MAC address match a given MAC address and their VLAN match the VLAN tag in the pair. Packet Type is unicast or multicast.
  - {VLAN} table: Used to define the VLANs to which a VSI(s) belongs for egress and ingress checks. This table can point to two lists - one list that is used for ingress VLAN filtering and one used for egress VLAN filtering.

Note: In all above tables, a VLAN value of zero includes untagged packets

Note: A port in these tables means either a VSI or the LAN port for transmit packets and a VSI for receive packets.

- Promiscuous modes per VSI:
  - {UPE}: Used to forward to a VSI, all unicast packets.
  - {MPE}: Used to forward to a VSI, all multicast packets.
  - {BAM}: Used to forward to a VSI, broadcast packets.
- {VPE}: Used to forward to a VSI, packets with any VLAN tag.
- Loopback rules:
  - {ALLOWLOOPBACK} Should this port be allowed to send packets to other virtual ports?
  - {PruneEnable} Should this port receive packets it sent? This mode is useful to allow offload of a software switch connected to this virtual port.
- Anti Spoofing parameters. These parameters defines if a packet should be allowed to enter the switch.
  - MAC Anti spoofing enable
  - VLAN anti spoofing enable
  - FCoE Enable - allows forwarding of FCoE packets

The forwarding filters are divided to groups. Those filters that are needed for tunneled packet formats and those ones that are shared for all packet formats (based on the outer MAC,VLAN). Filters that match tunneled packet formats take precedence on those filters that matches all packet formats (as illustrated in Figure 7-39 below).

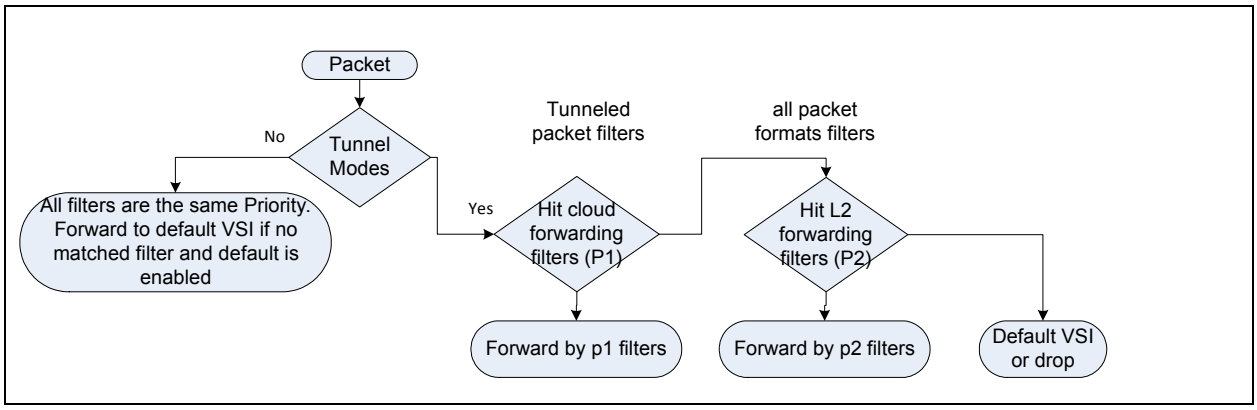


Figure 7-39. Cloud Forwarding Filters Priority

The *Add Control Packet Filter* (Section 7.4.9.5.8.9) and *Remove Control Packet Filter* (Section 7.4.9.5.8.10) commands are used to add or remove control filters settings. Each control filter can point to a single VSI. There is no support for packet replication based on control filters. An error is returned if an existing control filter is added to point to another VSI.

The *Add Cloud filters* (Section 7.4.9.5.8.11) and *Delete Cloud filters* (Section 7.4.9.5.8.12) commands are used to control cloud specific filters settings Each cloud filter can point to a single VSI. There is no support for packet replication based on cloud filters.In addition to those filters, in order for a packet to be received by one of the VSIs, it must also pass the {L2 MAC} filter. In order to enable L2 filtering, the *Flags.Enable L2 filtering* bit in the *Add VEB* command should be set when creating the cloud VEB.

- The *Add MAC, VLAN pair* (Section), *Remove MAC, VLAN pair* (Section 7.4.9.5.8.2), *Add VLAN* (Section 7.4.9.5.8.3), and *Remove VLAN* (Section 7.4.9.5.8.4) commands are used to set regular MAC/VLAN filters.

Note: The *Add MAC, VLAN pair* command allows filtering based on a MAC address (any VLAN) or on a MAC, VLAN pair. The *Add VLAN* command is used to define the VLAN membership of ports and is not used to add destination VSIs. Using this method, the VEB can allow MAC based filtering and promiscuous modes within specific VLANs.

- The *Set VSI promiscuous modes* (Section 7.4.9.5.8.5) command is used to set promiscuous filters
- The *Add Mirror Rule* (Section 7.4.9.5.9.1) and *Delete Mirror Rule* (Section 7.4.9.5.9.2) commands are used to control mirror rules

Note: It is assumed that manageability packets and control port packets will not be encapsulated.

7.4.4.5.3 Cloud VEB Flow

Figure 7-40 describes the generic flow of the cloud VEB element. The exact algorithm implementing this flow is described in Section 7.4.8.6.

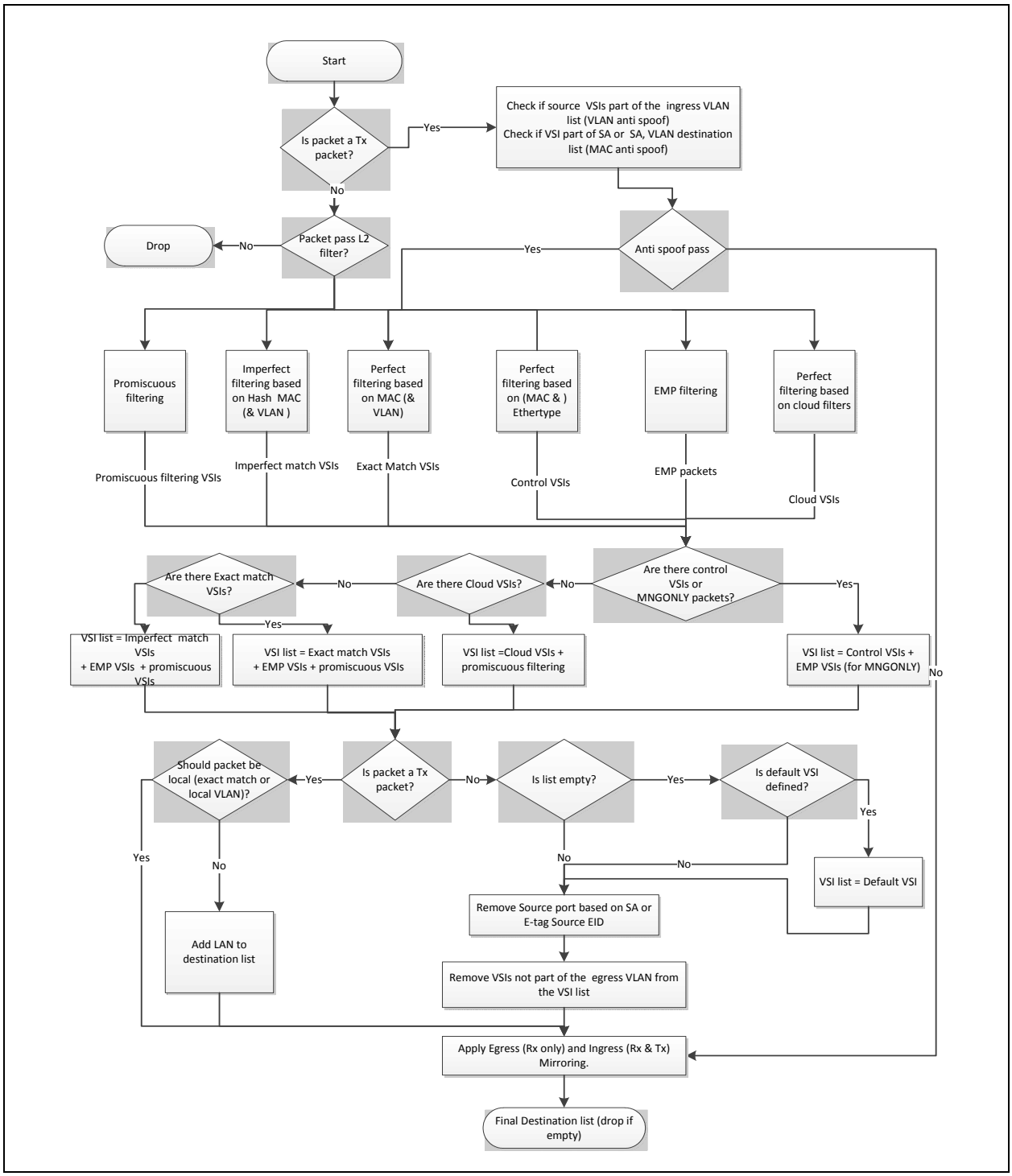


Figure 7-40. Cloud VEB flow diagram

7.4.4.6 Port Aggregator

The Port Aggregators (a.k.a VEPA) usage model is described in Section 7.4.2.2.1.

7.4.4.6.1 Port Aggregator Switching Rules

The behavior of a Port Aggregator is the same as the behavior of a VEB. The only difference is that in a port aggregator, Loopback should be disabled for all virtual ports (ALLOWLOOPBACK = 0b).

A VEPA can be created using the *Add VEB* admin command described in Section 7.4.9.5.6.1 and by disabling loopback in the VSIs connected to this VEB.

7.4.4.6.2 Port Aggregator Flow

Figure 7-41 describes the generic flow of the VEPA element. The exact algorithm implementing this flow is described in Section 7.4.8.6.

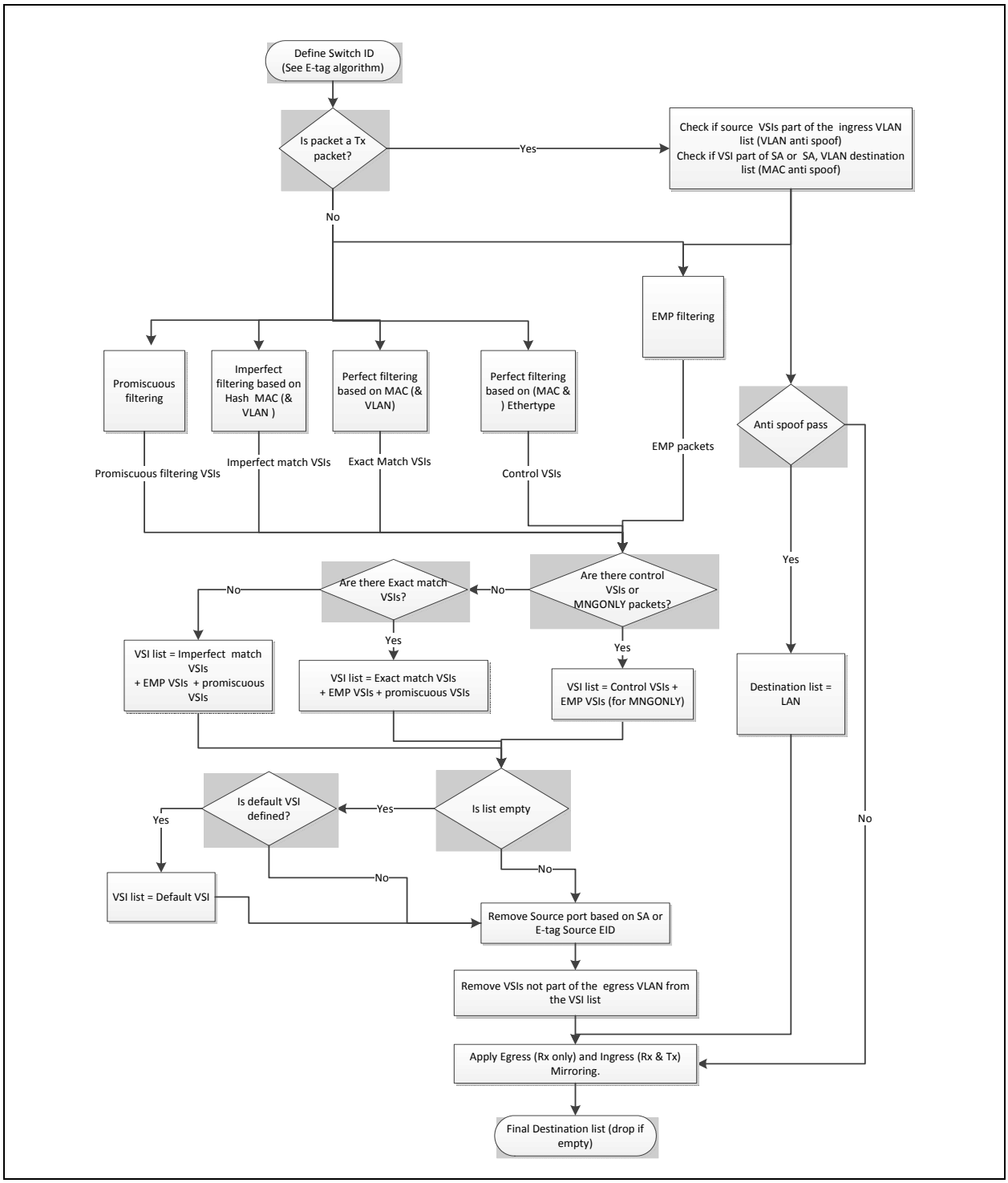


Figure 7-41. VEPA flow diagram

— Storm control. Last setting wins.

7.4.4.7 Manageability Sideband Switching

The network interface may be shared between the host and a sideband manageability interface. As described in Section 10.1.2, the sideband interface can be either over a 100 Mbps Ethernet interface using NC-SI, over SMBus using a legacy pass through mode or NC-SI over MCTP transport, or over PCIe using NC-SI over MCTP transport.

The specific filters used to define which packets are forwarded to the sideband interface are described in Section 10.3. The current section only describes the relationships between the sideband and other switching elements.

7.4.4.7.1 Network to BMC Filtering

The relationships between the sideband filtering and other switching elements follow:

- If no switching elements are defined (i.e. no VEB or Port Virtualizer), the sideband decision filters described in Section 10.3 are applied to the network traffic. Packets which pass these filters are forwarded to the sideband interface.
- If an S-comp is defined on the port, the sideband may be defined as part of one of the channels in the S-comp. In this case, the sideband decision filters are applied only to the traffic with the matching S-tag. The same S-tag is added to packets sent from the sideband interface.
- The sideband switching is never part of a VEB, port aggregator or Port Virtualizer and is done in parallel to the forwarding decisions of these elements.

7.4.4.7.2 OS to BMC traffic filtering

The following rules are used to decide whether OS traffic should be sent to the sideband interface:

- If no switching elements are defined, for each function associated with a given port, the sideband decision filters associated to this port and applicable to host traffic are applied to the function traffic. Packets that passes these filters are forwarded to the sideband interface. If the sideband interface defines these packets as exclusive (*PRT_MNG_MNGONLY* is set for this filter), the packets are not sent to the network. Otherwise, they are sent both to the sideband interface and to the network.
- If an S-comp is defined on the port, the sideband may be defined as part of one of the channels in the S-comp. In this case, the sideband decision filters are applied only to host traffic with the matching S-tag. Traffic from the host to the BMC is sent to the BMC without the S-tag .

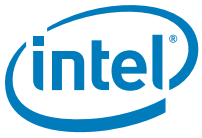
Note: It is assumed that traffic sent from an S-channel different than the S-channel of the BMC is forwarded by an external switch.

- If a VEB, port aggregator or Port Virtualizer is defined, they will not receive packets exclusively sent to the sideband interface.

7.4.4.7.3 BMC to OS Traffic Filtering

The following rules are used to decide whether traffic received from the sideband interface should be sent to the host:

- If no switching elements are defined, the port L2 filters are applied to the sideband traffic. Packets that passes these filters are forwarded to the host. Unicast packets that pass exclusive L2 filtering as defined in Section 7.4.4.3.1 are sent only to the host. Multicast or broadcast packets and unicast



packets that pass non exclusive filtering as defined in Section 7.4.4.3.2, are sent both to the host and to the network. Other packets are sent only to the network.

- If a VEB or port aggregator are defined, the switching algorithms of the VEB or port aggregator are applied to the sideband traffic. Packets which pass these filters are forwarded to the host. Unicast packets that pass perfect L2 filtering as defined in Section 7.4.4.3.2 are sent only to the host. Multicast or broadcast packets and unicast packets that pass imperfect filtering as defined in Section 7.4.4.3.2, are sent both to the host and to the network. Other packets are sent only to the network.
- If an S-comp is defined on the port, the sideband may be defined as part of one of the channels in the S-comp. In this case, the switching elements relevant to this channel are applied to the sideband traffic. Traffic from the BMC to the host will have no S-tag appended.

Note: It is assumed that traffic sent from the S-channel of the BMC to a function on a different S-channel is forwarded by an external switch.

- If a Port Virtualizer is defined on the port or S-channel of the sideband interface, it will not receive traffic from the BMC directly. It may still receive such traffic through an external switch.

7.4.4.8 Out of Band Filtering Operation

Part of the type of traffics are expected to be operational also in Sx (equivalent to D3 or Dr state of the device). The following types of network inbound traffic are included in this category:

- Wake-up per PF
- Manageability traffic (two channels)
- LLDP traffic (untagged only).

In order to support this traffic, a separate simplified switch mechanism (PF classifier) is provided which is independent of the regular switch and is used to forward packets to these specific locations.

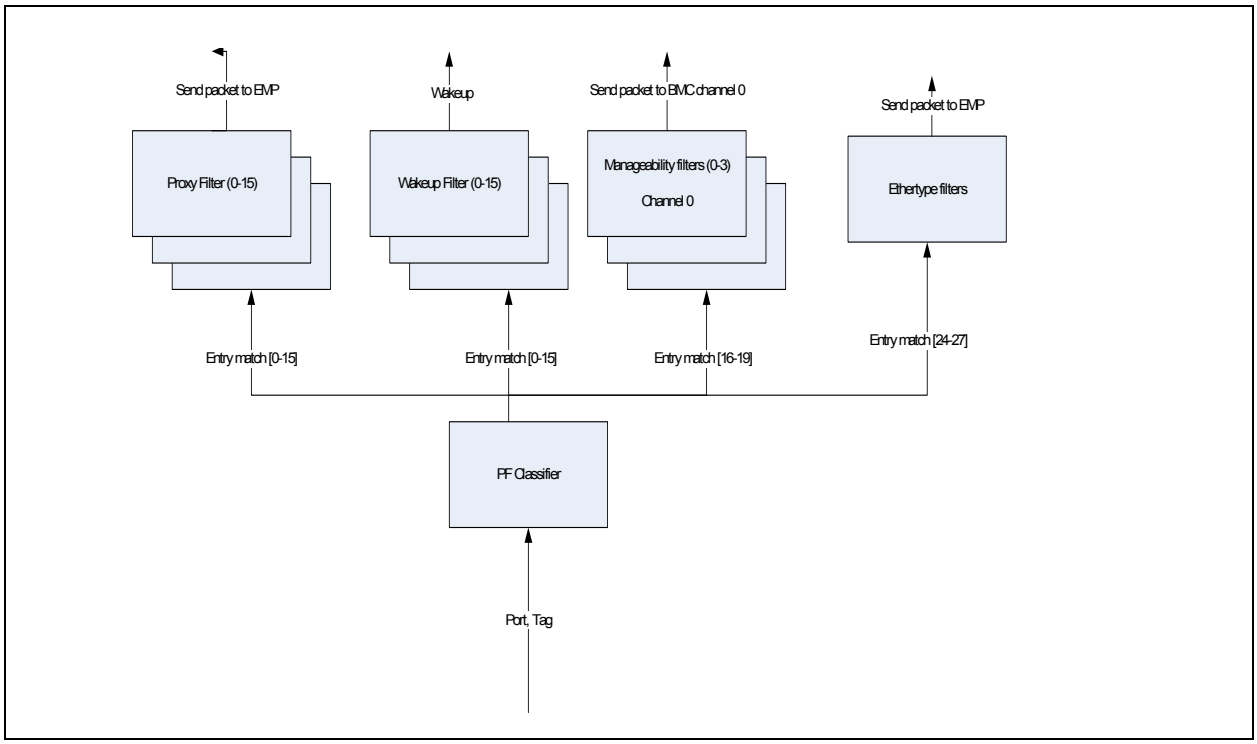


Figure 7-42. Out of Band Filtering Operation

7.4.5 Ports

7.4.5.1 Physical Ports

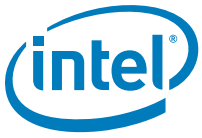
Physical ports are considered as uplink ports and are not part of the L2 forwarding table. In each switch, there can be one physical port only. All the traffic that is not forwarded to one of the other ports is sent to the uplink egress port. In addition, multicast or broadcast traffic not received from the uplink ingress port is always forwarded to the uplink port, unless it is part of a local VLAN or the switch has no connection to an uplink. Local VLANs are defined in Section 7.4.6.1.5.

7.4.5.1.1 Ethernet Ports

Each of the Ethernet ports (either 10gbps or 40gbps) can be assigned as an uplink port of a switch if not connected to an S-component. In this case, all the traffic received from this ethernet port is handed to the switch.

7.4.5.2 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 384 VSIs in the XL710. All the VSIs are equivalent.

At init time, each physical function is assigned a VSI. A Default VSI can also be used to provide the VMDq1 or control VSIs functionality described below.

Other VSIs can be assigned to the physical function or to Virtual functions and used for the following purposes:

1. VMDq2
2. VMDq1
3. Control ports
4. Mirroring
5. FCoE

In addition, EMP VSIs can also be defined. The EMP VSIs are used for:

1. Pass through traffic
2. Control ports

The different types of VSIs and their attributes are described in the following sections.

Mapping of queues to VSIs is described in Section 7.4.5.2.3.

7.4.5.2.1 Host VSIs Types

7.4.5.2.1.1 VMDq2 VSI

XL710 supports offloading of a VMM software switch. In this mode, the switching between VMs is done using a VEB element or an Port Virtualizer element. However, all or part of the ports may not be directly connected to the VEB/Port Virtualizer as SR-IOV functions (VFs) as they may be hidden behind the VMM or a service VM. These VMs can be represented to the VEB as groups of queues in a single physical function. There can be up to 256 VSIs for VMDq2.

The only difference between a VMDq2 VSI and a regular VSI associated with the PF is the ability to apply the VM reset flow described in Section 4.1.2.6.

Note: A VMDq2 VSI should not be defined as FCoE (Section 7.4.5.2.1.5).

7.4.5.2.1.2 VMDq1- cascaded VEB/S-comp VSI

A VSI may be used for VMDq1 offload. There are two types of VMDq1 offloads:

- A cascaded VEB.
- A cascaded S-comp.

In a cascaded VEB, a software switch is tied to this VSI and uses the MAC, VLAN pairs to queue packets to the right receive queue.

In a cascaded S-comp, the VMDq1 VSI is used to offload a Software based S-comp. A cascaded S-comp VSI is created by setting the *Cascaded Port Virtualizer* flag in the *Add VSI* command. When an S-channel is used to convey multiple S-tags, it should be connected directly to a VSI; VEB or VEPA elements should not be connected on top of it. Filtering of packets to be sent to a cascaded S-comp VSI should be based only on S-tag and not on MAC, VLAN or other filters.

7.4.5.2.1.3 Control VSI

A single PCI function may be used to control one or multiple switching elements. For example, a single function may be used to control a Port Virtualizer and a VEB element associated with the function. As we need to separate the control packets of different elements to different queues in receive and identify the switch element that should process the packets in transmit, a separate control VSI should be defined for each element.

A control VSI may receive part or all the link local traffic received by the controlled element. So for example a Port Virtualizer control VSI should receive LLDP untagged packets. A VEB control VSI behind an Port Virtualizer will get LLDP packets tagged with the S-tag associated with this VEB. As the control packet's traffic may be handled by a different control port, each control port driver should request forwarding of the relevant packets using the *Add Control Packet Filter* admin command.

Any VSI can be used as a control port as long as the adequate packets are routed to it. A Control VSI should have the "Allow Destination Override" flag set to allow it to bypass the switch when sending packets.

At init time, the control VSI of the MAC is assigned to the EMP. If at a later stage, one of the physical functions decides to take ownership of this control port, it should assign one of its VSI as the control port of the MAC. The EMP should be notified of the change using *Stop LLDP Agent* command and should disconnect the EMP control port. The physical function may elect to add a control port in addition to the EMP control port.If a control port is added to a VEB or a VEPA (either connected to the LAN via a Port Virtualizer or floating), it will not impact the connectivity MAC control port to the EMP. A control port of a VEB directly connected to the MAC may use the MAC control port or may use a separate VSI. After a port is defined, the owner of the control port (Firmware or Driver) should set the right filters using the *Add Control Packet Filter* admin command.

A typical configuration may be for example:

- Untagged (no S-tag) LLDP packets with a Nearest Bridge (01-80-C2-00-00-0E) address are forwarded to the control port of the MAC or of a switch element directly connected to the MAC (Port Virtualizer or VEB) on the EMP. These packets includes DCBx or EEE TLVs.
- Untagged (no S-tag) LLDP packets with a Nearest Non TPMR (01-80-C2-00-00-03) or Nearest Customer Bridge (01-80-C2-00-00-00) addresses are forwarded to the control port of the switch element directly connected to the MAC (Port Virtualizer or VEB) on the host. These packets includes CDCP TLVs.
- Untagged ECP packets are forwarded to the control port of the switch element directly connected to the MAC (Port Virtualizer or VEB) on the host. These packets includes VDP TLVs.
- S-Tagged LLDP packets with a Nearest Customer Bridge (01-80-C2-00-00-00) address are forwarded to the control port of a VEB connected to this S-channel.

Note: A control port can also be used as a default port.

7.4.5.2.1.3.1 Transmission of Packets from a Control VSI.

A control VSI may need to send directed multicast packets, for example sending an LLDP packet with a link local multicast address to the link partner or to one of the VSIs. According to the regular forwarding rules of the switch, such packets are forwarded back to the control port or dropped. To overcome this, the control VSI should set the *SWTCH* field and optional the *DEST_VSI* field in the transmit context descriptor to indicate the desired destination of the packet. See Section 8.4.2.2.1 for details.

7.4.5.2.1.4 Mirroring VSI

A VSI may be used to direct mirror traffic as defined by the mirroring rules in the associated VEB/VEPA. In this case, such a VSI is not expected to receive any traffic apart from the mirrored traffic. Any VSI in the VEB can be set as a mirror port.

A mirror VSI should be set to promiscuous VLAN mode to enable reception packets from any VLAN.

7.4.5.2.1.5 FCoE VSI

In some systems, the behavior of FCoE traffic may differ from the behavior of the LAN traffic in a way that can not be modeled by different TCs. For example, FCoE may require a different PVID. Thus a separate VSI may be assigned to FCoE traffic.

7.4.5.2.1.6 Default VSI

In each port, or each VEB one VSI can be defined as the default VSI. A port default VSI is available only if a Port Virtualizer is instantiated on this port, otherwise a VEB default port should be used.

A Port default VSI is added by defining the Connected VSI port type in the *Add Port Virtualizer* command to "Default".

A VEB default VSI is added by defining the Downlink VSI port type in the *Add VEB* command to "Default".

If a default port is defined, all the received traffic within the Port/VEB not matching any forwarding rule is sent to the default port, otherwise it is dropped.

Transmit traffic is not impacted by the default port. Unmatched transmit traffic is sent to the LAN.

A Default port is subject to VLAN filtering rules. In order to allow all the unmatched traffic to be default VSI, it should be set to promiscuous VLAN using the *Set VSI Promiscuous Modes* command.

7.4.5.2.2 EMP VSI Types

7.4.5.2.2.1 Pass Through VSI

A pass through VSI is used to send and receive traffic from an Out of Band manageability interface. The forwarding rules. If this VSI is located behind a Port Virtualizer, S-tag insertion and removal for packets sent to or received from the manageability interface is done by the device. VLAN tagging is done by the external BMC.

7.4.5.2.2.2 Control VSI

An EMP control VSI behaves as a host control VSI (Section 7.4.5.2.1.3). It sends and receive packets without S-tag and can send or receive VLAN-tagged or VLAN-untagged packets.

7.4.5.2.3 Mapping of Functions and Queues to VSIs.

The function to which each VSI belongs is indicated by the VSI_VSI2F register in the VSI context This register is initiated by the *Add VSI* admin command (Section 7.4.9.5.4.1).

Transmit queues are grouped in Queue Groups as described in Section 7.8.1.5. Each queue group is associated with a VSI, thus creating an association between transmit queues and VSIs. This association is used when applying the VSI policies to transmit traffic.

Receive queues are mapped to VSIs using the *VSILAN_QTABLE* registers as described in Section 8.2.1.2. These registers are also initiated by the *Add VSI* admin command.

Mapping can be either contiguous within the function queues or scattered. In case of scattered mapping, up to 16 queues can be associated with a VSI. Both receive and transmit queues should be associated within the queue set of the VF/PF to which this VSI is associated. So for VSIs associated to a VF, up to 16 queues can be associated with the VSI. For VSIs associated with a PF, all the queues of the PF can be associated with a VSI.

A VSI may choose to queue packets to different receive queues either using the RSS and traffic class information or using some other dedicated filters.

7.4.5.2.4 VSI Connections to the Switching Elements

Most VSIs can be connected to the following switching elements as regular ports:

- MAC, VEB, Port Virtualizer or port aggregator.
- Port Virtualizer.

7.4.5.2.5 VSI Context

The context of a VSI contains the following parameters.

Table 7-56. VSI context

Parameter	Description	Register.Field	Notes
Function pairing			
Function Type	Defines the owner this VSI can be either a PF, a VF or the EMP.	*VSI2_VSIVF.FUNCTIONTYPE*	00b: VF 01b: VM 10b: PF 11b: EMP
VF Function Number	The VF number of the function this VSI belongs to.	*VSI_VSI2F.VFVMNUMBER*	
PF function number	The PF number of the function this VSI belongs to.	*VSI_VSI2F.PFNUMBER*	This field should be set also for VFs.
VSI_ENABLE	Enables transmit and receive from VSI	VSI_VSI2F.VS_ENABLE	
Switching parameters			
SwitchID	Defines the Switch to which this VSI belongs. In case of a VSI connected to a VEB/VEPA/MAC, this should be the same as the uplink switch ID. In case of a VSI connected to a Port Virtualizer, this is the S-tag associated with the S-channel of this VSI.	*VSI_SRCSWCTRL.SWID[11:0], VSI_SRCSWCTRL.ISNSTAG, VSI_SRCSWCTRL.SWIDVALID*	The ISNSTAG should be set if the switchID is not an S-tag.
Enable VLAN Anti Spoof	Should we check the VLANs used by this VSI to send packets are legitimate.	*VSI_SRCSWCTRL.VLANAS*	

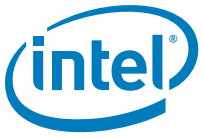


Table 7-56. VSI context

Parameter	Description	Register.Field	Notes
Enable MAC Anti Spoof	Should we check the SA or SA/VLAN used by this VSI to send packets is legitimate.	*VSI_SRCSWCTRL.MACAS*	
Allow destination override	Allows this VSI to set the destination of a packet.	*VSI_SRCSWCTRL.ALLOWDESTOVERRIDE*	
Enable Local Loopback	Allows forwarding to a VSI of packets sent from this VSI	*VSI_RXSWCTRL.PRUNEENABLE* (inverse logic).	
Enable Loopback	Allows forwarding of packets from this VSI to local VSIs.	*VSI_SRCSWCTRL.ENABLELLOOPBACK.*	
LAN Enable	Defines if the VSI is part of a VEB connected to the network	*VSI_SRCSWCTRL.LANENABLE.*	
Tag insertion & admission parameters			
Tag accept mode	Defines which tags to accept	*VSI_TAR.ACCEPTTAGGED* and *VSI_TAR.ACCEPTUNTAGGED.*	
Port based tag	Define the VLAN/S-tag or other tags to insert into packet	*VSI_TIR.PORT_TAG_ID*	
Port based tag insert	Defines whether to use the tags above.	*VSI_L2TAGSTXVALID.PORTBASEDTAGS*	
Tag strip policy.	Defines for each tag, if it should be left in packet, extracted to descriptor or removed.	*VSI_TSR.*	
Queue mapping			
Queue base	The first queue allocated to this VSI.	*VSILAN_QBASE.VSIBASE*	
Number of receive queues per TC	Defines for each TC how many receive queues are allocated	*VSIQF_TCREGION*	
Queue mapping	Maps 16 virtual queues to physical queues	*VSILAN_QTABLE* and *VSILAN_QBASE.VSIQTABLE_ENA*	
Classification Filters control			
FCoE forwarding	Enable FCoE filter context for the VSI. When this flag is cleared, the hardware does not look for matched FCoE filter.	VSIQF_CTL.FCOE_ENA	
DCB control			
Enabled UPs	Defines the user priorities used by this VSI	Implemented in scheduler.	

7.4.5.2.6 VSI Add and Remove Flows

Addition or removal of a VSI can be done only by the PF. The following sections describes the flow a PF driver should use to add or remove a VSI.

7.4.5.2.6.1 VSI Init Flow

In order to init a VSI, the software should use the *Add VSI* command to create the required VSI context. As part of this command, the queues allocated to the VSI should be specified. After this command is executed, the queues may be initiated as defined in sections 8.3.3.1.1 and 8.4.3.1.1.

7.4.5.2.6.2 VSI Disable Flow

VSIs can be disabled either individually or as part of their function. For example, when a VF id disabled, the VF reset will stop the VSIs associated with this VF. There are cases where a PF may need to disable a single VSI. The use case for this flow is when a VSI is used as a VMDq interface to a VM that is disabled or moved to another machine. In order to allow a quick disable of a VSI, the following flow should be used.

1. SW should stop scheduling packets to the transmit queues assigned to the VSI(s).
2. Stop the VSI traffic:
   a. For VMDQ VSIs, assert the VM reset using the *VSIGEN_RTRIG.VMSWR* bit and wait until the *VSIGEN_RSTAT.VMRD* bit clears. This step will stop the traffic in all the queues associated with this VSI.
   b. For VFs, assert the VF reset using the *VPGEN_VFRTRIG.VFSWR* bit and wait until the *VPGEN_VFRSTAT.VFRD* bit clears. This step will stop the traffic in all the queues associated with this VF.
3. Stop the relevant queues using the flows described in sections 8.3.3.1.2 and 8.4.3.1.2.
4. The PF 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.
5. Remove all MAC, VLAN filters pointing to this VSI/VF using Remove MAC,VLAN pair admin command.
6. Send a *Delete Element* admin command with the VSI SEID (for each VSI associated to the VF). This step will clear all the VSI context values. Remove the VSI from the switch topology and remove it from the scheduler nodes.
7. Wait until the *Delete Element* command complete.
8. Clear the reset bit (*VSIGEN_RTRIG.VMSWR* or *VPGEN_VFRTRIG.VFSWR*).

7.4.6 Advanced Switching Capabilities

7.4.6.1 Security Features

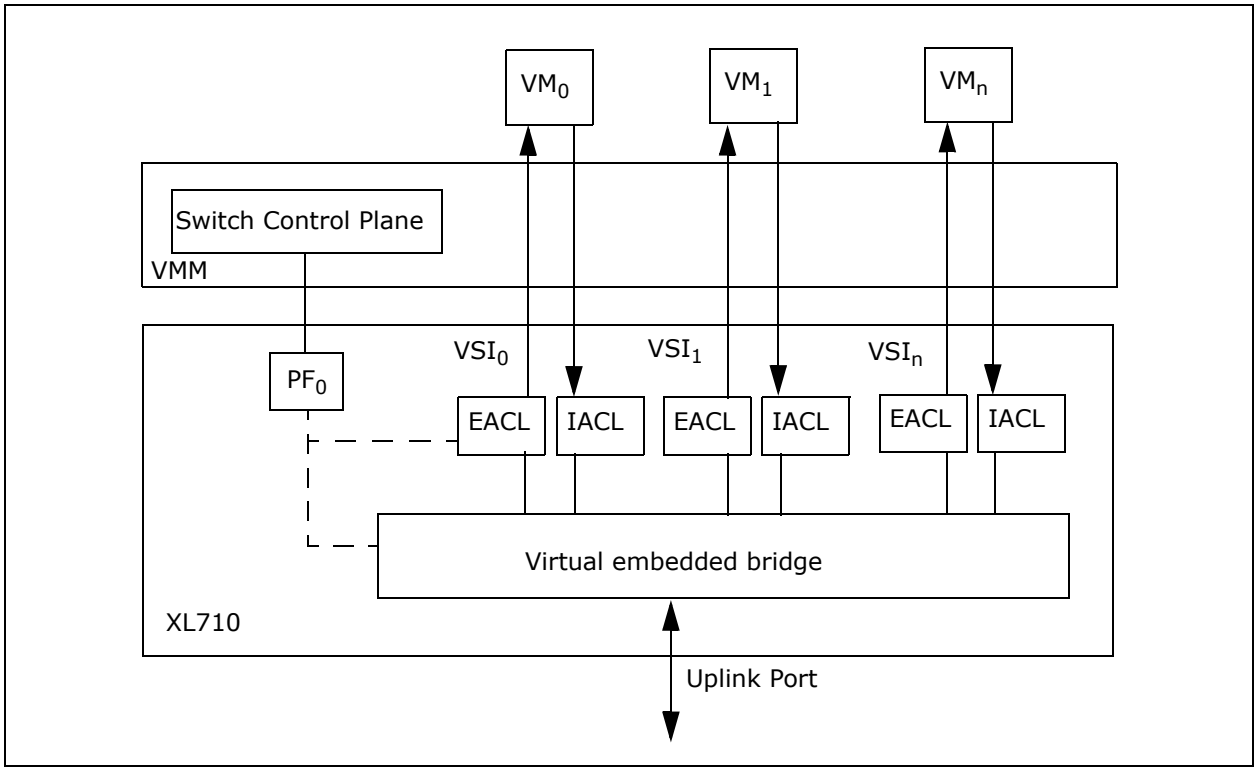


Figure 7-43. Switch control plane

7.4.6.1.1 Storm Control

As there is no separate path for multicast & broadcast packets, too much replicated packets might cause congestions in the data path. In order 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.

7.4.6.1.1.1 Storm Control Functionality

Note: Most of the parameters in this section refers to the *Set Storm Control Configuration* admin command described is Section 7.4.9.5.10.1.

The time interval over which Broadcast Storm control is performed is controlled by three factors:

- *PRT_SWT_SCBI* register
- Port speed

- INTERVAL value

The first two factors determine the Unit time interval as described in Table 7-57. 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).

The *PRT_SWT_SCBI* value is accessible only through the register interface.

Table 7-57. Storm Control Basic Interval by Speed

Port Speed	MIN Time Interval	MAX Time Interval	Default Time Interval (1 Mbit)
40 Gb/s	2.5 µs	2.5 ms	25 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 s	10 ms

The number of 64 bytes chunk of Broadcast or Multicast packets that are allowed in a given storm control interval is determined by the Set Storm control admin command.

In each Storm Control interval, the XL710 counts the number of 64 bytes chunks received that were part of broadcast or 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.

The XL710 supports two modes of reactions to storm event:

1. 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) or *BDICW* (for broadcasts). This mode is used as a rate limiter.
2. Block all Multicast or Broadcast packets from the moment the threshold is crossed until a full interval without threshold crossing is registered. The block is removed at the end of the interval until the threshold is crossed again. This mode is set by asserting *MDICW* and *MDIPW* (for multicast) or *BDICW* and *BDIPW* (for broadcasts). This mode is used for storm blocking.

The XL710 can be programmed to add all unknown packets to the broadcast storm control calculation. For example, packets that reached the port and where sent to the default VSI, may be counted as broadcast packets. This mode is activated by setting the *BIDU* field.

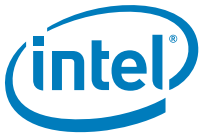
Any change in the storm control state (block or pass of multicast or broadcast packets) is indicated to all the PFs on this port via the *PFINT_ICR0.STORM_DETECT* interrupt cause. The current state is reflected in the *PRT_SWT_SCSTS* register.

For diagnostic purpose only, the storm control timer and counters can be read via the *PRT_SWT_SCTC*, *PRT_SWT_MSCCNT* & *PRT_SWT_BSCCNT* registers.

7.4.6.1.1.2 Storm Control Programming.

The following admin commands described are used to Program the storm control elements:

- Set Storm Control Configuration (see Section 7.4.9.5.10.1)
- Get Storm Control Configuration (see Section 7.4.9.5.10.2)



7.4.6.1.2 Anti Spoofing

XL710 supports Anti-spoofing functionality. This is a security feature that ensures a VM is sending frames with MAC address and VLAN associated with that VSI. A malicious guest can impersonate as a trusted host by performing MAC address spoofing. If the Anti-Spoofing feature is turned on then only trusted MAC addresses are allowed on the ingress VSI.

7.4.6.1.2.1 MAC Anti Spoofing

Trusted MAC address check is performed by performing a MAC address or {VLAN, MAC} lookup on the forwarding database. Since the forwarding database is populated by the VMM or control plane software when the VM/VSI is created, performing a MAC source address lookup on this database ensures that the VSI is transmitting with a source MAC assigned to that VSI.

MAC anti spoofing is enabled by the *VSI.Enable MAC anti spoof* field in the VSI context.

7.4.6.1.2.2 VLAN Anti Spoofing (Ingress check)

VLAN ingress check is performed on incoming packets from VMs to ensure if the VSI is a member of the VLAN. This check is performed on the ingress VLAN membership table. VLAN ingress check can be enabled or disabled on each VSI. So a VM cannot transmit with a VLAN on which the VSI is not a member. This is applicable for VLAN aware guests. If Port or protocol based VLAN is configured, a guest is not expected to send packets with 802.1Q tags then the "Admit Untagged/Priority Tagged" entry has to be set in port list.

VLAN ingress filtering is enabled by the *VSI.Enable VLAN anti spoof* field in the VSI context

7.4.6.1.2.3 FCoE Anti Spoofing

There are special conditions for the anti Spoofing of FCoE packets. FCoE packets can be sent only if FCoE is enabled for the VSI. FCoE enable is controlled per queue using the *FCENA* parameter in the transmit queue context (Section 8.4.3.4).

7.4.6.1.3 Port Based VLAN

XL710 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 7-58 describes the port VLAN parameters in the Port VLAN list and the matching parameters in the "Add VSI" command.

Table 7-58. 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" and "FCoE PVID + Default UP"	A different PVID is allowed for FCoE queues.
Default UP		

Table 7-58. Port VLAN List and VSI Parameters

Port Based VLAN list	Add VSI parameter	Notes
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	
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 7.4.6.1.6.1
Egress UP translation table	Egress UP translation table	see Section 7.4.6.1.6.2

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 7.4.6.1.6.1.

Once the VLAN ID association is made, the packet forwarding is performed using the {VLAN, MAC} forwarding table and VLAN membership table.

7.4.6.1.3.1 FCoE Dedicated VLAN

In order to allow different VLANs for LAN and FCoE traffic, each VSI can define up to two port based VLAN. The traffic from each queue can be associated with one of the two VLANs using *ALTERNATE_TAG* bit in the transmit queue context.

The software interface used to implement the port based VLAN is described as part of Section 7.2.3.2.2.

There is no enforcement of the separate VLAN for FCoE receive traffic

7.4.6.1.4 Private VLAN

XL710 supports configuration of Private VLANs (PVLAN) in the virtual embedded switching (VEB) mode. Private VLANs are used to provide Layer 2 level security by isolation of Virtual Machines within a VLAN (or IP subnet). Since VLAN defines a broadcast domain, all servers connected to the same VLAN can listen to broadcast packets. So a malicious VM can listen to neighboring Virtual servers and launch direct attacks bypassing the security policies enforced by Enterprise network switches. PVLAN provides Layer 2 security by creating sub domains within the same primary VLAN without the need for an L3 Router. These sub-domains are called secondary VLANs.

A virtual port in a private VLAN can be configured to one of 3 modes as follows:

- **Isolated ports** — These cannot talk to any other virtual ports except to ports that are configured as Promiscuous ports in the same primary VLAN. So a VM that only needs access to external network through uplink port has to be connected to isolated ports.
- **Community ports** —The ports within a PVLAN community can talk to one another within the same community group (community VLAN) and also to the Promiscuous ports in the same primary VLAN. There can be one or more PVLAN communities within the same primary VLAN. So a group of VMs that need to communicate with each other and also need external network access can be configured as members to one of PVLAN communities in the primary VLAN.

- **Promiscuous ports** — These can talk to all of the ports in the same primary VLAN. Typically uplink ports and virtual ports that are connected to service VMs (for management purposes) can be designated as promiscuous ports. (Note: The meaning of Promiscuous ports within the context of PVLAN is different from the generic promiscuous definition.)

Figure 7-44 provides an illustration of PVLAN configuration.

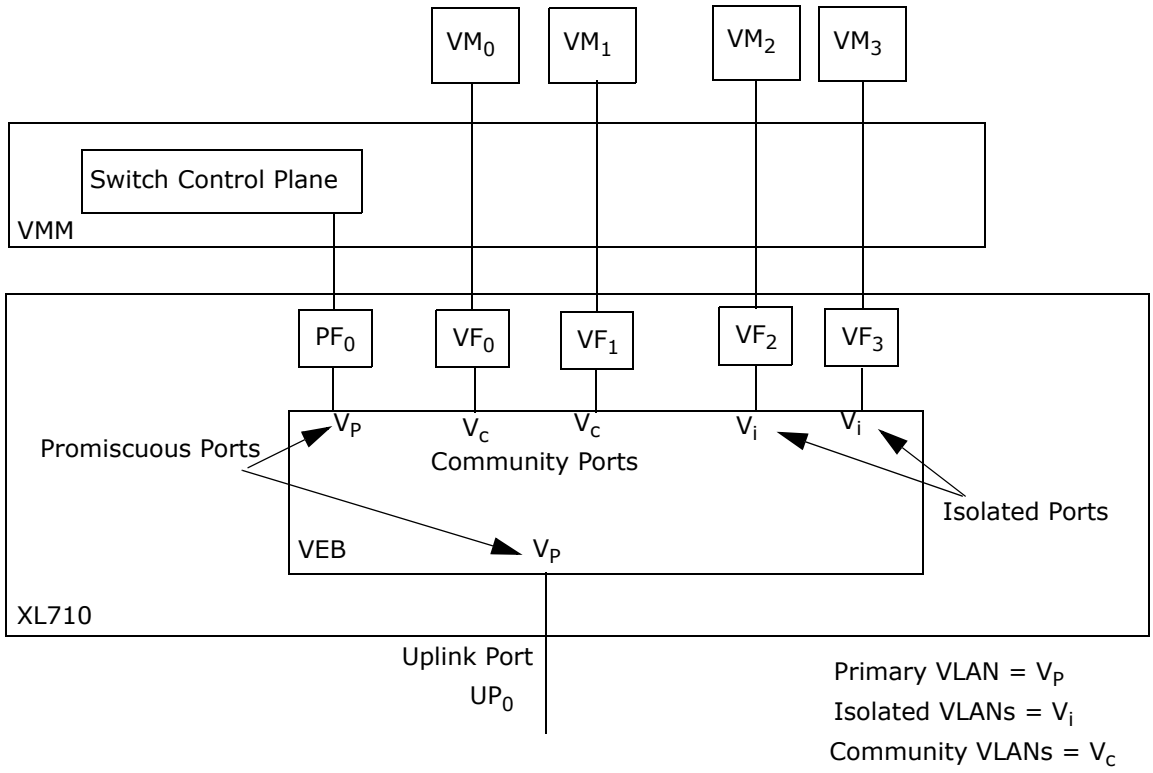


Figure 7-44. Private VLAN configuration

A primary VLAN V_P is configured on the VEB. Secondary VLANs, V_c is configured as community VLAN, and V_{i1}, V_{i2} are configured as isolated VLANs. The uplink port and the VEB management ports are configured as Promiscuous Ports; these ports will communicate with all the ports in the primary VLAN (V_p), including the community VLAN (V_p,V_c), and isolated VLAN (V_p,V_i).

The Community VLAN ports (VF_0 and VF_1) can communicate only the between themselves and the uplink port UP_0(and the management port PF_0). Isolated ports VF_2 and VF_3 can only communicate with uplink UP_0 (and the management port PF_0). The isolated ports cannot communicate with other isolated ports (VF_2 cannot communicate with VF_3) or to ports in community VLAN (VF_0 or VF_1).

So VLAN pairing is used to represent secondary VLANs. There can be multiple community VLANs configured within a single primary VLAN (for example Vc1, Vc2, etc.), however only one VLANID Vi is used to represent isolated VLANs.

So the virtual port configuration and associated VLAN IDs are created as VLAN pairs as follows:

- Virtual ports (connected to VMs) are configured with VLAN pairs (Primary VLAN, Secondary VLAN)

- The distinction between the different roles is done by defining the ingress and egress membership of each VSI for the primary and secondary VLAN as described in the table below:

Table 7-59. Private VLAN Membership

Port Type	Member of Primary egress list	Member of Primary ingress list	Member of Secondary egress list	Member of Secondary ingress list
Regular VLAN	Yes	Yes	N/A	N/A
Promiscuous	Yes	Yes	Yes	Yes
Community	Yes	Yes	Yes	No
Isolated	No	Yes	Yes	No

7.4.6.1.4.1 Isolated VLAN Configuration and Forwarding

Virtual Port VF2, VF3 are configured as isolated ports to be members of both Vp and Vi. Frames from the network to virtual port (ingress) will be arriving with primary VLAN ID Vp. However, frames from the virtual port to the network (egress) will always be assigned secondary VLAN ID Vi. Broadcast and multicast frames from the virtual port will not be forwarded to other isolated ports in the isolated secondary VLAN Vi; they will only be forwarded to uplinks (or ports configured as promiscuous ports in primary VLAN).

The forwarding database will have one entry {Vp, MAC2} pairing for virtual port VF2. If a port based VLAN is assigned to the port, then this will be Vi. Otherwise, Vi will be the only VLAN ID in the private VLAN to be part of the egress VLAN list for VF2. Packets arriving from network will carry Primary VLAN ID {Vp, MAC2}. Only Vp will be part of the ingress VLAN list for this port.

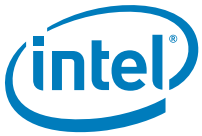
If a unicast packet arrives from network that is designated as isolated VLAN, then it cannot be forwarded to isolated ports. Such frames should be dropped. If a multicast or broadcast packet arrives from the network with VLAN ID Vi, then it cannot be forwarded to isolated ports.It can only be forwarded to promiscuous ports, in this case it is management port PF0.

7.4.6.1.4.2 Community VLAN Configuration and Forwarding

Virtual Port VF0, VF1 are configured as community ports to be members of both Vp and Vc. Frames from the network to virtual port (ingress) will be arriving with either primary VLAN ID Vp or secondary VLAN ID Vc. However, frames from the virtual port to the network (egress) will always be assigned secondary VLAN ID Vc. Broadcast and multicast frames from the virtual port will only be forwarded to other virtual ports that are members of the community VLAN Vc and to uplinks (or ports configured as promiscuous ports in primary VLAN).

The forwarding database will have two entries {Vp, MAC1} and {Vc, MAC1} pairing for virtual port VF1. If a port based VLAN is assigned to the port, then this will be Vc. Otherwise, Vc will be the only VLAN ID in the private VLAN to be part of the egress VLAN list for VF2. Ingress packets from network towards VF2 will used either {Vp, MAC1} or {Vc, MAC2} for forwarding. Both Vp and Vc will be part of the ingress VLAN list for this port.

If a unicast packet arrives from network that is designated as community VLAN Vc then it can be forwarded only to community ports that are members to the same community VLAN Vc. If a multicast or broadcast packet arrives from the network with VLAN ID Vc then they can be forwarded to other community ports in the same community VLAN Vc and to promiscuous ports, in this case it is management port PF0.



7.4.6.1.4.3 Primary VLAN, Promiscuous Port Configuration and Forwarding

A primary VLAN Vp is created with all the community ports, isolated ports and promiscuous ports as members. Unicast packets arriving from the network with primary VLAN Vp are forwarded based on {Vp, MACx}. Egress check is performed to find out the member ports. Multicast or Broadcast packets are forwarded to based on {Vp, Multicast MACx} and forwarded to member ports. Broadcast is forwarded to all members of primary VLAN.

Ports that are configured as Promiscuous ports are typically Uplink ports (UP0) and switch management Ports (PF0). The promiscuous ports are members of primary VLAN Vc and all secondary VLANs, Vc, Vi. So the promiscuous ports can receive broadcast packets from any of the secondary VLANs.

Note: In order to assure proper operation of Private VLANs both internal VEB and external access switch need to be configured with the same Private VLAN designations.

7.4.6.1.5 Local VLANs

A VLAN can be defined as local. In this case, packets sent with this VLAN ID will not be forwarded to the network and packet from the network with this VLAN are dropped.

Locality is defined at the VEB level.

A VLAN can be defined as local using the *Add VLAN* admin command with the *LocalVLAN* attribute set.

7.4.6.1.6 User Priority Bits (802.1p) Handling

The UP bits are used to differentiate between multiple classes of traffic. XL710 supports multiple use cases related to the handling of the UP bits:

1. An OS which is not DCB/traffic type aware and uses a single TCID.
2. An OS which 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 FCoE 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.
3. An un-trusted OS, that may set UP bits of TCs it is not allowed to use.
4. An OS that can use a single queue per TCID, but would like to use more UPs in order to gain from the differentiated QoS in the network.

The following sections describes the handling of UP bits in transmit and receive to support the above use cases.

7.4.6.1.6.1 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 7.4.6.1.3.

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 will be no drop of packets due to a wrong 802.1p tagging, if a wrong tag is requested the tag will be 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.

The following diagram shows the algorithm:

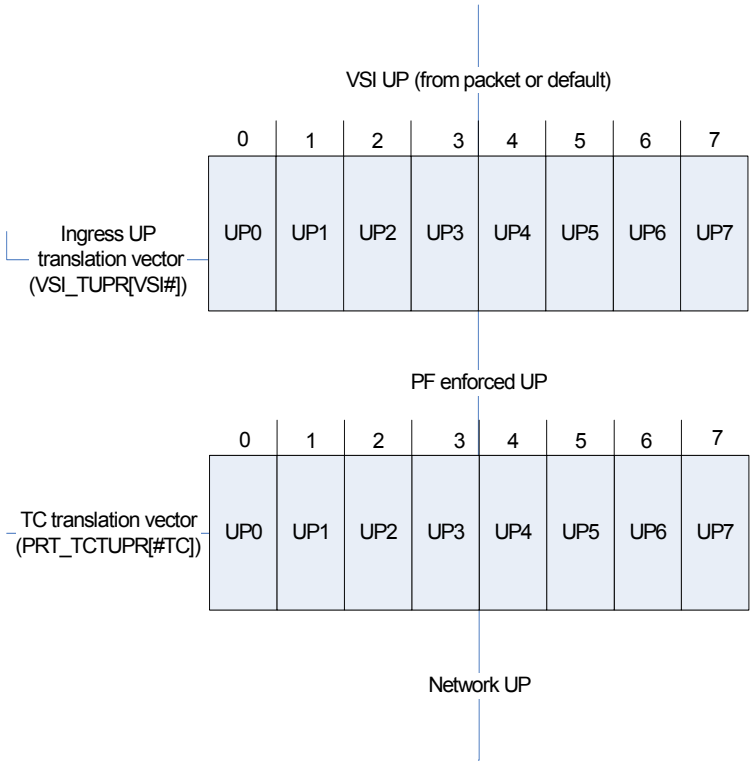


Figure 7-45. Example: Non VLAN aware, Non DCB aware OS

Assume a VF that queues LAN traffic and FCoE traffic to different queues, but is not VLAN aware. In this case, the packet will be sent by the driver with no VLAN and the default UP set in the port based VLAN (e.g. 0) will be the initial UP of the packet. In this case, the *Ingress UP translation vector* (*VSI_TUPR*) is not relevant and may 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 may be {0,1,2,0,0,0,0,0} and the TC table for storage may be {4,4,4,4,5,4,4,4}. So a LAN packet will go out with a UP of 0, and a storage packet will go 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 and LAN low priority traffic with a UP of 3 and the storage traffic as UP 7. In this case, the VF translation table may be {0,1,2,0,2,4,4} and the TC tables as above. The low priority LAN will go 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, i.e. mapping a UP to itself. If UP translation is not needed these mapping should not be changed.

7.4.6.1.6.1.1 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 then the UP may 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, then 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).

The tag to which translation is applied can be either S-tag or external VLAN.

Figure 7-46 describes the translation process.

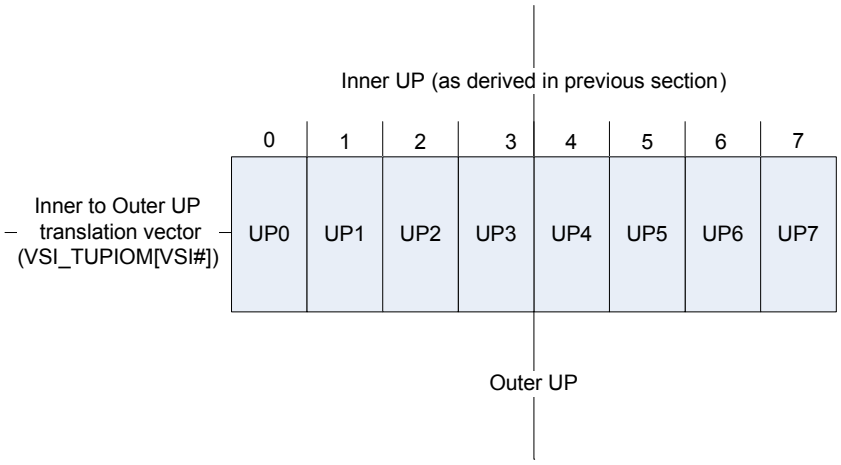


Figure 7-46. TX Inner to Outer UP translation

7.4.6.1.6.2 Receive Functionality

7.4.6.1.6.2.1 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 may 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.

7.4.6.1.6.2.2 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 follow:

1. For a monolithic OS, VMM or for a guest OS which is VLAN aware, the UP is exposed as part of the VLAN as requested by driver (either in the packet or in the descriptor).
2. If the guest OS is DCB aware but not VLAN aware, it will get the priority bits as part of the receive descriptor in the VLAN tag field. The VLAN ID will be zero, but the priority bits will be valid.
3. A guest OS that is not VLAN aware and not DCB aware will not get the user priority bits (UP) at all.

7.4.6.2 Switch Diagnostics

This section describes the mirroring and loopback features. The statistics of the switch are described in Section 7.11.

7.4.6.2.1 Mirroring

The XL710 supports 64 mirroring rules. Each rule can be coupled to any of the VEB or port aggregator in the device. The destination port of each rule (the mirror port) must be connected to the switching element in which the rule is applied. Packets that match any of the mirroring rules assigned to this element will be forwarded to the VSI defined in the mirror rule.

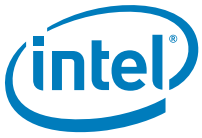
In order to differentiate packets received according to the different rules, the matched rule will be indicated in the Mirror Rule ID *(MIRR)* field in the receive descriptor.

Note: The switching algorithm allows mirroring to mirror ports even if the loopback is disabled for this port. The controlling software needs to make sure mirror ports are assigned only to ports that can receive local traffic.

Each rule can is defined by the following parameters:

1. A rule ID - a value in the 0..63 range reflected in the MIRR field in the receive descriptor.
2. A mirror port to which packets matching the mirroring conditions will be sent.
3. A set of mirrored ingress VSIs - All the packets received by these VSIs will be sent also to the mirror port.
4. A set of mirrored egress VSIs - All the packets sent by these VSIs will be sent also to the mirror port.

Additional rules can be defined to create ingress VLAN mirroring - When such a rule is created, all the traffic received in a set of given VLANs by any of the virtual ports either from the uplink or from local VMs within a given VEB are forwarded to a mirror port.



Note: Packets forwarded by Ingress VLAN mirroring will not be identified by the *MIRR* field and the *UMBCAST* field as a mirror packet. The matched mirror rules can be inferred from the VLAN of the packet.

Mirroring rules can be applied using the following admin commands:

- Add Mirror rule (Section 7.4.9.5.9.1).
- Remove Mirror rule (Section 7.4.9.5.9.2).

Note: A mirror port can not be defined in the ingress mirroring list of another mirroring rules.

A single rule can be either ingress VSI, egress VSI or VLAN and can not include multiple types of rule.

Ingress or egress VSI traffic can not be mirrored to multiple mirrored ports.

A VSI can not be part of more than a single port mirror rule (can not be mirrored twice). This limitation does not include VLAN mirroring.

A control VSI that uses the switch override field (SWTCH) in transmitted packets descriptor can not be part of an ingress mirror rule.

7.4.6.2.1.1 Ingress Mirroring Flow

Ingress mirroring acts on all the packets that are sent from a given VSI, no matter if the packet is sent to the network or to a local VSI and doesn't depend on the "Allow Loopback" flag in the VSI context.

Packets dropped due to anti spoofing are also mirrored.

7.4.6.2.1.2 Egress Mirroring Flow

Egress mirroring acts on all the packets that are received by a given VSI no matter if the packet was received from the network or from a local VLAN.

7.4.6.2.1.3 Mirror VSI Setup

A VSI used to receive mirror traffic should be dedicated to this type of traffic and thus should not be used to send or receive other traffic.

7.4.6.2.2 VSI Loopback

Some systems requires a local loopback capability at the switch level allowing to receive all the packets sent from a VSI back to the same VSI.

This capability is achieved by using the following steps:

1. When creating the VSI using the *Add VSI* command, set the *Allow destination override* bit.
2. When sending a packet, set in the descriptor the *SWTCH* field to 11b (Target VSI) and set the *Segmentation Parameters.VSI* field to the value of this VSI.

Packets sent using descriptors with the attributes mentioned above will be sent back to a receive queue of the same VSI.

7.4.6.3 Cascaded Port Virtualizer Offloads

XL710 supports VSIs used as cascaded port virtualizers as described in Section 7.4.2.4.3. Such a VSI provides some special offloads to support this mode.

7.4.6.3.1 Transmit Offloads

S-tag insertion: As this VSI supports multiple S-tags, the S-tag/ can not be inserted by the XL710 using the port based tagging mechanism. The driver may indicate the S-tag to insert in the *L2TAG2* field in the transmit descriptor and set the *IL2TAG2* bit. The hardware will then insert the right tag to the packet.

The ability of a VSI to control the S-tag from the descriptor or from the packet is set by the *Accept tag from host* in the *Add VSI* command.

Note: If a packet is sent by the driver with the S-tag as part of the packet, it must also include the VLAN.

If S-tag needs to be inserted, 32 bytes transmit descriptors must be used.

7.4.6.3.2 Receive Offloads

- S-tag extraction: As this VSI supports multiple S-tags the driver must receive the S-tag in order to process the packet. These tags can be extracted from the packet and stored in the receive descriptor in the *L2TAG2 (1st)* field. The presence of these tags is indicated by the *L2TAG2P* bit. This offload is activated via the *Report S-tag* field in the *Add VSI* command.

Note: This capability is orthogonal to the VLAN tag extraction and may be requested with or without VLAN extraction.

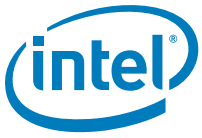
- Source Pruning: This capability should be disabled in a cascaded S-comp by clearing the *Prune based on ingress E-PID* enable field in the VSI context.
- VMDq1 queueing: A cascaded Port Virtualizer may request to queue packets with different tag values to different queues, similar to the VMDq1 mode supported for cascaded VEBs. In this mode packets with an unrecognized S-tag are forwarded to the default queue of the VSI.

7.4.6.4 DCB and Rate Limit Support

Each VSI can be associated with a scheduling element and with a set of user priorities and traffic classes. Each user priority is represented by a set of queue pairs and a scheduler element. In case multiple UPs are associated with the same traffic class, a scheduler element may be added later to group those UPs into a single TC. The VSI scheduling element can be used to define an SLA for this VSI. The UP and TC scheduling elements are used to set an ETS policy for this VSI. Each S-channel created is also associated with a scheduler element allowing a different SLA for each channel.

The assignment of scheduling elements is done internally when an *Add VSI* or an *Add VEB* admin command is received. The disassociation is done when a *Delete Element* admin command is received.

The traffic classes created are set according to the *Enabled UPs* field in the *Add VSI* command. The handles to the traffic class rate limiters are returned as part of the *Add VSI* response. The addition of traffic class scheduler elements can be done after the VSI exists using the *Configure VSI Bandwidth per Traffic Class* admin command (Section 7.8.4.7).



In order to control the rate limiter behavior, the scheduler admin commands (Section 7.8.4) should be used. When accessing a switching element (VSI or S-channel) scheduler element, the SEID is used as the handle to the node. When accessing a TC the handles returned in the *Add VSI* response are used.

7.4.7 Flows

This section provide a summary of the processing across all the switching elements (day in a life of a packet) in Tx and Rx.

7.4.7.1 Tx Flow

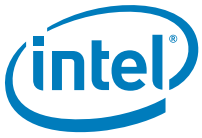
The following steps are taken once a packet is accepted for transmission by the switching engine. This flow assumes any stateful offload needed had been applied to the packet:

1. **Anti Spoofing:** If enabled in the Add VSI command VSI sending the packet is part of those addresses destination, the packet is allowed to proceed
2. **VLAN insertion**: If needed insert the VLAN tag to the packet, either as a host request or as a port based VLAN.
3. **S-tag insertion**: If the VEB or the function is connected to an S-comp, add the S-tag of the channel to the packet.
4. **UP replacement:** replace the UP bits as described in Section 7.4.6.1.6.1.
5. **VEB filtering**: If the port/queue that sends the packet is connected to a VEB, the algorithm described in Section 7.4.8.6 is applied to create a list of destination ports. Otherwise, the list of destination ports is simply the network.The VEB filtering for Tx packets includes the following steps:
   a. Defining the list of VSIs that are candidate for reception of the packet. These VSI are defined according to the filters set using the *Add MAC, VLAN pair*, *Add Cloud filters* and *Add Ethertype filters* commands (Section 7.4.9.5.8).
   b. Add EMP owned VSI that should receive the packet (Section 7.4.4.8). This includes forwarding to the BMC channel using decision filters (Section 10.3) and control packets.
   c. Potentially removing from this list the VSIs that do not belong to the VLAN group as defined by the *Add VLAN* command (Section 7.4.9.5.8.3).
   d. Potentially removing from this list the VSI that sent the packet. This removal can be based on comparison of the SA of the packet to the MAC table.
   e. Add ingress mirror VSI as defined by the *Add Mirror Rules* admin command (see Section 7.4.6.2.1).
6. For a copy of the packet sent to the network follow the steps in step7 For a copy of the packet sent to the sideband interface follow the steps in step8 For copies of the packet sent to local functions follow the steps in step9
7. **Processing of Packets sent to the network:**
   a. **MAC processing**: If applies, insert a CRC on the packet.
   b. Forward to the network.
8. **Processing of packets sent to the sideband interface**:
   a. Forward the packet to the sideband interface.
9. **Processing of packets sent to local functions**:
   a. **Forward to function**: Forward the packet to the selected function for further processing.

7.4.7.2 Rx Flow

The following steps are taken for each packet received from the network by the switching engine:

1. **MAC processing**: Check the CRC and other error conditions (see packets *PRT_SBPVSI.SBP*) is set. If Pass Bad Packets is set, add the Bad Frames VSI (*PRT_SBPVSI.BAD_FRAMES_VSI*) to the VSI list . Identify flow control packets and Priority flow control packets and react to them (see Section 3.2.1.5).
2. **Storm Control:** If the storm control mechanism determines that too much broadcast or multicast packets where received, the packet may be dropped (see Section 7.4.6.1.1).
3. **Untagged packets processing**: If the packet is not tagged (no S-tag) and a default switch ID is defined for the port (*PRT_SWT_SWITCHID.SWIDVALID* = 1) append a default switch ID to the packet description. The resulting Switch Id/S-tag defines the context in which the next step is done.
4. **VEB/Port aggregator/L2 filter**: According to the configured switching element connected to the S-channel/physical port, apply the algorithms defined in Section 7.4.8.3, or Section 7.4.8.6, to define a list of destination ports. This includes the following steps:
   a. Defining the list of VSIs that are candidate for reception of the packet. These VSI are defined according to the filters set using the *Add MAC, VLAN pair*, *Add Cloud filters* and *Add Ethertype filters* commands (Section 7.4.9.5.8).
   b. Add EMP owned VSI that should receive the packet (Section 7.4.4.8). This includes forwarding to the BMC channel using decision filters (Section 10.3) and control packets.
   c. Potentially removing from this list the VSIs that do not belong to the VLAN group as defined by the *Add VLAN* command (Section 7.4.9.5.8.3).
   d. Potentially removing from this list the VSI that sent the packet. This removal can be based on comparison of the SA of the packet to the MAC table.
   e. Add egress mirror VSIs as defined by the *Add Mirror Rules* admin command (see Section 7.4.6.2.1).
5. The resulting list of destination ports can include local ports or sideband interface.
6. For a copy of the packet sent to the sideband interface follow the steps in step8. For copies of the packet sent to local functions follow the steps in step8.
7. **Processing of packets sent to local functions**:
   a. **S-tag and VLAN removal**: If apply, remove the S-tag and VLAN tag from the packet and optionally report them in the Rx descriptor.
   b. **UP replacement:** replace the UP bits as described in Section 7.4.6.1.6.2.
   c. **Forward to function**: Forward the packet to the selected function for further processing including queuing.
8. **Processing of packets sent to the sideband interface or to the EMP**:
   a. Forward the packet to the sideband interface or to the EMP. The VSI context defines the destination within the EMP VSI (one of the sideband interfaces or the EMP) either using the regular switch or the packet classifier.



7.4.8 Switching Algorithms

7.4.8.1 LAN/SAN Port Extender Algorithm

This algorithm translates a Destination MAC address to a switch ID. After this algorithm is applied, the regular per switch forwarding algorithm should be applied to each of the switch IDs. The algorithm is relevant only to receive traffic.

```
// Global parameters

Port.DefaultStag; Can be Null or an S-tag.

// Define Lookup tables

DA Table: Array of {DA ,LAN , Switch ID}

EComp_function(Packet) {

// Variables

    STag = Port.DefaultStag

    SourceStagFlag = FALSE // SourceStagFlag is not used by this algorithm.

//Define packet parameters

    DA = Packet.DA: the DA of the packet.

    Source Port // The port from where the packet was received.

// Switching algorithm

// Selection of S-tag using DA

    If ({DA, source port} match entry e in DA table): DSTag = e.S-tag;

    Return DStag, SourceStagFlag;

}
```

7.4.8.2 S-comp Forwarding Algorithm

The algorithm below refers only to S-comp Receive traffic. As described above, transmit traffic is always sent to the LAN.

Packets with unmatched S-tag will be forwarded to a default VSI of the port, if enabled.

Packets forwarded to the default VSI can not be subject to L2 filtering.

This algorithm defines a switch ID that identify the S-channel to which the packet should be forwarded. The switch ID is used to identify a potential VEB connected to this S-channel. The forwarding inside this channel is defined according to the L2 forwarding algorithm (Section 7.4.8.3) or VEB/VEPA algorithm (Section 7.4.8.6)

If an S-comp is not defined on the port, then all the packets with an S-tag are dropped.

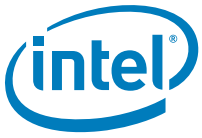
```
// Global parameters

Port.DefaultVSI; Can be Null or a VSI
```

```
Port.Default_Switch_ID; // The switch ID to use for untagged.

DefaultVSIValid - per port// If set, packets with an S-tag that do not match any of these
programmed S-tags or an untagged packet that do not match the expected ethertypes will be sent to
the default VSI, otherwise the packet is dropped.

// Define Lookup tables

StagTable: Array of {STAG}

// Control Port forwarding rules

SComp_function(Packet) {

// Variables

Dest_VSI - Null; // Destination VSI in case of default VSI;

Switch_ID = Null;

//Define packet parameters

STag = Packet.Stag: the S-Tag of the packet.

Untagged // True if packet has no S-tag.

// Switching algortihm

// control port forwarding

// Note - the usual case of a control VSI is to receive LLDP packets or other packets based on
special Ethertypes. However, the device allows any L2 forwarding to the control VSI assuming the
right forwarding rule is added. See Section 7.4.8.6 for details of the available rules.

If (Untagged) { Switch_ID = Port.Default_Switch_ID;}

    // S-tag forwarding

If(Stag match entry e in StagTable)  {

        Switch_ID = S-tag

} else if (DefaultVSIValid)

    Dest_VSI = Port.DefaultVSI

else drop packet.

Return Switch_ID, Dest_VSI;

}
```

7.4.8.3 L2 Filtering Algorithm

The following pseudo code describes the algorithm used to determine if a packet passes the L2 filtering element.

```
// Global parameters

// Define Lookup tables

MacVlan Table: Array of {MAC (MAC Address), VLAN (VLAN tag)}

Mac Table: Array of {MAC (MAC Address)}
```

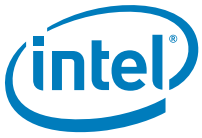


```
VLAN table: Array of {VLAN (VLAN tag)}

HashMacVlan Table: Array of {HashMAC (Hash Values), VLAN (VLAN tag), AddressType}

HashMac Table: Array of {HashMAC (Hash Values) , AddressType}

EtherType Table : Array of {Etype (Ethertypes Values)}

MacEtherType Table: Array of {MAC (MAC Address), Etype (Ethertype value)}

// Define Virtual ports modes

Port.PUE // Promiscuous Unicast Enable

Port.PME // Promiscuous Multicast Enable

Port.BAM // Broadcast Enable

Port.VPE // Promiscuous VLAN enable

Port.MaxSize: Max Packet size

L2_function(Packet)

// Variables

MFilter: = False // MAC Filtering

VFilter: = False // VLAN Filtering

EFilter: = False // Exclusive Filtering

NEPMFilter = False // Non Exclusive Perfect MAC Filtering

NEPFilter = False // Non Exclusive Perfect Filtering

NEIPMFilter = False // Non Exclusive Imperfect MAC Filtering

NEIPFilter = False // Non Exclusive Imperfect Filtering

Pass = False // Final decision.

//Define packet parameters

DA = Packet.DA //Destination Address of the packet

VID = Packet.VLAN ID // Vlan tag of the packet

Etype = Packet.Ethertype // Ethertype of the packet

AddressType //Type of address of the DA. Can be Unicast, Multicast or Broadcast.

HDA = HashFunction(DA).

PSize: Packet size. This do not include any tag or other header removed by previous stages or to be
added by following stages.

// Exclusive Filters

For Each entry e in MacVlan Table

    If (DA == e.MAC and (VID == e.VLAN or (VID == NULL and e.VLAN == 0)) MFilter = True;

For Each entry e in Mac Table

    If (DA == e.MAC) MFilter = True;
```

```
For Each entry e in Ethertype Table

    If (Etype == e.Etype) MFilter = True;

For Each entry e in MacEthertype Table

    If (DA == e.MAC and Etype == e.Etype) MFilter = True;

For Each entry e in Vlan Table

    If (VID == e.VLAN or (VID == NULL and e.VLAN == 0)) VFilter = True;

// VLAN filters are ANDed with the previous filters unless promiscuous VLAN is enabled

EFilter = MFilter and (VFilter or p.VPE} and AddressType == Unicast;

// Non Exclusive Perfect Filters

If (AddressType == Unicast and p.PUE == 1) NEPMFilter = True;

If (AddressType == Multicast and (p.PME == 1 or MFilter)) NEPMFilter = True;

If (AddressType == Broadcast and (p.BAM == 1 or MFilter)) NEPMFilter = True;

// VLAN filters are ANDed with the previous filters unless promiscuous VLAN is enabled

NEPFilter = NEPMFilter and (VFilter or p.VPE};

// Non Exclusive Imperfect Filters

For Each entry e in HashMACVlan Table

    If (HDA == e.HashMAC and (VID == e.VLAN or (VID == NULL and e.VLAN == 0)) and AddressType ==
e.AddressType) NEIPMFilter = True;

For Each entry e in HashMAC Table

    If (HDA == e.HashMAC and AddressType == e.AddressType) NEIPMFilter = True;

// VLAN filters are ANDed with the previous filters unless promiscuous VLAN is enabled

NIEPFilter = NIEPMFilter and (VFilter or p.VPE};

// Packet size filtering is done at the queue level, so it is not part of the switch algorithm

Pass = (EFilter or NEPFilter or NEIPFilter)

Return Pass;

}

HashFunction(MAC){

Hash = MAC[6:0] // Use the 7 least significant bits of the MAC address (last bits on the wire).

Return Hash

}
```

7.4.8.4 VEB/VEPA Switching Algorithm

The following pseudo code describes the algorithm used to determine which ports should receive a packet in a VEB/VEPA.



// Global parameters

// Define Lookup tables

MacVlan Table: Array of {MAC , VLAN , DestinationVSIList };

MAC Table: Array of {MAC , DestinationVSIList }

VLAN table: Array of {VLAN, IngressVSIList, EgressVSIList, Local}

HashMacVlan Table: Array of {HashMAC (Hash Values), VLAN, AddressType (unicast/Multicast), DestinationVSIList }

HashMac Table: Array of {HashMAC (Hash Values), AddressType (unicast/multicast), DestinationVSIList }

MacEthertype Table : Array of {MAC , Etype , Source, DestinationVSIList, Drop}

Ethertype Table : Array of {Etype , Source, DestinationVSIList, Drop}

// Define VSI modes

VSI[i].PUE: Promiscuous Unicast Enable per VSI // Target VSI

VSI[i].PME: Promiscuous Multicast Enable per VSI // Target VSI

VSI[i].BAM: Broadcast Enable per VSI // Target VSI

VSI[i].VLANPruneEnable // Target VSI - Should be cleared in case of promiscuous enable

VSI[i].AllowLoopback: Loopback Enable per VSI // Source VSI

VSI[i].MACPruneEnable: // Defines if the source VSI (identified either by SA,VLAN or by SA) should be removed from the list of destination VSI.

VSI[i].EnableMACAntiSpoof: // Source VSI

VSI[i].EnableVLANAntiSpoof: // Source VSI

VSI[i].LANEnable // Source VSI - should be set in each VSI tied to a switch that is connected to the LAN. This bit is cleared if the VSI is added to a floating VEB.

VSI[i].AllowOverride // Allows override of the destination - usually set for control VSIs.

// Switch generic parameters

DefaultVSI; Can be Null or one of the VSIs. // Defined as part of the S-tag table or as a default VSI of the port.

ControlVSI; Can be one of the VSIs or the embedded controller.

// Mirroring rules

Int n_mirror: Number of mirror rules

Mirror rule[1 .. n_mirror] = {Bool Ingress_valid, Bool Egress_valid, Bool VLAN_Valid, IngressVSIList (List of VSI), EgressVSIList (List of VSI), VLANList (List of VID), MirrorPort};

Dport Switch_function(SVSI : Source VSI, Packet, FromLAN, FromHost)

// Variables

PFDPort: List of VSI = {} // Perfect Filtering - empty list at startup.

IFDport: List of VSI = {} // Imperfect Filtering - empty list at startup.

PMDport: List of VSI = {} // Promiscuous Filtering - empty list at startup.

OFDPort: List of VSI = {} // Override VSI Filtering - empty list at startup.

VFDPort: List of VSI = {} // VLAN membership VSI Filtering - empty list at startup.

MNGPort: List of VSI = {MDEF filtering result as defined in Section 10.3}

MNGOnly: Bool = {MNGonly result as defined in Section 10.3}

DPort: List of VSI = {} // Final List - empty list at startup. Can include also the LAN port for Tx packets.

PassAntiSpoof = False;

PassMACAntiSpoof = False;

PassVLANAntiSpoof = False;

//Define packet parameters

DA = Packet.DA: Destination Address of the packet

SA = Packet.SA: Source Address of the packet

VID = Packet.VLAN ID: Vlan tag of the packet - equal to zero if untagged packet.

Etype = Packet.Ethertype: Ethertype of the packet

AddressType: Type of address of the DA. Can be Unicast, Multicast or Broadcast.

HDA = HashFunction(DA).

PSize: Packet size. This do not include any tag or other header removed by previous stages or to be added by following stages.

LocalVLAN = False: Define if the packet is part of a local VLAN. See below for calculation.

Destination: // Can be either uplink (send only to LAN), local (switch decides on destination, but do not send to LAN), switchBased (regular switching algorithm) or a TargetVSI. This is a descriptor bit for packets received from a control port allowing override of the switching decision. This may be useful when sending switch generated packets like DCBX packets. Relevant only for transmit packets

Source = Tx or Rx // defines if this is a packet sent by the XL710 or received by the XL710.

// Combine override destination option.

// This logic creates the following indications to be used later

Specific_Port = NULL; // A specific port defined as the sole destination of the packet.

SwitchToLAN = TRUE; // Allow forwarding to LAN

SwitchToLocal = TRUE; // Allow forwarding to local VSI.

else if (Destination == TargetVSI and SVSI.AllowOverride) {Specific_Port = Target VSI from Descriptor};

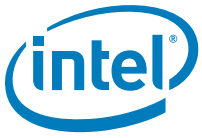
if ((Destination == UpLink and SVSI.AllowOverride) or DoNotLoopback or SVSI.ALLOWLOOPBACK == FALSE) SwitchToLocal = FALSE;

if ((Destination == UpLink and SVSI.AllowOverride) {Specific_Port = LAN};

if ((Destination == Local and SVSI.AllowOverride) or SVSI.LANDisable == TRUE) SwitchToLAN = FALSE;

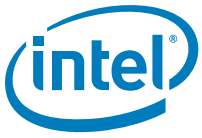
If (FromLan) SPort = LAN else SPort = SVSI.

// Anti Spoofing



```
// Local VLAN calculation

For each v in VLAN Table {if (v.VLAN == VID and v.LocalVLAN == True) LocalVLAN = TRUE}

// Need to check if the packet can enter the switch before applying all the rules

If (SVSI.EnableMACAntiSpoof == True and FromHost) {

// Only perfect match filters are used for anti spoofing checks

        For Each entry e in MacVlan Table

        if (SA == e.MAC and (VID == e.VLAN or (VID == NULL and e.VLAN == 0)) and SVSI in
e.DPortList ) PassMACAntiSpoof = True

    For Each entry e in Mac Table

        if (SA == e.MAC and SPort in e.DestinationVSIList ) PassMACAntiSpoof = True

    } else PassMACAntiSpoof = True;

// The step below (VLAN anti spoof) is really ingress VLAN filtering

If (SPort.EnableVlanAntiSpoof == True and FromHost) {

    For Each entry e in  VLAN Table

    // Note - an untagged packet will be compared with an entry of the table with VLAN = 0.

        if (VID == e.VLAN and SVSI in e.IngressVSIList) PassVLANAntiSpoof = True;

} else if (FromLAN and LocalVLAN == True) {

    PassVLANAntiSpoof = FALSE;

} else PassVLANAntiSpoof = True;

    PassAntiSpoof = PassVLANAntiSpoof and PassMACAntiSpoof;

If (PassAntiSpoof == False)  {DPort = null; Goto Mirroring Rules }// Drop Packet as DPort is empty;

// Switching algorithm

// Perfect Filters

For Each entry e in MacVlan Table {

    If (DA == e.MAC and (VID == e.VLAN or (VID == NULL and e.VLAN == 0)))

    PFDPort = PFDPort + e.DestinationVSIList; // Add VSI matching the MAC, VLAN pair.

}

For Each entry e in Mac Table { :

    If ((DA == e.MAC )

        PFDPort = PFDPort + e.DestinationVSIList; // Add ports matching the MAC only entry.

}

// Imperfect Filters

    For Each entry e in HashMACVlan Table where (HDA == e.HashMAC and (VID == e.VLAN or (VID ==
NULL and e.VLAN == 0 and AddressType == e.AddressType))):

        IFDPort = IFDPort + e.DestinationVSIList; // Add VSIs matching the HashMAC, VLAN pair.
```

```
    For Each entry e in HashMAC Table where (HDA == e.HashMAC and AddressType == e.AddressType):

        IFDPort = IFDPort + e.DestinationVSIList;// Add VSIs matching the HashMAC.

}

// Apply promiscuous modes

For each port p in VEB {

    If (AddressType == Unicast and p.PUE == 1) PMDPort = PMDPort + p;

    If (AddressType == Multicast and p.PME == 1) PMDPort = PMDPort + p;

    If (AddressType == Broadcast and p.BAM == 1) PMDPort = PMDPort + p;

}

// Override filters

For Each entry e in Ethertype Table where (Etype == e.Etype and Source == e.Source):

    OFDPort = OFDPort + e.DPortList; // Control port filtering.

For Each entry e in MacEthertype Table where (DA == e.MAC and Etype == e.Etype and Source ==
e.Source):

    OFDPort = OFDPort + e.DPortList; // Control port filtering.

// Create unified list

If (MngOnly) DPort = MNGPort;

Else If (OFDPort not empty) Dport = OFDPort + MNGPort; // Override

Else if (PFDPort not empty) DPort = PFDPort + PMDPort + MNGPort; // exact & promiscuous

else DPort = IFDPort + PMDPort + MNGPort; //imperfect & promiscuous

// Add default port - based only on forwarding tables.

if (DPort is empty and FromLAN and DefaultVSI != Null)

    DPort = DefaultVSI;

DPort_for_mirroring = Dport; // mirroring ignores VLAN prunning - should be fixed in future
project

// VLAN egress Filtering

// Note - to use VLAN only fitering (ignore all MAC addresses), the UPE, MPE and BAM bits should be
set in all ports. See Section 7.4.6.1.4 for rules to set the egress and ingress VLAN lists

For each entry p in DPort {

    if (p.VLANPruneEnable == True) {

        If (VID match VLAN in VLAN table) {

            for each e in Egress Vlan Table where (VID == e.VLAN or (VID == NULL and e.VLAN == 0))

                if (p NOT in e.EgressVSIList) DPort = DPort - p; // prune ports which are not member
of the VLAN;

        }

        Else Dport = Dport - p; // Remove all ports which are not in Promiscuous VLAN if VLAN not
```



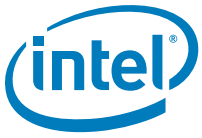
```
found in VLAN table.

    }

}

// Add external portif needed

// Note: In VEPA mode, all Tx packets should go to LAN only (VSI[i].AllowLoopback == 0)

if ((PFDport is empty or AddressType == Multicast or AddressType == Broadcast or
VSI[i].AllowLoopback == 0) and OFDPort is empty and FromHost and LocalVLAN == FALSE and
SVSI.EnableLAN) {

    DPort = Dport + LAN port;

    DPort_for_mirroring = Dport_for_mirroring + LAN port;

}// Prunning Rules

// Control Packets drop rules.

For Each entry e in Ethertype Table where (Etype == e.Etype and Source == e.Source):

        if (e.Drop == True) {

            DPort = Null;

        DPort_for_Mirroring = Null;

    }

    For Each entry e in MacEthertype Table where (DA == e.MAC and Etype == e.Etype and Source ==
e.Source):

        if (e.Drop == True){

            DPort = Null;

            DPort_for_Mirroring = Null;

        }

// Find the source port based on the source MAC address.

// The algorithm allows multiple source port, but the assumption is that a unicast address will
point to a single VSI.

For Each entry e in Mac Table

        if (SA == e.MAC ) SourcePorts = e.DestinationVSIList; For Each entry e in MacVLAN Table

        if (SA == e.MAC and VLAN = e.VLAN ) SourcePorts = SourcePorts + e.DestinationVSIList;

For Each port p in Dport {

        // Remove ports from list if loopback disabled for this port

    if (From Host SwitchToLocal == FALSE and p != LAN Port)

        DPort = Dport - p;

    // Remove uplinks  from list if requested by descriptor

    if (p == LAN Port and SwitchToLAN == FALSE) Dport = Dport - p ;

    // Prune Source port
```

```
    if (p.MACPruneEnable== True and ( p in SourcePorts) DPort = Dport - p;

// Prune by size is done at the queue level, so this is not handled in the switch.

}

// Mirroring rules

MatchedRuleID = Null;

for ( i = 1 to n_Mirror) { // Apply all mirroring rules

// Note - a port can be member only in a single VSI based mirror rule.

    Bool Mirror = FALSE;

    Bool VLAN_Mirror = FALSE;

    // This rule is applied even if packet is droped.

    if (Mirror rule[i].Ingress_valid and SPort in Mirror rule[i].IngressPortList) Mirror = True;

    For each p in Dport_for_mirroring {

        if (Mirror rule[i].egress_valid and p in Mirror rule[i].EgressPortList) Mirror = True;

    }   if (Mirror rule.VLAN_Valid and VID in Mirror rule.VLANList and DPort is not empty)
VLAN_Mirror = True;

    if (Mirror) {

        DPort = Dport + Mirror rule[i].MirrorPort;

        MatchedRuleID = i; // Only a single non VLAN rule should match

    }

    if (VLAN_Mirror) DPort = Dport + Mirror rule[i].MirrorPort;

}

If (Specific_Port != NULL) DPort = Specific_Port; // Descriptor from control port can override the
entire switch decision

Return Dport // Note - the Dport list, should include a single copy of each VSI.

HashFunction(MAC){

Hash = MAC[6:0] // Use the 7 least significant bits of the MAC address (last bits on the wire).

Return Hash

}
```

7.4.8.5 Cloud VEB Algorithm

The following pseudo code describes the algorithm used to determine which ports should receive a packet in a Cloud VEB/VEPA.

These algorithms describe the forwarding rules if the packet matches one of the cloud specific filters. Packets not forwarded by these algorithms will be forwarded using the regular VEB/VEPA algorithm. Thus the algorithms below are used to add VSIs to the OFDPort list in the regular VEB flow. So these algorithms should be inserted within the regular VEB algorithms as part of the OFDPort calculation.



Note: The promiscuous modes of the regular VEB will not capture packets forwarded by the cloud filters. Thus a packet forwarded by the cloud filter will NOT be sent to a VSI in unicast promiscuous mode.

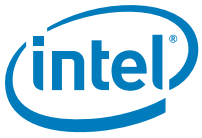
7.4.8.5.1 IP in IP, IP in GRE and MAC in GRE Cloud VEB Forwarding Algorithm

```
// Global parameters:

// Define Lookup tables - these tables are in addition to the tables described in the regular VEB
algorithm.

OuterIP Table: Array of {IP, DestinationVSI}; // This table can be matched by packets with IP in IP
or IP in GRE tunneling.

OuterIP_GRE Table: Array of {IP, GRE_Key,  DestinationVSI}; // This table can be matched by packets
with IP in GRE tunneling.

InnerMAC_VLAN Table: Array of {Inner MAC, Inner VLAN, DestinationVSI}; // This table can be
matched by packets with MAC in GRE tunneling.

InnerMAC_VLAN_GRE_: Array of {Inner MAC, Inner VLAN, GRE Key DestinationVSI}; // This table can be
matched by packets with MAC in GRE tunneling.

InnerMAC_VLAN_OuterIP Table: Array of {Inner MAC, Inner VLAN, Outer IP DestinationVSI}; // This
table can be matched by packets with MAC in GRE tunneling.

InnerMAC_GRE Table: Array of {Inner MAC, GRE Key, DestinationVSI}; // This table can be matched by
packets with MAC in GRE tunneling.

InnerMACFilter Table: Array of {Inner MAC, DestinationVSI}; // This table can be matched by
packets with MAC in GRE tunneling.

L2_filtering : Array of {Outer MAC, MembersVSIList};

// The variables used are the same as the one defined in the regular VEB algorithm. In addition the
following parameters are used:

CDport: List of VSI = {} // Cloud Filtering - empty list at startup.

OuterIp = Packet.OuterIP;

InnerMAC = Packet.InnerMAC;

InnerVLAN = Packet.InnerVLAN;

GRE = Packet.GRE_key;

VSI[i].Cloud_VSI: // Destination VSI

// Exclusive forwarding rules

// This section should be added as part of the Exclusive filters creation in the VEB algorithm:

// IP in IP or IP in GRE - outer IP filtering. Filter 0x1 in Add Cloud Filters command

For Each entry e in OuterIp Table where (OuterIp == e.OuterIp):CPort = CPort + e.DestinationVSI;

// IP in GRE - outer IP + GRE filtering. Filter 0x2 in Add Cloud Filters command

For Each entry e in OuterIP_GRE Table where (OuterIp == e.OuterIP and  GRE == e.GRE_Key):

       CPort = CPort + e.DestinationVSI; // MAC in GRE - Inner MAC/VLAN filtering. Filter 0x3 in
Add Cloud Filters command
```

```
For Each entry e in InnerMAC_VLAN Table where (InnerMAC == e. InnerMAC and InnerVLAN ==
e.InnerVLAN):

        CPort = CPort + e.DestinationVSI; // MAC in GRE - Inner MAC/VLAN/GRE Key filtering. Filter
0x4 in Add Cloud Filters command

For Each entry e in InnerMAC_VLAN_GRE Table where (InnerMAC == e. InnerMAC and InnerVLAN ==
e.InnerVLAN and GRE == e.GRE_Key):

        CPort = CPort + e.DestinationVSI; // MAC in GRE - Inner MAC filtering. Filter 0xA in Add
Cloud Filters command

For Each entry e in InnerMACFilter Table where (InnerMAC == e. InnerMAC):

        CPort = CPort + e.DestinationVSI; // MAC in GRE - Inner MAC/VLAN/Outer IP filtering. Filter
0x5 in Add Cloud Filters command

For Each entry e in InnerMAC_VLAN_OuterIP Table where (InnerMAC == e. InnerMAC and and InnerVLAN ==
e.InnerVLAN and OuterIp == e.OuterIp):

        CPort = CPort + e.DestinationVSI; // MAC in GRE - Inner MAC/GRE filtering. Filter 0x6 in
Add Cloud Filters command

For Each entry e in InnerMAC_GRE Table where (InnerMAC == e. InnerMAC and GRE == e.GRE_Key):

        CPort = CPort + e.DestinationVSI; // Update the unification of lists as follow:

// Create unified list

If (OFDPort not empty) Dport = OFDPort; // Override

Else if (CPort not empty) DPort = CPort + PMDPort; // cloud & promiscuous

Else if (PFDPort not empty) DPort = PFDPort + PMDPort; // exact & promiscuous

else DPort = IFDPort + PMDPort; //imperfect & promiscuous

}

HashFunction(MAC){

    Hash = MAC[5:0] // Use the 6 least significant bits of the MAC address (last bits on the wire).

Return Hash

}
```

7.4.8.5.2 MAC in UDP Cloud VEB Forwarding Algorithm

```
// Global parameters:

// Define Lookup tables - these tables are in addition to the tables described in the regular VEB
algorithm.

// These tables can be matched by packets with MAC in UDP tunneling.

InnerMAC_VLAN_VN_Key Table: Array of {Inner MAC, Inner VLAN, VN Key Low, DestinationVSI};

InnerMAC_Key1_key2 Table: Array of {Inner MAC, VN Key Low, VN Key High, DestinationVSI};

InnerMAC_Key Table: Array of {Inner MAC, VN Key Low, DestinationVSI};

InnerMACFilter Table: Array of {Inner MAC, DestinationVSI};

InnerMAC_VLAN Table: Array of {Inner MAC, Inner VLAN, DestinationVSI}; // This table can be
```



```
matched by packets with MAC in UDP tunneling.

// The variables used are the same as the one defined in the regular VEB algorithm. In addition the
following parameters are used:

CDport: List of VSI = {} // Cloud Filtering - empty list at startup.

InnerMAC = Packet.InnerMAC.

InnerVLAN = Packet.InnerVLAN

VNKL = Packet.VN_key Low.

VNKH = Packet.VN_key High.

VSI[i].Cloud_VSI: // Destination VSI

// Exclusive forwarding rules

// This section should be added as part of the Exclusive filters creation in the VEB algorithm:

// MAC in UDP - Inner MAC/VLAN/VN Key filtering. Filter 0x7 in Add Cloud Filters command

For Each entry e in InnerMAC_VLAN_VN_Key Table where (InnerMAC == e. InnerMAC and InnerVLAN ==
e.InnerVLAN and VNKL == e.VN_Key Low):

        CDPort = CDPort + e.DestinationVSI; // MAC in UDP - Inner MAC filtering. Filter 0xA in Add
Cloud Filters command

For Each entry e in InnerMACFilter Table where (InnerMAC == e. InnerMAC):

        CPort = CPort + e.DestinationVSI; // MAC in UDP - Inner MAC/VN Key filtering. Filter 0x6 in
Add Cloud Filters command

For Each entry e in InnerMAC_VLAN_VN_Key Table where (InnerMAC == e. InnerMAC and VNKL == e.VN_Key
):

        CDPort = CDPort + e.DestinationVSI; // MAC in UDP - Inner MAC/VLAN filtering. Filter 0x3 in
Add Cloud Filters command

For Each entry e in InnerMAC_VLAN Table where (InnerMAC == e. InnerMAC and InnerVLAN ==
e.InnerVLAN):

        CPort = CPort + e.DestinationVSI;

// Create unified list

If (OFDPort not empty) Dport = OFDPort; // Override

Else if (CPort not empty) DPort = CPort + PMDPort; // cloud & promiscuous

Else if (PFDPort not empty) DPort = PFDPort + PMDPort; // exact & promiscuous

else DPort = IFDPort + PMDPort; //imperfect & promiscuous

}

HashFunction(MAC){

    Hash = MAC[5:0] // Use the 6 least significant bits of the MAC address (last bits on the wire).

Return Hash

}
```

7.4.8.6 Cloud VEB algorithm (B-step)

// Global parameters

// Define Lookup tables

// L2 forwarding tables

MacVlan Table: Array of {MAC , VLAN , DestinationVSIList };

MAC Table: Array of {MAC , DestinationVSIList }

VLAN table: Array of {VLAN, IngressVSIList, EgressVSIList, Local}

HashMacVlan Table: Array of {HashMAC (Hash Values), VLAN, AddressType (unicast/Multicast), DestinationVSIList }

HashMac Table: Array of {HashMAC (Hash Values), AddressType (unicast/multicast), DestinationVSIList }

// Control packets forwarding tables

MacEthertype Table : Array of {MAC , Etype , Source, DestinationVSIList, Drop}

Ethertype Table : Array of {Etype , Source, DestinationVSIList, Drop}

// Cloud Forwarding tables

OuterIP: Array of {Outer IP, DestinationVSIList}; // Outer IP is valid only for encapsulated packets (Filter type = 0x1).

InnerMAC_InnerVLAN: Array of {Inner MAC, InnerVLAN, DestinationVSIList}; // This table can be matched by packets with NVGRE, VXLAN or Geneve tunneling (Filter type = 0x3).

InnerMAC_InnerVLAN_Key: Array of {Inner MAC, InnerVLAN, Key, EncapsulationType, DestinationVSIList}; // This table can be matched by packets with NVGRE, VXLANor Geneve tunneling (Filter type = 0x4).

InnerMAC__Key: Array of {Inner MAC, Key, EncapsulationType, DestinationVSIList}; // This table can be matched by packets with NVGRE, VXLANor Geneve tunneling (Filter type = 0x6).

InnerMAC: Array of {Inner MAC, DestinationVSIList}; // This table can bbe matched by packets with NVGRE, VXLAN or Geneve tunneling (Filter type = 0xA).

InnerMAC_OuterMAC_Key: Array of {Inner MAC, OuterMAC, Key, EncapsulationType, DestinationVSIList}; // This table can be matched by packets with NVGRE, VXLAN or Geneve tunneling (Filter type = 0xB).

InnerIP: Array of {Inner IP, DestinationVSIList}; // Inner IP is valid for all packets. In non encapsulated packets, it is the sole IP (Filter type = 0xC).

Promiscuous_List: Array of {VLAN, Address_type DestinationVSIList}; // Promiscuous modes per VLAN

// Define VSI modes

VSI[i].PUE: Promiscuous Unicast Enable per VSI (promiscuous VLAN only)// Target VSI

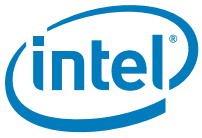
VSI[i].PME: Promiscuous Multicast Enable per VSI (promiscuous VLAN only)// Target VSI

VSI[i].BAM: Broadcast Enable per VSI (promiscuous VLAN only)// Target VSI

VSI[i].VLANPruneEnable // Target VSI - Should be cleared in case of promiscuous enable

VSI[i].AllowLoopback: Loopback Enable per VSI // Source VSI

VSI[i].MACPruneEnable: // Defines if the source VSI (identified either by SA,VLAN or by SA) should



be removed from the list of destination VSI.

VSI[i].EnableMACAntiSpoof: // Source VSI

VSI[i].EnableVLANAntiSpoof: // Source VSI

VSI[i].LANEnable // Source VSI - should be set in each VSI tied to a switch that is connected to the LAN. (VSI_SRCSWCTRL.LANENABLE). This bit is cleared if the VSI is added to a floating VEB.

VSI[i].AllowOverride // Allows override of the destination - usually set for control VSIs.

VSI[i].Cloud_VSI: // Destination VSI

// Switch generic parameters

DefaultVSI; Can be Null or one of the VSIs. // Defined as part of the S-tag table.

ControlVSI; Can be one of the VSIs or the embedded controller.

// Mirroring rules

Int n_mirror: Number of mirror rules

Mirror rule[1 .. n_mirror] = {RULETYPE type, IngressVSIList (List of VSI), EgressVSIList (List of VSI), VLANList (List of VID), MirrorPort};

Dport Switch_function(SVSI : Source VSI, Packet, FromLAN, FromHost)

// Variables

CDport: List of VSI = {} // Cloud Filtering - empty list at startup.

PFDPort: List of VSI = {} // Perfect Filtering - empty list at startup.

IFDport: List of VSI = {} // Imperfect Filtering - empty list at startup.

PMDport: List of VSI = {} // Promiscuous Filtering - empty list at startup.

OFDPort: List of VSI = {} // Override VSI Filtering - empty list at startup.

VFDPort: List of VSI = {} // VLAN membership VSI Filtering - empty list at startup.

MNGPort: List of VSI = {MDEF filtering result as defined in Section 10.3}

MNGOnly: Bool = {MNGonly result as defined in Section 10.3}

DPort: List of VSI = {} // Final List - empty list at startup. Can include also the LAN port for Tx packets.

PassAntiSpoof = False;

PassMACAntiSpoof = False;

PassVLANAntiSpoof = False;

//Define packet parameters

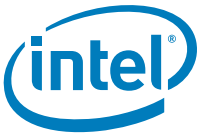
DA = Packet.DA: Destination Address of the packet

SA = Packet.SA: Source Address of the packet

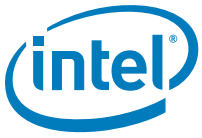
VID = Packet.VLAN ID: Vlan tag of the packet - equal to zero if untagged packet.

Etype = Packet.Ethertype: Ethertype of the packet

AddressType: Type of address of the DA. Can be Unicast, Multicast or Broadcast.

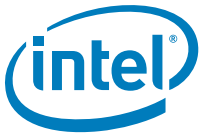



```
HDA = HashFunction(DA).

OuterIP = Packet.OuterIP // Relevant only for encapsulated packets.

InnerIP = Packet.InnerIP // Inner IP is valid for all packets. In non encapsulated packets, it is
the sole IP.

InnerMAC = Packet.InnerMAC;

InnerVLAN = Packet.InnerVLAN;

Key = Packet.NVGRE TNI key or Packet.VXLAN VNI key or Packet.Geneve VNI Key;

EncapsulationType = NVGRE, VXLANGeneve;

PSize: Packet size. This do not include any tag or other header removed by previous stages or to be
added by following stages.

LocalVLAN = False: Define if the packet is part of a local VLAN. See below for calculation.

Destination: // Can be either uplink (send only to LAN), local (switch decides on destination, but
do not send to LAN), switchBased (regular switching algorithm) or a TargetVSI.  This is a
descriptor bit for packets received from a control port allowing override of the switching
decision. This may be useful when sending switch generated packets like DCBX packets. Relevant
only for transmit packets

Source = Tx or Rx // defines if this is a packet sent by the XL710 or received by the XL710.

// This logic creates the following indications to be used later

Specific_Port = NULL; // A specific port defined as the sole destination of the packet.

SwitchToLAN = TRUE; // Allow forwarding to LAN

SwitchToLocal = TRUE; // Allow forwarding to local VSI.

else if (Destination == TargetVSI and SVSI.AllowOverride) {Specific_Port = Target VSI from
Descriptor};

if ((Destination == UpLink and SVSI.AllowOverride) or DoNotLoopback or SVSI.ALLOWLOOPBACK ==
FALSE) SwitchToLocal = FALSE;

if ((Destination == UpLink and SVSI.AllowOverride) {Specific_Port = LAN};

if ((Destination == Local and SVSI.AllowOverride) or SVSI.LANDisable == TRUE) SwitchToLAN = FALSE;

If (FromLan) SPort = LAN else SPort = SVSI.

// Anti Spoofing

// Local VLAN calculation

For each v in VLAN Table {if (v.VLAN == VID and v.LocalVLAN == True) LocalVLAN = TRUE}

// Need to check if the packet can enter the switch before applying all the rules

If (SVSI.EnableMACAntiSpoof == True and FromHost) {

// Only perfect match filters are used for anti spoofing checks

        For Each entry e in MacVlan Table

        if (SA == e.MAC and (VID == e.VLAN or (VID == NULL and e.VLAN == 0)) and SVSI in
e.DPortList ) PassMACAntiSpoof = True

    For Each entry e in Mac Table
```



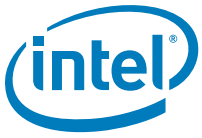
```
        if (SA == e.MAC and SPort in e.DestinationVSIList ) PassMACAntiSpoof = True
    } else PassMACAntiSpoof = True;
// The step below (VLAN anti spoof) is really ingress VLAN filtering
If (SPort.EnableVlanAntiSpoof == True and FromHost) {
    For Each entry e in  VLAN Table
    // Note - an untagged packet will be compared with an entry of the table with VLAN = 0.
        if (VID == e.VLAN and SVSI in e.IngressVSIList) PassVLANAntiSpoof = True;
} else if (FromLAN and LocalVLAN == True) {
    PassVLANAntiSpoof = FALSE;
} else PassVLANAntiSpoof = True;
    PassAntiSpoof = PassVLANAntiSpoof and PassMACAntiSpoof;
If (PassAntiSpoof == False)  {DPort = null; Goto Mirroring Rules }// Drop Packet as DPort is empty;
// Switching algorithm
// Perfect Filters
For Each entry e in MacVlan Table {
    If (DA == e.MAC and (VID == e.VLAN or (VID == NULL and e.VLAN == 0)))
    PFDPort = PFDPort + e.DestinationVSIList; // Add VSI matching the MAC, VLAN pair.
}
For Each entry e in Mac Table { :
    If ((DA == e.MAC )
        PFDPort = PFDPort + e.DestinationVSIList; // Add ports matching the MAC only entry.
}
// Imperfect Filters
    For Each entry e in HashMACVlan Table where (HDA == e.HashMAC and (VID == e.VLAN or (VID ==
NULL and e.VLAN == 0 and AddressType == e.AddressType))):
        IFDPort = IFDPort + e.DestinationVSIList; // Add VSIs matching the HashMAC, VLAN pair.
    For Each entry e in HashMAC Table where (HDA == e.HashMAC and AddressType == e.AddressType):
        IFDPort = IFDPort + e.DestinationVSIList;// Add VSIs matching the HashMAC.
}
// Apply promiscuous modes
For each port p in VEB {
    If (AddressType == Unicast and p.PUE == 1) PMDPort = PMDPort + p;
    If (AddressType == Multicast and p.PME == 1) PMDPort = PMDPort + p;
    If (AddressType == Broadcast and p.BAM == 1) PMDPort = PMDPort + p;
```

```
}

// Apply per VLAN promiscuous modes

For Each entry e in Promiscuous_List Table where (VID == e.VLAN and AddressType == e.AddressType):

PMDPort = PMDPort + e.DestinationVSIList; // Add VSIs matching the promiscuous list.

// Cloud filters

// Outer IP filtering. Filter 0x1 in Add Cloud Filters command

For Each entry e in OuterIp Table where (OuterIP == e.OuterIP):

        CDPort = CDPort + e.DestinationVSI;

// Inner MAC/VLAN/Key filtering. Filter 0x4 in Add Cloud Filters command

For Each entry e in InnerMAC_InnerVLAN_Key Table where (InnerMAC == e. InnerMAC and InnerVLAN ==
e.InnerVLAN and Key == e.Key_Key Low and EncapsulationType = e.EncapsulationType):

        CDPort = CDPort + e.DestinationVSI;

// Inner MACfiltering. Filter 0xA in Add Cloud Filters command

For Each entry e in InnerMAC Table where (InnerMAC == e. InnerMAC):

        CPort = CPort + e.DestinationVSI;

// Inner MAC/Key filtering. Filter 0x6 in Add Cloud Filters command

For Each entry e in InnerMAC_VLAN_Key Table where (InnerMAC == e. InnerMAC and Key == e.Key and
EncapsulationType = e.EncapsulationType):

        CDPort = CDPort + e.DestinationVSI;

// Inner MAC/VLAN filtering. Filter 0x3 in Add Cloud Filters command

For Each entry e in InnerMAC_VLAN Table where (InnerMAC == e. InnerMAC and InnerVLAN ==
e.InnerVLAN):

        CPort = CPort + e.DestinationVSI;

// Outer MAC/Key/Inner MAC filtering. Filter 0xB in Add Cloud Filters command

For Each entry e in InnerMAC_OuterMAC_Key Table where (InnerMAC == e.InnerMAC and OuterMAC ==
e.MAC and Key == e.Key_Key Low and EncapsulationType = e.EncapsulationType):

        CPort = CPort + e.DestinationVSI;

// Inner IP filtering. Filter 0xC in Add Cloud Filters command

For Each entry e in IP Table where (InnerIP == e.InnerIP):

        CDPort = CDPort + e.DestinationVSI;


// Override filters

For Each entry e in Ethertype Table where (Etype == e.Etype and Source == e.Source):

    OFDPort = OFDPort + e.DPortList; // Control port filtering.

For Each entry e in MacEthertype Table where (DA == e.MAC and Etype == e.Etype and Source ==
e.Source):
```



```
    OFDPort = OFDPort + e.DPortList; // Control port filtering.

// Create unified list

If (MngOnly) DPort = MNGPort;

Else If (OFDPort not empty) Dport = OFDPort + MNGPort; // Override

Else if (CPort not empty) DPort = CPort + PMDPort; // cloud & promiscuous

Else if (PFDPort not empty) DPort = PFDPort + PMDPort + MNGPort; // exact & promiscuous

else DPort = IFDPort + PMDPort + MNGPort; //imperfect & promiscuous

if (DPort is empty and FromLAN and DefaultVSI != Null)

    DPort = DefaultVSI;

DPort_for_mirroring = Dport; // mirroring ignores VLAN prunning - should be fixed in future
project

// VLAN egress Filtering

// Note - to use VLAN only fitering (ignore all MAC addresses), the UPE, MPE and BAM bits should be
set in all ports. See Section 7.4.6.1.4 for rules to set the egress and ingress VLAN lists

For each entry p in DPort {

    if (p.VLANPruneEnable == True) {

        If (VID match VLAN in VLAN table) {

            for each e in Egress Vlan Table where (VID == e.VLAN or (VID == NULL and e.VLAN == 0))

                if (p NOT in e.EgressVSIList) DPort = DPort - p; // prune ports which are not member
of the VLAN;

        }

        Else Dport = Dport - p; // Remove all ports which are not in Promiscuous VLAN if VLAN not
found in VLAN table.

    }

}

// Add external ports and default port if needed

// Note: In VEPA mode, all Tx packets should go to LAN only (VSI[i].AllowLoopback == 0)

if ((PFDport is empty or AddressType == Multicast or AddressType == Broadcast or
VSI[i].AllowLoopback == 0) and OFDPort is empty and FromHost and LocalVLAN == FALSE and
SVSI.EnableLAN) {

    DPort = Dport + LAN port;

    DPort_for_mirroring = Dport_for_mirroring + LAN port;

}

// Prunning Rules

// Control Packets drop rules.

For Each entry e in Ethertype Table where (Etype == e.Etype and Source == e.Source):

        if (e.Drop == True) DPort = Null;DPort for mirroring = NULL;}
```

```
    For Each entry e in MacEthertype Table where (DA == e.MAC and Etype == e.Etype and Source ==
e.Source):

        if (e.Drop == True) DPort = Null;DPort for mirroring = NULL;}

// Find the source port based on the source MAC address.

// The algorithm allows multiple source port, but the assumption is that a unicast address will
point to a single VSI.

For Each entry e in Mac Table

        if (SA == e.MAC ) SourcePorts = e.DestinationVSIList; For Each entry e in MacVLAN Table

        if (SA == e.MAC and VLAN = e.VLAN ) SourcePorts = SourcePorts + e.DestinationVSIList;

For Each port p in Dport {

        // Remove ports from list if loopback disabled for this port

    if (From Host and SwitchToLocal == FALSE and p != LAN Port) {

        DPort = Dport - p;

        DPort_for_mirroring = DPort_for_mirroring - p;

}

    // Remove uplinks  from list if requested by descriptor

    if (p == LAN Port and SwitchToLAN == FALSE) {

    Dport = Dport - p ;

    DPort_for_mirroring = DPort_for_mirroring - p;

}

    // Prune Source port

    if (p.MACPruneEnable== True and ( p in SourcePorts) DPort = Dport - p;

// Prune by size is done at the queue level, so this is not handled in the switch.

}

// L2 filtering rule

// Checks that the packet source MAC address was added by one of the MAC address based filters
(Either initial PF MAC address, Add MAC VLAN AQ command, Add Ethertype AQ command or filer 9 in Add
Cloud Filters.

if (DA does not match any entry in MAC table and L2 filtering is enabled) {

    For each entry p in DPort {

        if (p in CPort) {  DPort = DPort - p; // Prune ports that received the packet due to the
cloud filters anddo not match any MAC address;

        }

    }

}

// Mirroring rules
```



```
MatchedRuleID = Null;
for ( i = 1 to n_Mirror) { // Apply all mirroring rules
// Note - a port can be member only in a single VSI based mirror rule.
    Bool Mirror = FALSE;
    Bool VLAN_Mirror = FALSE;
If(FromHost) {
    Case Mirror rule[i].type {
    ALL_EGRESS {
    If (DPort_for_mirroring not empty) Mirror = True;
    };
    ALL_INGRESS {
        Mirror = True;
    };
    VSI_INGRESS
    // This rule is applied even if packet is droped.
    if (SPort in Mirror rule[i].IngressPortList) Mirror = True;
};
    VSI_EGRESS {
        For each p in Dport_for_mirroring {
            if (p in Mirror rule[i].EgressPortList) Mirror = True;
        }
    }
}
If (FromLan) {
    Case Mirror rule[i].type{
        ALL_EGRESS {
        Mirror = True;
        };
        VSI_EGRESS {
        For each p in Dport_for_mirroring {
            if (p in Mirror rule[i].EgressPortList) Mirror = True;
        }
    };
};
```

```
    }   if (Mirror rule.ty[e= VLAN_MIRROR and VID in Mirror rule.VLANList and DPort is not empty)
VLAN_Mirror = True;

    if (Mirror) {

        DPort = Dport + Mirror rule[i].MirrorPort;

        MatchedRuleID = i; // Only a single non VLAN rule should match

    }

    if (VLAN_Mirror) DPort = Dport + Mirror rule[i].MirrorPort;

}

If (Specific_Port != NULL) DPort = Specific_Port; // Descriptor from control port can override the
entire switch decision

Return Dport // Note - the Dport list, should include a single copy of each VSI.

}

HashFunction(MAC){

Hash = MAC[6:0] // Use the 7 least significant bits of the MAC address (last bits on the wire).

Return Hash

}
```

7.4.9 Switch Programming Model

7.4.9.1 Switching Structure Representation

This section describes the model used to represent the switching elements.

The objects represented in the models are identified by a 10 bits device wide switch element identifier (SEID). The SEID of an element is returned upon creation of the element and should be referenced when creating ties between elements. The SEID of elements created automatically can be retrieved using the *Describe Structure* command.

The objects represented can be either external or internal to the switch. Possible external elements are:

Table 7-60. External Elements

External Element	Abbreviation	Element Type	Description
Physical port MAC	MAC	1	This element is always present. An SEID is allocated at power on for each enabled port.
Physical functions	PF	2	Physical function elements is created at PCIe reset for each enabled PF.

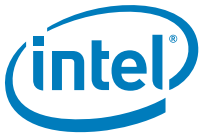


Table 7-60. External Elements

External Element	Abbreviation	Element Type	Description
Virtual functions	VF	3	Virtual function elements are created when SR-IOV is enabled on a physical function.
Embedded Micro Processor	EMP	4	The Embedded Micro Processor can be used as a control port for the different switching elements. All the ties to the EMP should be explicitly created when creating an element. An EMP can contain multiple EMP Queues that are only used to represent logical connections to the switch. This is used to enable multiple control ports in the EMP. This element includes the BMC interface used to allow pass through traffic to an external BMC via an out of band connection.

The internal elements are elements that are used as part of the switching decision. The following elements are available:

Table 7-61. Internal Elements

Internal Element	Abbreviation	Element Type	Description	Related commands	Described in section
Port Virtualizer (S-comp or Port Extender)	PV	16	Defines an S-comp or a Port Extender. Together with a PV, create a set of S-channels elements	Add Port Virtualizer Update PortVirtualizer Parameters Get Port Virtualizer Parameters	7.4.4.2
S-channel	SC	N/A	An S-channel created as part of an S-comp/M-comp element. There is no separate S-Channel element. An S-channel connected directly to an external element (PF/VF/EMP) is considered as a VSI. An S-channel connected to a VEB or a VEPA is considered as part of the VEB/VEPA.	N/A	7.4.4.2
L2 Filter	L2	N/A	An L2 filter element. Such an element is created by default to connect a MAC and the first PF tied to this MAC. An L2 is not created as part of the topology. Rather it is implemented using the filtering options to the VSI tied to it.	N/A	7.4.4.3
VEB/Port Aggregator	VEB/VEPA	17	A Virtual Ethernet Bridge or a Virtual Port Aggregator.	Add VEB Get VEB Parameters	7.4.4.4
Virtual Station Interface	VSI	19	A VSI can be part of a VEB, VEPA or PV element and is connected to an host external element (PF, VF or EMP).	Add VSI Update VSI parameters Get VSI Parameters Delete Element	7.4.5.2

The SEIDs are allocated according to the following table:

Table 7-62. SEID Mapping

SEID	Associated Element
0	Non valid connection. When used as an Uplink, indicates the network (no internal uplink). When used as a downlink indicates the host (no internal downlink)
1	EMP
2-5	Port 0-3 MAC
6-15	Reserved
16-31	PF 0-15
32-159	VF 0-127
160-287	Reserved
288 - 511	Internal structures (VEB/SC/L2/VEPA/PV)
512-895	VSI 0-383 (including EMP VSIs)
896-1023	Reserved.

The following table is used to represent the switching hierarchy and should be used by the embedded firmware and the host software as a common database:

Table 7-63. Switching Structure Representation

Entry in Table	Description	Notes
SEID	The handle to the element	
Element Type	As defined in Table 7-60 and Table 7-61	
Up Link Connection(s)	Indicates the SEID of the switching element connected directly below the current element. Below means towards the network in this case.	For sub elements of a composed switch element, the uplink is the containing element
Down Link Connection(s)	Indicates the SEID of the switching element connected directly above the current element. Above means towards the host in this case.	The down link is provided only for elements connected to PFs, VFs or to the EMP. In other cases, the value is zero.
Control Port	For elements with a control port - points to the control port.	
Scheduler Node	The scheduler node associated with this switch element	

Table 7-64 shows an example of the representation of a switching configuration as described in Figure 7-47.

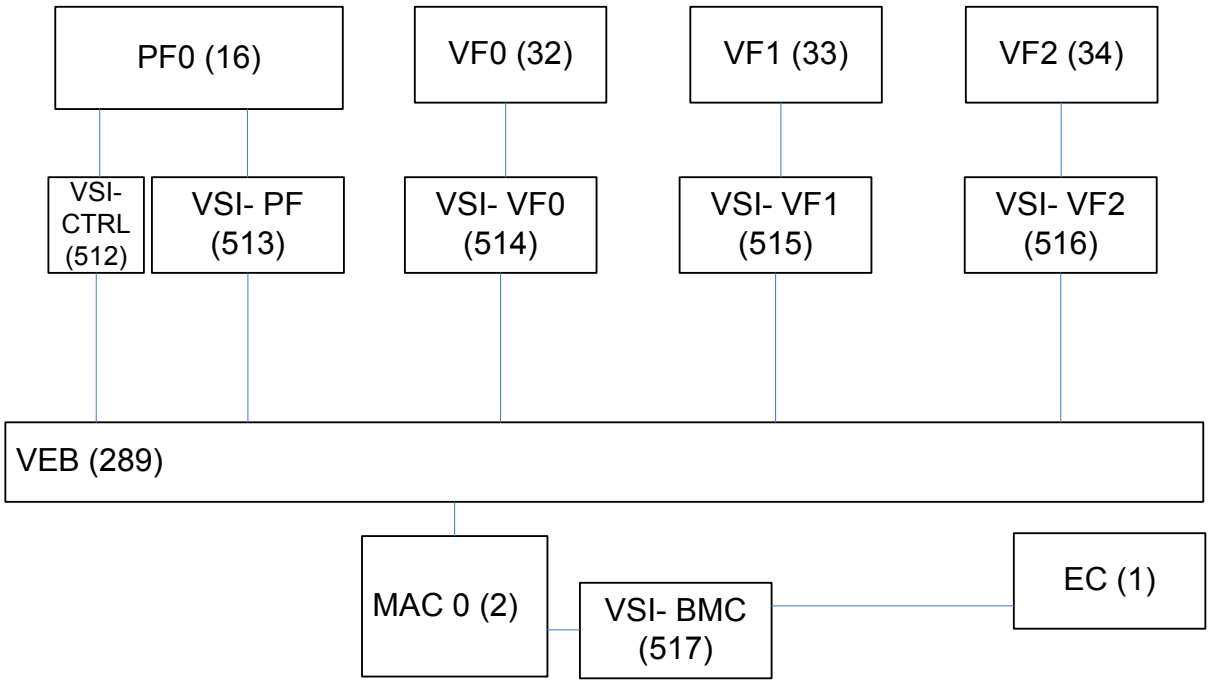


Figure 7-47. Switching structure Example

Table 7-64. Switching Structure Representation - Examples

SEID	Description	Element Type	Up Link	Down Link
0	Null	Null	0	0
1	Embedded controller	EMP	517	0
2	Port 0 MAC port	MAC	0	0
16	Physical function 0	PF	512	0
32	VF0	VF	514	0
33	VF1	VF	515	0
34	VF2	VF	516	0
289	VEB	VEB	2	0
512	VSI (PF)	VSI	289	16
513	VSI - Control VSI for VEB	VSI	289	16
514	VSI (VF0)	VSI	289	32
515	VSI (VF1)	VSI	289	33
516	VSI (VF2)	VSI	289	34
517	VSI (EMP)	VSI	2	1

7.4.9.2 Supported Switch Profiles

This section describes the supported switching profiles.

The profiles described are per port.

The following profiles are supported by the XL710:

1. Basic L2 (default configuration)
2. VEB/VEPA only
3. Port Virtualizer
4. Port Virtualizer + VEB/VEPA

Note: There is no central location in which a profile is set. These profiles are examples of possible configuration that are expected in the XL710. The configuration of each profile is done using the admin commands described in Section 7.4.9.4.2

7.4.9.2.1 Basic L2

A basic L2 configuration is used for monolithic OSes (non virtualized) systems where the network is not distributed to virtual channels. In this case, the switch configuration is as follow:

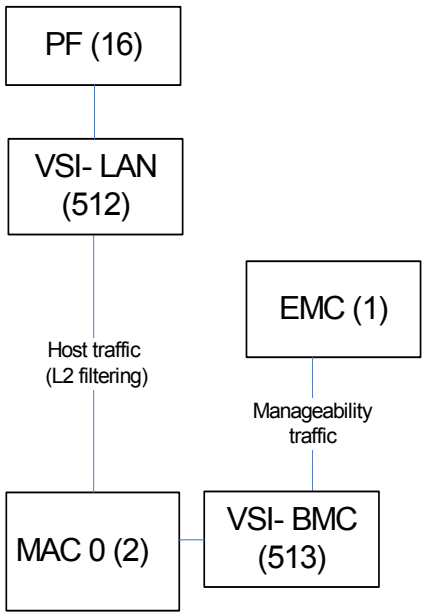


Figure 7-48. Switch Configuration - Basic L2

7.4.9.2.1.1 Basic L2 - Resource Allocation

In this profile, all the resources allocated in the NVM to the port are available to this single function.

7.4.9.2.2 VEB/VEPA

A VEB configuration is used for virtualized OSes systems where the network is not distributed to virtual channels. The switch configuration for the VEB/VEPA case is described in the figure below.

The use cases for VEB and VEPA are described in Section 7.4.2.1.2 and Section 7.4.2.2.1 respectively.

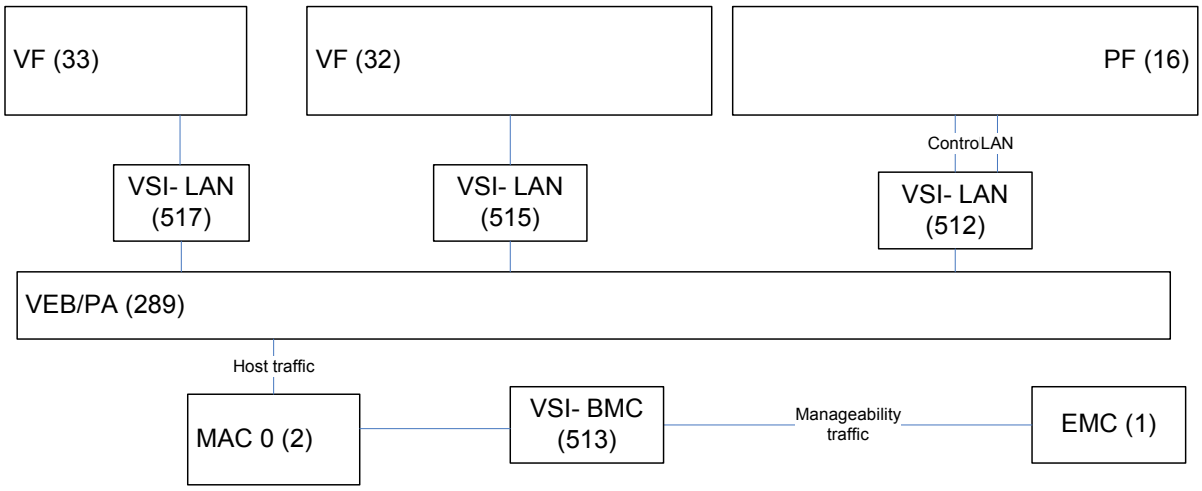


Figure 7-49. VEB/VEPA

Note: The control port of the VEB may be either a separate VSI or part of the regular VSI of the PF.

7.4.9.2.2.1 VEB - Resource Allocation

There is no limitation on the number VSIs on a single VEB (apart from the total numbers of VSIs supported) and up to 4 VEBs connected to a single PF.

The switching resources are allocated to the PF. The PF can decide to allocate the resources to any of the VEBs or to any of the VSIs according to its own policy.

7.4.9.2.3 Port Virtualizer

This mode is used when all the switching decisions are done in an external switch and each VSI is connected directly to one or more virtual channels. The usage models of the Port Virtualizer profile are described in Section 7.4.2.2.2. The switch configuration for the Port Virtualizer case is described in the figure below. In the case of a VF requiring multiple VSIs, it may request a new VSI using the *Add VSI* command. This command will be forwarded to the PF for completion.

Note: The control port of the Port Virtualizer may be either a separate VSI or part of the regular VSI of the PF. In this case, the tagging of packets is controlled by the host. Packets sent from the control port should be non tagged.

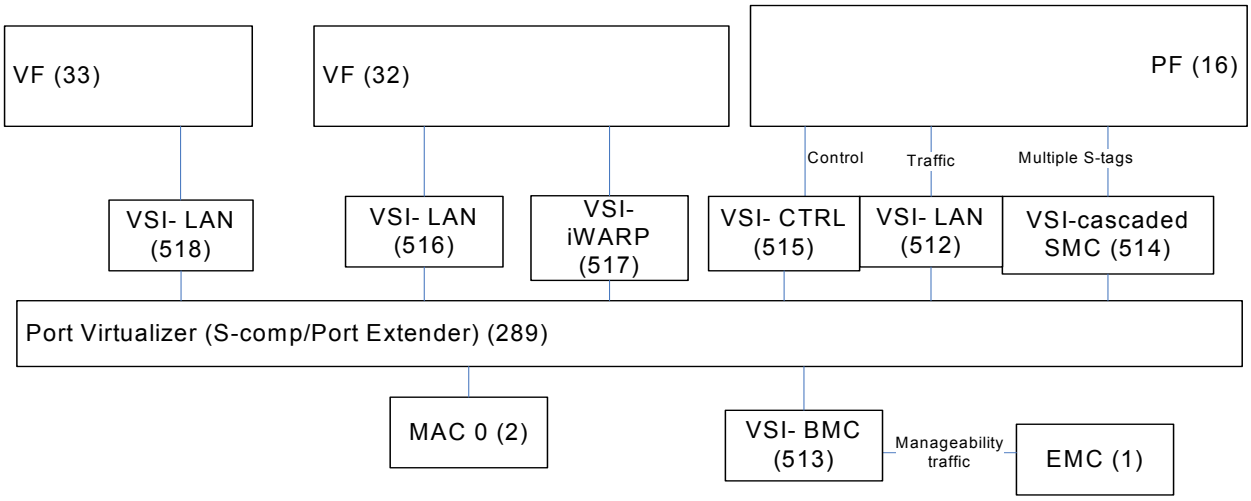


Figure 7-50. Port Virtualizer

7.4.9.2.4 Port Virtualizer + VEB/VEPA

A Port Virtualizer +VEB/VEPA configuration is used for virtualized OSes systems where the network is distributed to virtual channels. The switch configuration for the Port Virtualizer +VEB/VEPA case is described in Figure 7-51. In the case of a VF requiring multiple VSIs, it may request a new VSI using the *Add VSI* command. This command will be forwarded to the PF for completion.

Note: The control port of the Port Virtualizer in this mode is the EMP, as there are multiple PFs connected to the same Port Virtualizer.

A special case of this mode is the mode where the link is divided to virtual channels and each channel is connected to a physical function, but none of these are virtualized. In this case, there are no VEB/PAs in the device.

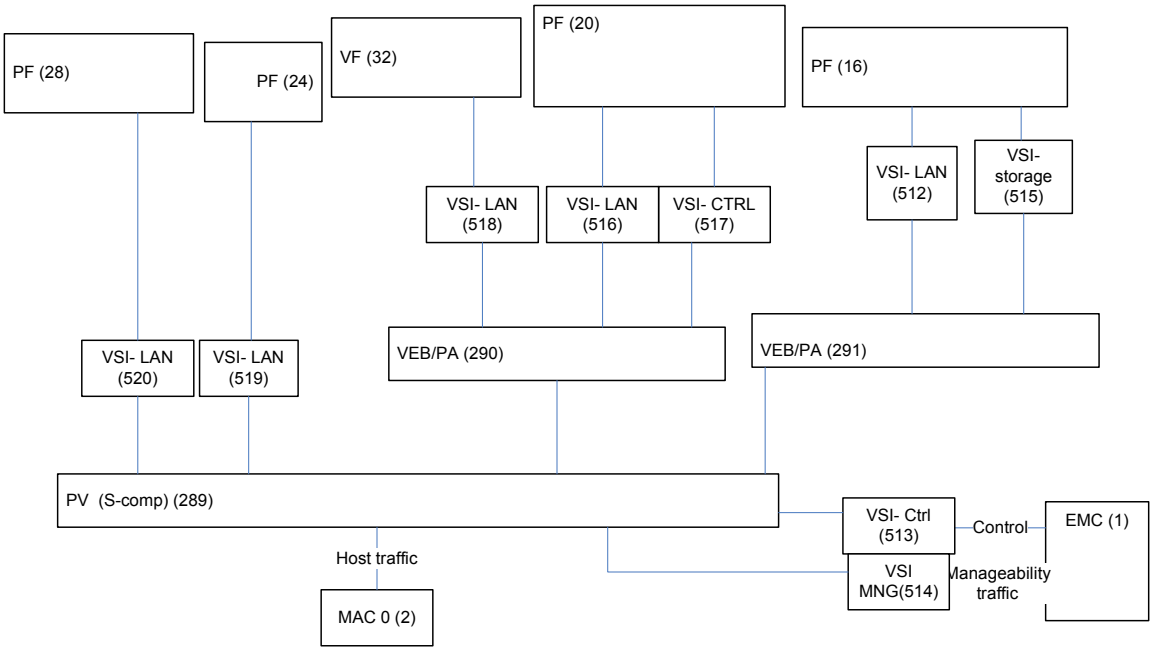


Figure 7-51. Port Virtualizer + VEB/VEPA

7.4.9.2.5 LAN/SAN S-comp

See Section 7.4.2.1.2 & Section 7.4.4.2.3 for more detail.

The initial configuration as exposed by the device is depicted in Figure 7-52, "LAN/SAN initial configuration".

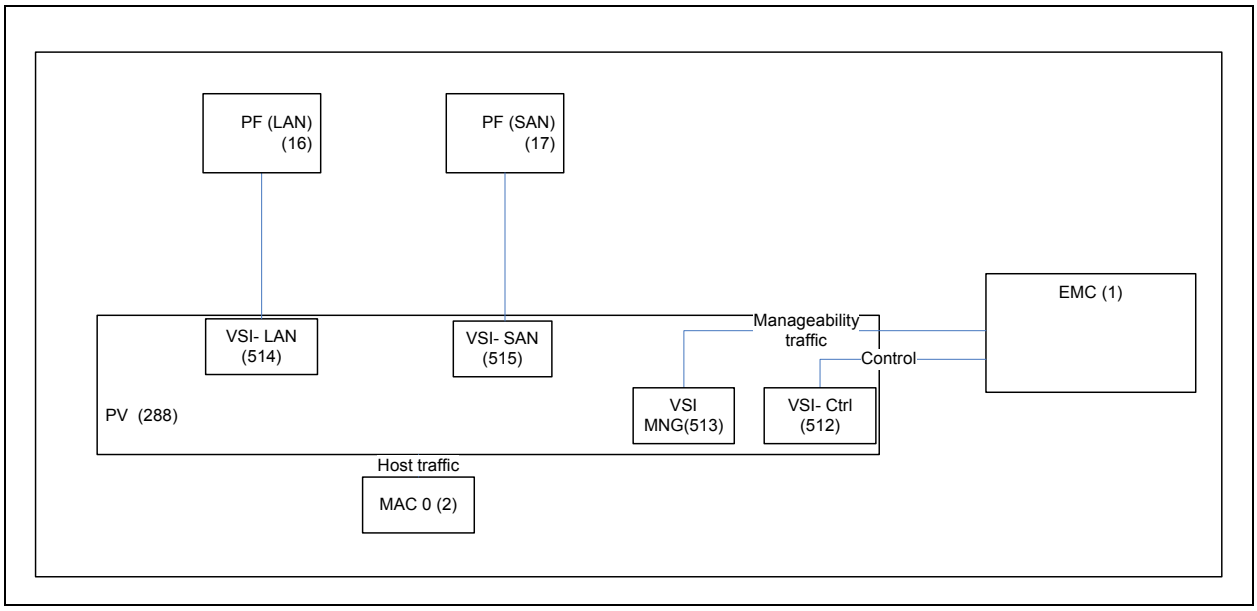


Figure 7-52. LAN/SAN initial configuration

7.4.9.3 Switch Resource Allocation

7.4.9.3.1 Allocatable Resources

The tables below describe the resources that are part of the switch and should be allocated to the different switch elements.

Table 7-65 describes the Switching elements resources. These resources are available no matter what the configuration of the forwarding tables is. Table 7-66 describes the forwarding rules resources. These resources are different for different configuration of the forwarding tables.

Note: Each element is associated with a port, so two ports using the same value (for example the same MAC address) will consume two entries.

Table 7-65. Switching Elements resources

Resource	Total Available	Notes
VEBs	16	There is no limitation on the number of VEBs assigned to a PF.
VSIs	384	Up to 12 of the 384 VSIs are reserved for EMP traffic (LLDP and pass through traffic forwarding per port) and should not be allocated.
VSI List entries	6144	3 VSI in each entry. These entries are used to create up to 2048 VSI lists such as multicast VSI lists or VLAN membership lists.
E-tag List entries	2048	3 S-tags in each entry. These entries are used to create up to 512 E-tag multicast lists .
VLAN Statistic pools	128	
TC Statistic pools	128	Statically associated with the 16 VEBs (8 sets for 8 TCs for each of the VEBs)

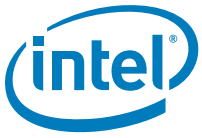


Table 7-65. Switching Elements resources

Resource	Total Available	Notes
Mirror rules	64	
Queue sets	1024	
Teredo UDP ports	16	

Table 7-66 describes the resources allocated according to the filters configuration used. The image used is defined in the *Features enable.Switching mode* NVM field.

Table 7-66. Switching rules resources

Resource	Total Available				Notes
NVM Map	**EVB switch (A0)**	**MAC in UDP Cloud (A0)**	**Other Clouds (A0)**	**Unified Image (B0)**	
First level filters					
Perfect match MAC addresses	1536	1535		1535	Each MAC address can be used for multiple VEBs within the same port. There can be up to 2048 VEB,MAC associations. One MAC address is statically allocated to the EMP
VLANs	512	256			The entire range of 4096 VIDs can be covered; however, only 256 different VLANs can be stored in the forwarding tables. Each VLAN can be used in multiple VEB within a port. There can be 1024 VEB, VLAN associations.
Ethertype filters	256	64		64	These filters are tuples of switch ID, Ethertypes and direction.
S-tags	512	448 1024			The entire range of 4096 S-VIDs can be covered; however, only 448 different S-tags can be stored in the forwarding tables.
Promiscuous rules					Unicast, broadcast or multicast promiscuous for different VEBs.
VLAN Promiscuous					The entire range of 4096 VIDs can be covered; however, only 1024 different VLANs/ packet types (unicast, multicast, broadcast) /VEB can be stored in the forwarding tables.
Multicast hash entries	512	1024	512	512	
Unicast hash entries	512		512	512	
Inner VLAN	N/A		N/A	512	Inner VLAN used for UDP cloud filtering.
VN Key Low / GRE Key/Tenant ID	N/A	512			
Inner MAC	N/A	496		1024	Each such filter added occupies an entry in the table shared with the inner MAC, inner VLAN, Tenant ID filters
Destination IPv4 / IPv6 Address (application IP)	N/A	128		256	Each such filter added occupies an entry in the table shared with the inner MAC, inner VLAN filters
Outer IP		N/A	N/A		Each such filter added occupies an entry in the table shared with the MAC-Ethertype filters

Table 7-66. Switching rules resources

Resource	Total Available				Notes
NVM Map	EVB switch (A0)	MAC in UDP Cloud (A0)	Other Clouds (A0)	Unified Image (B0)	
MAC,VLAN pairs	2048	2048	2048	2048	For each entry in this table, an entry in the Perfect match MAC addresses and a VLAN entry is used. The same is true for all the other types of filters.
Hash MAC,VLAN pairs	2048	2048		1024	For each entry in this table, an entry in the hash MAC addresses and a VLAN entry is used. The same is true for all the other types of filters.
MAC-Ethertype filters	512	512		1024	Each entry is associated with an S-tag or switch ID. For each entry in this table, an entry in the Perfect match MAC addresses and a Ethertype filters entry is used.
MAC L2 filtering	N/A	512		512	This filter shares the same entries as the perfect Match MAC address.
Inner MAC, Tenant ID	N/A			1024	
Outer MAC, Tenant ID, Inner MAC					
Inner MAC, Inner VLAN, Tenant ID	N/A	2048	N/A	1024[1]	For each entry in this table, an entry in the Inner MAC addresses, in the inner VLAN and a Tenant ID filters entry is used.
Inner MAC, Inner VLAN pair	N/A	N/A	512	1024[2]	For each entry in this table, an entry in the Inner MAC addresses, in the inner VLAN filters entry is used.
Inner MAC, inner VLAN, Outer IP tuple	N/A	N/A	1024		For each entry in this table, an entry in the Inner MAC addresses, in the inner VLAN and an outer IP filters entry is used.
Inner MAC, inner VLAN, MAC tuple	N/A	N/A	512		For each entry in this table, an entry in the Inner MAC addresses, in the inner VLAN and a perfect Match MAC address filters entry is used.

1. Shared between Inner MAC, Inner VLAN, Tenant ID filters and Inner MAC filters.
2. Shared between Inner MAC, Inner VLAN filters and inner IP filters.

7.4.9.3.2 PFs Allocation Method

The resources can be categorized as shared resources that are used by multiple PFs and resources that are allocated to a single PF. Resources used by multiple PFs are inherently shared and thus can not be allocated to any PF. Each PF requesting a resource will get it given there are still empty entries in the shared pool. It is the responsibility of each driver to release resources it doesn't use.

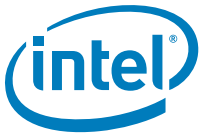
For resources allocated to a single PF, there is a basic allocation in the NVM for each PF (that can be zero). The resources not reserved to a PF are allocated on a first come first served basis.

The *PF allocations* structure in the NVM contains the resource allocation initial values.

If, after the enumeration stage, a PF is not enabled, its resources are given back to the shared pool.

Resources are allocated first from the dedicated pools and only if the dedicated pool of a PF is exhausted, a resource from the shared pool is given.

The resources that can be dedicated to a PF are:



- VEBs (TCs statistics pools are also allocated as part of the VEB).
- VSIs
- Perfect match MAC addresses
- S-tags
- VLAN Statistic pools
- Mirror rules
- Queue sets

All the other resources are considered as shared.

Note: The response of commands used to allocate dedicated resources includes information about the dedicated and shared resource left after the command is applied. The response of commands used to allocate shared resources includes information about the shared resource left after the command is applied.

The *Get Switch Resources Allocation* command returns the reserved allocation of each type of resource and the current level of usage of each resource.

7.4.9.3.2.1 Statistics Allocation

There are a few sets of statistics used by the switch as described in Section 7.11.5. The allocation of statistics resources to the different switching elements is as follow:

- For each VSI, a set of per VSI statistics is allocated. The set to use is returned in the *Add VSI* response buffer in the *statistic counters* field. See Section 7.11.5.1 for details.
- For each port and Port Virtualizer, a set of per port/Port Virtualizer statistics accessed using per port registers is allocated as described in Section 7.11.5.1.
- For each VEB allocated to a function, a set of VEB uplink statistics and per VEB per TC statistics are allocated as described in Section 7.11.5.1 and Section 7.11.5.2. The set to use is returned in the response of the Add VEB command in the *statistic counter* field.
  - In addition, a VEB can be allocated a set of statistics per VLAN per VEB using the *Add Statistics* admin command. The set to use is returned in the *statistic counters* field in the response of this command. This request may fail if there are no free sets.

7.4.9.3.3 VFs Allocation Method

XL710 manages VF resources as part of the PF resources. The PF driver should manage the VF requests based on the PF allocation.

7.4.9.4 Switch Init Flow

The init flow of the switch is composed of the following steps:

1. Pre BIOS stages:
   a. Initialization of the switch hardware (Section 7.4.9.4.1).
   b. Initialization of a basic switch configuration according to NVM (Section 7.4.9.4.2).
   c. Define the supported virtualization modes.
   d. Update from external configuration source like EVB management protocol.
2. BIOS configuration (Section 7.4.9.4.3)

a. Update of the switch configuration according to SMASH CLP.
b. Configuration of switch according to the updated configuration.

3. Software Configuration (Section 7.4.9.4.4)
   a. Optionally, creation of VEB or VEPA elements by VMM.
   b. Allocation of MAC addresses and VLANs to VSIs.

At the end of stage1, basic connectivity to the pass through management traffic and the EMP is enabled. In addition WoL functionality is also available. If already known, the connectivity to Physical functions is also provided.

At the end of stage2, connectivity to the host is added This is the configuration exposed to the pre boot driver and to the Operating system.

7.4.9.4.1 Initialization of Switch Hardware

The switch is implemented using a programmable logic, allowing parsing of different packet types and implementation of different forwarding rules. This programmable hardware is configured as part of the the XL710 auto load of the CORE Registers Auto-load NVM section. The lookup tables and the switching rules logic are configured at this stage.

This stage is part of the Auto-load 2 initialization stage described in Section 4.2.1.3.

This stage is activated at each core reset.

7.4.9.4.2 Initial Switch Configuration

The internal EMP firmware defines the initial switch structure. This stage is done after all the hardware auto-load is done and the number of currently enabled PCIe PF functions is known. Before any NVM configuration the following elements are defined by the Firmware:

Table 7-67. Initial Elements

Elements	SEID	Note
EMP	1	
Ports	2-5	Only for the ports enabled
Physical functions	16-31	Only for the functions enabled
Virtual functions	32-159	For all VFs. VFs not used will not be connected to VSIs.

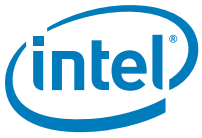
At this stage, the firmware loads the switch configuration from NVM. The *EMP Settings* NVM module contains the entire configuration needed. It contains the following information relevant to the switch operation:

- A list of capabilities supported by the switch (type of filtering, types of connections).
- PF allocations module (Section 7.4.9.4.2.2).

The *EMP Settings Module* is described in Section 7.2.1.14.

This stage is part of the Firmware initialization stage described in Section 4.2.1.4.

This stage is activated at each core reset.



7.4.9.4.2.1 Initial VSIs

At init time, only minimal forwarding capabilities are needed:

- Forwarding of pass through traffic to manageability interface
- Forwarding of LLDP traffic to the EMP.
- Wake on LAN detection.

The initial switching configuration includes only the forwarding rules needed to support these flows. At this stage, the host is not connected to the network and can not get traffic. If additional pre boot configuration is not expected, then the connectivity to the host is also done at this stage.

The basic configuration for port connectivity is achieved by connecting a single VSI to each port to one Physical function. The connection is from the port to the function connected to it with the lowest function number. The mapping of functions to port is reflected in the *PFGEN_PORTNUM.PORT_NUM* field. These VSIs are created with a *Regular Data Port* connection type and a PF VSI type.

All the VSIs created at this stage have the default switch ID of the port. The following parameters should be set to a non default value in these VSIs:

- Allow destination override should be set.
- For the PF VSIs, *Queue Mapping* should be set to contiguous and all the queues of the PF should be allocated to this VSI. The first queue is always queue 0 (of the PF).

As this flow is applied on each core reset, it is possible that a valid alternate RAM already exists. In this case, the alternate RAM based configuration, as described in Section 7.4.9.4.3, is applied.

7.4.9.4.2.2 PF Allocations

For each PF, the *PF allocation* sub module in the *EMP Core Module* contains guaranteed allocations for the following resources:

- VEBs
- VSIs
- Perfect match MAC addresses
- S-tags
- VLAN Statistic pools
- Mirror rules

After the basic topology is defined, an L2 filter should be added for each PF using the factory MAC address of this PF as stored in the PF allocations structure in the NVM or any alternate RAM override of those addresses. The *Load PF MAC Address* field in the *PF flags* word in the *PF allocation* sub module is used to define which MAC addresses to add to the switching decision. If this bit is set, only packets that passes the MAC and if needed, the S-tag are forwarded to the PF.

In addition an L2 filter forwarding packets to the EMP should be added for each port using the port MAC addresses stored in the *PRTGL_SAL* and *PRTGL_SAH* registers

7.4.9.4.3 BIOS Configuration

As part of the SMASH CLP stage, commands affecting the switch configuration may be received. These commands should be translated by the SMASH CLP code to the updates of the Alternate RAM.

After all the SMASH CLP commands are received, the CLP code should update the firmware with the content of the alternate RAM. This flow is described in Section 4.2.2.2.

For each of the enabled functions, a VSI is added and is connected to the MAC .

The VSIs created at this stage have the default switch ID of the port. The following parameters should be set to a non default value in these VSIs:

- Allow destination override should be set.
- *Queue Mapping* should be set to contiguous and all the queues of the PF should be allocated to this VSI. The first queue is always queue 0 (of the PF).
- FCoE Enable should be set only if *PFGEN_STATE[PF#].PFFCEN* is set

Once the alternate RAM is updated and executed by the firmware, it may, depending on the configuration, disable some of the admin commands the software is allowed to execute.

7.4.9.4.4 Operating System initialization

After the OS boot, each Physical function is connected to a physical port or an S-channel. Before adding further switching elements, the Physical function driver should use the *Get Switch Configuration* admin command (Section 7.4.9.5.3.1) to get the SEID of the switch element to which this function is connected.

After that, it may add VEB, VEPA or Port Virtualizer elements according to the instructions received from the switch control plane in the operating system using the admin commands described in Section 7.4.9.5.

Note: In operating systems where two separate VSIs are used for LAN and FCoE, the driver should create a VEB using the *Add VEB* command and connect a separate VSI for the LAN queues and the FCoE traffic. If a VEB is created for virtualization purposes, the same VEB may be used to connect the FCoE and LAN traffic VSIs.

7.4.9.5 Programming Interface

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.

7.4.9.5.1 Switch Configuration Admin Commands Summary

Table 7-68 summarize the different commands used to configure switch elements.

Table 7-68. Switch configuration admin commands (0x02xx)

Command	Opcode	Brief description	Detailed description
Generic Commands (0x020x)			
Get Switch Configuration	0x0200	Describe the networking structure of the port	7.4.9.5.3.1
Add Statistics	0x0201	Add a statistics block to a VLAN in a switch.	7.4.9.5.3.2
Remove Statistics	0x0202	Remove a statistics block for a VLAN in a switch.	7.4.9.5.3.3
Set Port Parameters	0x0203	Defines the default parameters of a LAN port	7.4.9.5.3.4
Get Resources Allocation	0x0204	Reports the resources allocated to the PF.	7.4.9.5.3.5
VSI Commands (0x021x)			

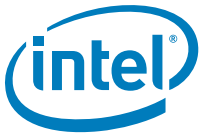


Table 7-68. Switch configuration admin commands (0x02xx)

Command	Opcode	Brief description	Detailed description
Add VSI	0x0210	Add a VSI to a switching element	7.4.9.5.4.1
Update VSI parameters	0x0211	Update parameters of a VSI	7.4.9.5.4.2
Get VSI Parameters	0x0212	Get the parameters of a VSI.	7.4.9.5.4.3
Port Virtualizer control (0x022x)			
Add Port Virtualizer	0x0220	Create an S-comp or a port extender	7.4.9.5.5.1
Update Port Virtualizer Parameters	0x0221		7.4.9.5.5.2
Get Port Virtualizer Parameters	0x0222		7.4.9.5.5.3
VEB/Port aggregator Control (0x023x)			
Add VEB	0x0230	Create a VEB/Port aggregator	7.4.9.5.6.1
Get VEB Parameters	0x0232	Get the parameters of a VEB/VEPA	7.4.9.5.6.2
Switch Connectivity Configuration (0x024x)			
Delete Element	0x0243		7.4.9.5.7.1
Forwarding Table Configuration (0x025x)			
Add MAC,VLAN pair	0x0250	Add a MAC/VLAN pair to the lookup table	
Remove MAC,VLAN pair	0x0251	Remove a MAC/VLAN pair from the lookup table	7.4.9.5.8.2
Add VLAN	0x0252	Add a VLAN to the VEB/Port aggregator (can be regular or local, primary or secondary,)	7.4.9.5.8.3
Remove VLAN	0x0253	Remove a VLAN or members of a VLAN from the VEB/ Port aggregator	7.4.9.5.8.4
Set VSI Promiscuous Modes	0x0254		7.4.9.5.8.5
Add S-tag	0x0255	Add an of S-tags to a VSI	7.4.9.5.8.6
Remove S-tag	0x0256	Remove an S-tag from a VSI	7.4.9.5.8.7
Add E-tag	0x0257	Add an Association from an E-tag to a set of S-tags.	
Update S-tag	0x0259	Update the default S-tag of a VSI.	7.4.9.5.8.8
Add Control Packet Filter	0x025A	Add Ethertype (+MAC) filter to control port.	7.4.9.5.8.9
Remove Control Packet Filter	0x025B	Remove Ethertype (+MAC) filter from control port.	7.4.9.5.8.10
Add Cloud Filters	0x025C	Add a set of filters for cloud connections	7.4.9.5.8.11
Remove Cloud Filters	0x025D	Remove a set of filters for cloud connections	7.4.9.5.8.12
Mirroring Configuration (0x026x)			
Add Mirror rule	0x0260	Define a mirroring rule	7.4.9.5.9.1
Delete Mirror rule	0x0261	Remove a mirroring rule	7.4.9.5.9.2
Storm Control Commands (0x028x)			
Set Storm Control Configuration	0x0280	Define the parameters for the Storm Control engine	7.4.9.5.10.1
Get Storm Control Configuration	0x0281	Read the current Storm control configuration	7.4.9.5.10.2

7.4.9.5.2 Configuration Flow Examples

This section describes examples of Software configuration flows for different types of switch topologies.

The following examples are provided:

- An SFP with a Port Virtualizer used to distribute the virtual machine traffic. In this mode, each VF/VM is assigned a different S-channel.
- An SFP with a VEB used to distribute the virtual machine traffic. In this mode, each VF/VM is assigned a VSI within the VEB. The same flow can be used to add a VEPA element.
- An MFP with a Port Virtualizer used to distribute the physical ports traffic. In this mode, each PF is assigned a different S-channel.

More complex network topologies are supported. The configuration of such topologies can be inferred from the examples below.

For each example, the admin command to use and the main parameters in each command are described.

7.4.9.5.2.1 Port Virtualizer (SFP)

In this mode, the internal firmware initiate a single VSI per port. This VSI is connected to the PF. Other VSIs may be connected to the EMP for BMC or local traffic.

The software driver should use the following command to create a Port Virtualizer and connect a set of VSIs to it:

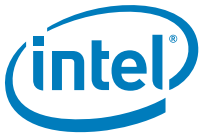
- *Get Switch Configuration* - this command will provide the Default VSI connected to the port.
- *Get VSI* - this command provides the content of the current VSI context. The PF may then update this context using an *Update VSI* command.
- *Add* Port Virtualizer - this command will add a Port Virtualizer on top of the port. The *uplink SEID* should be the Physical port. The *Default VSI SEID* should be the Default VSI received in the previous command.
- If needed, a separate Control Port may be added by using an *Add VSI* with the Port Virtualizer as the *uplink SEID* and a Control Port *Connection type*.
- For each of the VMs/VFs that needs to be connected to the Port Virtualizer, an *Add VSI* should be sent with the Port Virtualizer as the uplink SEID, a Regular Data Port *Connection type* and a switch ID value equal to the S-tag assigned to this VM/VF. For VMs, the VSI type should be *VM*.

7.4.9.5.2.2 VEB (SFP)

In this mode, the internal firmware initiate a single VSI per port. This VSI is connected to the PF. Other VSIs may be connected to the EMP for BMC or local traffic.

The software driver should use the following command to create a VEB and connect a set of VSIs to it:

- *Get Switch Configuration* - this command will provide the Default VSI connected to the port.
- *Get VSI* - this command provides the content of the current VSI context. The PF may then update this context using an *Update VSI* command.
- *Add VEB* - this command will add a Port Virtualizer on top of the port. The *uplink SEID* should be the Physical port. The *Default VSI SEID* should be the Default VSI received in the previous command. The driver should register the switch ID assigned to this VEB.
- If needed, a separate Control Port may be added by using an *Add VSI* with the VEB as the *uplink SEID* and a Control Port *Connection type*.



- For each of the VMs/VFs that needs to be connected to the VEB, an *Add VSI* should be sent with the Port Virtualizer as the uplink SEID, a Regular Data Port *Connection type* and a switch ID value equal to the switch ID of the VEB. For VMs, the VSI type should be *VM.*
- For each VM/VF, *Add MAC, VLAN pair* commands should be sent to allow adequate forwarding of the traffic.

7.4.9.5.2.3 Port Virtualizer (MFP)

In this mode, the internal firmware initiate a single Port Virtualizer and a VSI for each PF on the port. Other VSIs may be connected to the EMP for BMC or local traffic.

The software driver should use the following command to create a VEB and connect a set of VSIs to it:

- *Get Switch Configuration* - this command will provide the Default VSI connected to the port.
- *Get VSI* - this command provides the content of the current VSI context. Each PF may then update this context using an *Update VSI* command.

7.4.9.5.3 Generic Commands (Opcode 0x020x)

7.4.9.5.3.1 Get Switch Configuration (0x0200)

This command is used to discover the switch configuration of the port. The driver must use this command as part of the init flow before adding new switching elements or manipulating existing ones.

This function is available to any PF. In an MFP case, each physical function will see only the switch configuration owned by it. This usually includes the switching structure from the Port Virtualizer and upwards.

Table 7-69. Get Switch Configuration command (Opcode: 0x0200)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0200	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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
First SEID	16-17	0x0	If not zero, start the report from the requested SEID - used for continuation commands
Reserved	18-23	0x0	Must be zero
Data Address high	24-27		Address of response buffer.
Data Address low	28-31		

Table 7-70 describes the response buffer for the Get Switch Configuration command.

Table 7-70. Get Switch Configuration response (Opcode: 0x0200)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0200	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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Next SEID	16-17	0x0	If not zero, indicates that not all the configuration was returned and a new command should be sent with this value in the First SEID field
Reserved	18-23	0x0	Must be zero
Data Address high	24-27		Address of response buffer.
Data Address low	28-31		

Table 7-71 describes the structure of the response buffer for the Get Switch Configuration command.

Table 7-71. Get Switch Configuration command response buffer

Offset (bytes)	Description
0-15	Header - see Table 7-72 for details
16-31	Switch element #1- see Table 7-73 for details.
....	
16*n+15-16*n	Switch element #n - see Table 7-73 for details.

Table 7-72. Get Switch Configuration command response header

Offset (bytes)	Description
0-1	Number of switching elements in the structure. The size of the buffer is 16 * (Number of Elements + 1) If the buffer size is too small to return all the switching elements, only the elements fitting the buffer will be returned and the driver may request the subsequent elements using a different First element. The maximal number of entries that can be returned in a single command is 255.
2-3	Total Number of switching elements. The total number of switching elements. This may be larger than the number of elements in the structure.
4-15	Reserved

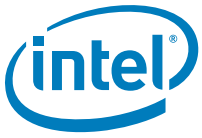


Table 7-73. Get Switch Configuration - Switch element

Offset (bytes)	Description	Notes
0	Element Type - as described in Table 7-60 and Table 7-61.	
1	Revision - describes the revision of the element type - For the XL710, the revision is always 1.	
2-3	SEID - defines the Switch Element ID of this element.	
4-5	Up Link Connection: Indicates the SEID of the switching element connected directly below the current element. Below means towards the network in this case.	For sub elements of a composed switch element, the uplink is the uplink of the entire switch element.
6-7	Downlink Connection - Indicates the SEID of the switching element connected directly above the current element. Above means towards the host in this case.	The Down link of a composed element is the default port of this element
8-10	Reserved	Reserved
11	Connection type: 0x0: Reserved 0x1: Regular Data Port 0x2: Default Port 0x3: Cascaded Port Virtualizer port 0x4 - 0xFF: Reserved.	Defines the type of connection to the uplink element
12-13	Reserved	
14-15	Element Specific information. Provides additional information according to the element type as described bellow: **Element type** **Information** MAC (1) The Physical port number PF (2) The Physical function number VF (3) The Virtual function number VSI (19) The VSI number This field is reserved for all other element types.	

7.4.9.5.3.2 Add Statistics (0x0201)

XL710 supports 128 smonVlanStats counters as described in Section 7.11.5.2.This command is used to allocate a set of smonVlanStats counters to a specific VLAN in a specific switch.

Table 7-74. Add Statistics command (Opcode: 0x0201)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0201	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Switch SEID	16-17		Defines the SEID of the switch for which the stats are requested.

Table 7-74. Add Statistics command (Opcode: 0x0201)

Name	Bytes.Bits	Value	Remarks
VLAN ID	18-19		The VLAN ID for which the statistics are requested: 15:12: Reserved 11:0: VLAN ID
Statistic counter	20-21	Statistic counters index	Zeroed by driver, FW Returns an index of the statistics counters block assigned to this VLAN.
Reserved	22-31	0x0	Reserved

Table 7-75. Add Statistics Response (Opcode: 0x0201)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0201	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - if the SEID do not point to a valid switch. ENOSPC - if there aren't enough resources to assign a statistics block.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Switch SEID	16-17		Copied from command.
VLAN ID	18-19		Copied from command.
Statistic counter	20-21	Statistic counters index	Zeroed by driver, FW Returns an index of the statistics counters block assigned to this VLAN.
Reserved	22-31	0x0	Reserved

7.4.9.5.3.3 Remove Statistics (0x0202)

XL710 supports 128 smonVlanStats counters as described in Section 7.11.5.2.This command is used to deallocate a set of smonVlanStats counters from a specific VLAN in a specific switch.

Table 7-76. Remove Statistics command (Opcode: 0x0202)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0202	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Switch SEID	16-17		Defines the SEID of the switch for which the stats are currently allocated.

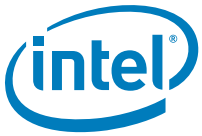


Table 7-76. Remove Statistics command (Opcode: 0x0202)

Name	Bytes.Bits	Value	Remarks
VLAN ID	18-19		The VLAN ID for which the statistics are currently allocated: 15:12: Reserved 11:0: VLAN ID
Statistic counter	20-21	Statistic counters index	The statistics counters block assigned to this VLAN.
Reserved	22-31	0x0	Reserved

Table 7-77. Remove Statistics Response (Opcode: 0x0202)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0202	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - if the SEID do not point to a valid switch. EINVAL - if the statistic block mentioned is not pointed by this VLAN, switch combination..
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31		Reserved

7.4.9.5.3.4 Set Port Parameters (0x0203)

This command is used to define the default parameters of a physical port. When received in MFP mode, the last configuration received is used. In MFP mode, the configuration is not reset by a PFR, only by a device wide reset. The only exception is pass bad frames which is reset, if the PF to which the bad frames VSI is associated is reset.

Table 7-78. Set Port Parameters command (Opcode: 0x0203)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0203	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command

Table 7-78. Set Port Parameters command (Opcode: 0x0203)

Name	Bytes.Bits	Value	Remarks
Command Flags	16-17	Bitfield	0: Save Bad Packets - if set, packets with errors are forwarded to the bad frames VSI. 1: Reserved - set to 1. 2: Enable Double VLAN: If set, this port expects double VLAN packets. The rest of the bits are Reserved.
Bad Frames SEID	18-19		Defines the SEID of the VSI to which bad frames are forwarded. Bit 15: Valid SEID - ignored in this field Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI. ***Note:*** Relevant only if Command Flags.Save Bad Packets is set.
Reserved	20-31	0x0	Reserved

Table 7-79. Set Port Parameters response (Opcode: 0x0203)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0203	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware. The following response may be returned by this command: EPERM - if the operation is not permitted.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW 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

7.4.9.5.3.5 Get Switch Resources Allocation (0x0204)

This command is used to query the resources allocated to a function.

Table 7-80. Get Switch Resources Allocation command (Opcode: 0x0204)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0204	Command opcode
Datalen	4-5		Length of response buffer - should be big enough to contain the expected response buffer size.
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command

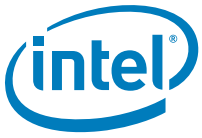


Table 7-80. Get Switch Resources Allocation command (Opcode: 0x0204)

Name	Bytes.Bits	Value	Remarks
Reserved	16-23	0x0	Reserved
Data Address high	24-27		Return Buffer Address
Data Address low	28-31		

Table 7-81. Get Switch Resources Response (Opcode: 0x0204)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0204	Command opcode
Datalen	4-5		Length of buffer
Return value/VFID	6-7		Return value. There are no specific errors to this command
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Number of entries	16	0xD/0x14[1]	Number of resource entries
Reserved	17-23	0x0	Reserved
Data Address high	24-27		Return Buffer Address
Data Address low	28-31		

1. In cloud images (*Features Enable.Switching mode* is set to "Cloud" or "UDP cloud", then the relevant filters are also returned.

The return buffer contains structures of 16 bytes for each resource type of the following format. The number of entries is returned in the Number of entries field in the response.

Name	Bytes.Bits	Description
Resource Type	0	0x0 = VEBs 0x1 = VSIs 0x2 = Perfect Match MAC addresses 0x3 = S-tags 0x4 = E-tags 0x5 = Multicast hash entries 0x6 = unicast hash entries 0x7 = VLANs 0x8= VSI List entries (3 VSI in each entry) 0x9 = E-tag List entries (3 S-tags in each entry) 0xA = VLAN Statistic pools 0xB = Mirror rules 0xC = Queue sets. 0xD = Inner VLAN Forward filters 0xE = Reserved 0xF = Inner MACs 0x10 = IPs 0x11 = GRE/VN1 Keys. 0x12 = VN2 Keys 0x13 = Teredo Ports 14- 0xFF = Reserved.
Reserved	1	Reserved
Guaranteed Allocation	2-3	Number of resources from this type guaranteed to this function
Total	4-5	Total number of resources from this type available to all functions
Currently used	6-7	Number of resources from this type used by this function
Total un-allocated	8-9	Total number of resources from this type still un-allocated and not reserved by any function.
Reserved	10-15	Reserved

7.4.9.5.4 VSI Commands (Opcode 0x021x)

7.4.9.5.4.1 Add VSI (0x0210)

This command is used to add a new VSI. A VSI must connect to an existing switching element. A function can connect a VSI only to a switching element it controls.

When an Add VSI command is received, the internal Firmware checks if all the requested resources are available and allocate them to the VSI. If part of the resources are not available, the command returns a list of unavailable resources

Table 7-82. Add VSI command (Opcode: 0x0210)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 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 driver. Written by Firmware.

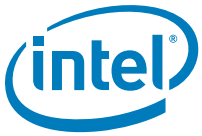


Table 7-82. Add VSI command (Opcode: 0x0210)

Name	Bytes.Bits	Value	Remarks
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
uplink SEID	16-17	0x0	Defines the uplink SEID to which this VSI should be connected.
Connection Type	18	0x0	Defines to which port of the uplink element this VSI connects: 0x0: Reserved 0x1: Regular Data Port 0x2: Default Port 0x3 - 0xFF: 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 (0-127) and not the number relative to the PF first VF.
Reserved	21		Reserved.
Command Flags	22-23		1:0: VSI type: • 0=VF, • 1=VMDq2 (a.k.a. VM), • 2=PF, • 3=EMP/MNG 2: Cascaded Port Virtualizer. The S-tag manipulation commands can be used only if this bit is set. Bit 15:3: Reserved.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

The following table describes the structure of the command buffer for the Add VSI command.

Table 7-83. Add VSI Command Buffer

Category	Byte/ Bit	Field	Description
Valid sections	0-1	Defines which sections are valid in the command	0: Switching section is valid. Must be set for *Add VSI* commands when containing SEID is a Port Virtualizer. 1: Security section is valid 2: VLAN handling section is valid 3: Cascaded Port Virtualizer section is valid. This bit should not be set in MFP mode. 4: Ingress UP translation section is valid 5: Egress UP translation section is valid 6: Queue Mapping section is valid. Must be set for *Add VSI* commands. If set for *Update VSI* command, modified queues must be disabled. 7: Queuing option section is valid 8: Outer UP section is valid. Should not be set if inner UP to outer UP mapping is the identity mapping. 9: Scheduler section is valid. This bit should not be set in *Update VSI* command. 10-15: Reserved
Switching	2-3.3	Switch ID	Defines the switch ID to which this VSI belongs. If is not S-tag is cleared, then this switch ID is an S-tag, otherwise it is a virtual switch ID If the VSI is connected to a Port Virtualizer, this value must be set. For a port connected to a VEB or directly to the MAC, this value and the Is not S-tag field may be left at zero and the VEB switch ID or the default switch ID will be used. For Update VSI command should be a valid tag (could be zero for VEB)
	3.4	Is not S-tag	See description of the Switch ID parameter for the usage of this bit.
	3.5	Allow Loopback	This bit should be set for VSIs that are connected to a VEB and should be cleared otherwise. Cleared if section not valid.
	3.6	Allow Local Loopback	This bit should be set for VSIs that are used as uplink of a software (cascaded) VEB, VEPA or Port Virtualizer. Cleared if section not valid.
	3.7-5.7	Reserved	Reserved for future switching parameters. Must be zero
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 control ports Cleared if section not valid.
	6.1	Enable VLAN anti spoof	Cleared if section not valid. ***Note:*** Enabling this mode may impact the transmit performance. It is recommended to use only the MAC anti spoof mode.
	6.2	Enable MAC anti spoof	Cleared if section not valid.
	6.3 - 7.7	Reserved	Reserved for future security parameters. Must be zero

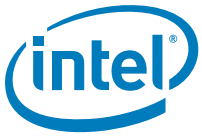


Table 7-83. Add VSI Command Buffer

Category	Byte/ Bit	Field	Description
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. Cleared if section not valid.
	10-11	FCoE PVID + Default UP (16 bits)	Defines the PVID to use for FCoE packets. This field is relevant only if the *Insert PVID* field is set. Cleared if section not valid.
	12.0:1	VLAN driver insertion mode	This field defines if the driver is allowed/should add a VLAN tag to the packets it sends. If *Insert PVID* field is set, this field should be set to 01b. 00b: Reserved 10b: Admit.1Q tagged only - Allow only packets with VLAN 01b: Admit untagged/Priority tagged only - allow only packets without VLAN or with VLAN tag = 0. 11b: Allow all packets If section not valid, the default value is 11b.
	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. If FCoE queues are expected, the *FCoE PVID + Default UP* field should be set also. Cleared if section not valid.
	12.3:4	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 and UP in descriptor (legacy behavior) 01b: Hide VLAN show UP (VLAN ID = 0) 10b: Hide VLAN and UP 11b: Do nothing (leave VLAN in packet) If section not valid, the default value is 11b.
	12.5-15.7	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: 16.0:16.2 UP set if received UP is 0. 16.3:16.5 UP set if received UP is 1. 16.6:17.0 UP set if received UP is 2. 17.1:17.3 UP set if received UP is 3. 17.4:17.6 UP set if received UP is 4. 17.7:18.1 UP set if received UP is 5. 18.2:18:4 UP set if received UP is 6. 18.5:18.7 UP set if received UP is 7. 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). If section is not valid, the default value is identity mapping.

Table 7-83. Add VSI Command Buffer

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: 20.0:20.2 UP set if sent UP is 0. 20.3:20.5 UP set if sent UP is 1. 20.6:21.0 UP set if sent UP is 2. 21.1:21.3 UP set if sent UP is 3. 21.4:21.6 UP set if sent UP is 4. 21.7:22.1 UP set if sent UP is 5. 22.2:22.4 UP set if sent UP is 6. 22.5:22.7 UP set if sent UP is 7. 23: Reserved This map is used to translate the 802.1P user priority bits sent by the host to the i user priority sent to the network. Note that the resulting user priority is further translated using the per TC translation table. If section is not valid, the default value is identity mapping.
Cascaded Port Virtualizer	24-25	S-tag	The S-tag/E-tag (S-VID/E-VID) to be inserted in transmit packets. If *Switch ID* parameter is set and is an S-tag and not a cascaded Port Virtualizer, should be set to the same value as the *Switch ID* parameter. This field is relevant only if *S-tag insert enable* field is set. This field is 16 bit and should include also the default PCP priority bits to insert in the S-tag Cleared in SFP mode and equal to the S-channel S-tag in MFP mode if section not valid.
	26.0:1	S/E-tag extract mode.	This field defines how received S-tags/E-tags are handled. 00b: Do Nothing (cascaded Port Virtualizer without offload) 01b: Extract tag and do not insert in descriptor (regular VM) 10b: Extract tag from packet and expose in descriptor (cascaded Port Virtualizer with offload). 11b: Reserved If section not valid, the default value is 00b in SFP mode and 01b in MFP mode.
	26.2:3	Reserved	Reserved
	26.4	S-tag insert enable.	This bit controls the port based insertion of S-tags/E-tags. Should be set, if VSI is part of an S-channel and is not a cascaded Port Virtualizer VSI. If section not valid, the default value is cleared in SFP mode and set in MFP mode if section not valid. When this bit is set, the *Accept tag from host* bit should be cleared
	26.5	Reserved.	Reserved.
	26.6	Accept tag from host.	Allow host to insert S-tag/E-tag in descriptor or in packet. Should be set only for cascaded port virtualizer or control ports. Set in SFP mode and cleared in MFP mode if section not valid. When this bit is set, the *S-tag insert enable* bit should be cleared. If set, firmware should also set the VSI_L2TAGSTXVALID.L2TAG2INSERTID to 1b (S-tag) and VSI_L2TAGSTXVALID.L2TAG2INSERTID_VALID to 1b.
	26.7-27.7	Reserved	Reserved for future port S-tags parameters. Must be zero

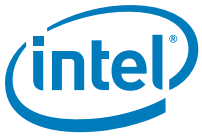


Table 7-83. Add VSI Command Buffer

Category	Byte/ Bit	Field	Description
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 queues. 1b: The VSI is assigned a scattered range of PF queues.
	28.1 - 29	Reserved	Reserved - must be zero.
	30-61	Queue mapping	If mapping method = 0 (contiguous): 30.0:31.2: The first queue allocated for this VSI in the PF space. This VSI can access all the queues from this queue to the end of the PF queues allocation, but will be bounded by the receive queues per TC configuration. 31.3:61: 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: • 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. • 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, 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 will be sent.
	62-77	Number and offset of queue pairs per TCs	Fixes the number of queue pairs assigned to the VSI for each traffic class and the offset of these queues. 62.0 - 63.0: Queue offset for TC0. 63.1 - 63.3: Number of queues allocated to TC0. The actual number is 2^n. The allowed number of queues are: 1; 2; 4; 8; 16; 32; 64. 63.4 - 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 will be sent to the default queue. The following addresses are used with the same format for the next TCs: 64:65: TC1 66:67: TC2 68:69: TC3 70:71: TC4 72:73: TC5 74:75: TC6 76:77: TC7
Queueing options	78.0:3	Reserved	Reserved
	78.4	TCP packets Enable	Reserved Cleared.
	78.5.	FCoe Enable	Enable FCoE filter context for the VSI. When this flag is cleared, the hardware does not look for matched FCoE filter. Cleared if section not valid.
	78.6 - 81	Reserved	Reserved for future queueing parameters. Must be zero

Table 7-83. Add VSI Command Buffer

Category	Byte/ Bit	Field	Description
Scheduler	82	Enabled TCs	A bitmap indicating which TCs are enabled in this VSI. If section not valid, only TC0 is enabled.
	83	Reserved	Reserved
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: 84.0:84.2 Outer UP set if inner UP is 0. 84.3:84.5 Outer UP set if inner UP is 1. 84.6:85.0 Outer UP set if inner UP is 2. 85.1:85.3 Outer UP set if inner UP is 3. 85.4:85.6 Outer UP set if inner UP is 4. 85.7:86.1 Outer UP set if inner UP is 5. 86.2:86.4 Outer UP set if inner UP is 6. 86.5:86.7 Outer UP set if inner UP is 7. 87: Reserved This map is used to translate the 802.1P user priority bits on the inner UP to outer UP If section is not valid, the default value is identity mapping. In order to get a fixed outer UP, all the values should be set to the requested Outer UP
Reserved	88-95	Reserved	Reserved
Response Space	96-127	Reserved	Reserved for Response space

Table 7-85 describes the Add VSI response and the Response buffer.

Table 7-84. Add VSI Response Buffer

Category	Byte/ Bit	Field	Description
Command Space	0-95	Reserved	Reserved for Command Space - should contain the values provided by the driver.

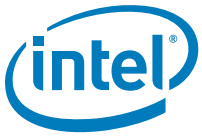


Table 7-84. Add VSI Response Buffer

Category	Byte/ Bit	Field	Description
Queue set Handles	96-97	QS_Handle 0	The handle for Queue set of TC0. Bits [9:0] of this handle are used by software to program the RDYList field in the transmit queues context for queues associated with TC0.
	98-99	QS_Handle 1	The handle for Queue set of TC1. Same format as QS_Handle 0
	100-101	QS_Handle 2	The handle for Queue set of TC2 Same format as QS_Handle 0
	102-103	QS_Handle 3	The handle for Queue set of TC3 Same format as QS_Handle 0
	104-105	QS_Handle 4	The handle for Queue set of TC4 Same format as QS_Handle 0
	106-107	QS_Handle 5	The handle for Queue set of TC5 Same format as QS_Handle 0
	108-109	QS_Handle 6	The handle for Queue set of TC6 Same format as QS_Handle 0
	110-111	QS_Handle 7	The handle for Queue set of TC7 Same format as QS_Handle 0
Statistic counter	112-113	Statistic counters index	Returns an index of the statistics counters block assigned to this VSI.
Reserved	116-127	Reserved	Reserved

Table 7-85. Add VSI Response (Opcode: 0x0210)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 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: ENOENT - if the uplink SEID or the Function Number do not point to valid elements. EINVAL - if the topology created is not valid. EEXIST - if a default port is requested and it already exists. ENOSPC - if there aren't enough resources to assign a VSI.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID	16-17		If the Return value is zero, Returns the SEID of the VSI. If Return value is ENOSPC, contains a bitmap that indicates which resources are missing: 0: No VSI left 1: Not enough Scheduler nodes. 2: Not enough statistics counters. 3: Not enough switching entries. 4-15: Reserved.

Table 7-85. Add VSI Response (Opcode: 0x0210)

Name	Bytes.Bits	Value	Remarks
VSI Number	18-19		Returns the assigned VSI number
VSIs used	20-21		Number of VSIs used by this function
Total VSIs un-allocated	22-23		Total number of VSIs still un-allocated and not reserved by any function.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

7.4.9.5.4.1.1 Add VSI Settings Recommendations

Table 7-86 describes the recommended settings for common types of VSIs:

Table 7-86. Add VSI Recommended settings

Category	Field	VSI type					
		PF	VF	VMDq2	VMDq1	Cascaded VEB	Cascaded S-comp
Valid sections	Switching section is valid	0	1	1	1	1	1
	Security section is valid	1					
	VLAN handling section is valid	1					
	Cascaded Port Virtualizer section is valid	1					
	Ingress UP translation section is valid	0					
	Egress UP translation section is valid	0					
	Queue Mapping section is valid	1					
	Queuing option section is valid	1					
	Scheduler section is valid	0 if non CDB					
Switching	Switch ID	If the VSI is connected directly to a Port Virtualizer, this value must be set. For a port connected to a VEB or directly to the MAC, this value may be left at zero and the VEB switch ID or the default switch ID will be used. When connecting to a Port Virtualizer, the Switch ID should be set to the S-tag of the channel and the Is not S-tag should be cleared.					Set to the first tag covered by this S-comp
	Is not S-tag						0
	Allow Loopback	0	1 if VEB 0 if VEPA		1 if part of a VEB, 0 otherwise	1 if VEB 0 if VEPA	0
	Allow Local Loopback	0	0	0	1 if part of a VEB, 0 otherwise	1 if VEB 0 if VEPA	0
Security	Allow destination override	1	0	1	1	1	1
	Enable VLAN anti spoof	0	1	0	0	0	0
	Enable MAC anti spoof	0	1	0	0	0	0

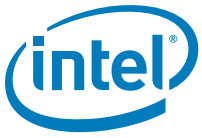


Table 7-86. Add VSI Recommended settings

Category	Field	VSI type					
		PF	VF	VMDq2	VMDq1	Cascaded VEB	Cascaded S-comp
VLAN handling	PVID+ Default UP (16 bits)	N/A	Port based VLAN	N/A	N/A	N/A	N/A
	FCoE PVID + Default UP (16 bits)	N/A	FCoE Port based VLAN	N/A	N/A	N/A	N/A
	VLAN driver insertion mode	11b	Depends on VLAN Awareness	11b	11b	11b	11b
	Insert PVID	0		0	0	0	0
	VLAN and UP expose mode (Rx)	00b (extract and show)		00b (extract and show)	00b (extract and show)	00b (extract and show)	00b (extract and show)
Ingress UP translation	Ingress UP translation table	Identity mapping (default)					
Egress UP translation	Egress UP translation table	Identity mapping (default)					
Cascaded Port Virtualizer	S-tag	Same as Switch ID if Port Virtualizer is part of the switch, N/A otherwise					N/A
	S-tag extract mode.	01b (Extract S-tag and do not insert in descriptor)					10b (Extract S-tag from packet and expose in descriptor)
	E-tag Extract mode.						
	S-tag insert enable.	1 if Port Virtualizer is part of the switch topology, 0 otherwise					0
	Prune based on internal S-tag enable.	1					0
	Accept tag from host.	0					1
Queue mapping	Mapping method	Depends on Queue mapping method					
	Queue mapping						
	Number and offset of receive queues per TCs	According to allocation to TCs.					
Queueing options	TCP packets Enable	1					
	FCoe Enable	Set for VSIs supporting FCoE					
Scheduler	Enabled TCs	According to allocated TCs (0x1 for non DCB environment)					

7.4.9.5.4.2 Update VSI (0x0211)

This command is used to update the parameters of an existing VSI. A function can update only a VSI it controls. Not all the parameters of a VSI can be updated. Only parameters that do not impact the scheduler tree structure can be modified. In order to change the traffic classes allocated to a VSI, the *Configure VSI Bandwidth per Traffic Class* command described in Section 7.8.4.7.should be used

When updating a VSI connected to a Port Virtualizer in MFP mode, the following restrictions applies:

- The *Switch ID* field should be set to zero.
- The *Cascaded Port Virtualizer section is valid* field should be cleared.

Table 7-87. Update VSI command (Opcode: 0x0211)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0211	Command opcode
Datalen	4-5	0x80	Length of buffer
Return value	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID Number	16-17		VSI SEID number
Reserved	18-21		Reserved
Command Flags	22-23		2:0: Reserved 3: Cloud VSI (A0 only). This bit can only be set. If cleared, it is ignored. Thus a cloud VSI cannot be reverted to a non-cloud VSI. 15:4: Reserved. 15:0: Reserved.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

The command buffer and the completion buffer used to update a VSI is the same as the command and completion buffers of the Add VSI command (Table 7-83 and Table 7-84). 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 type.

Table 7-88. Update VSI Response (Opcode: 0x0211)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0211	Command opcode
Datalen	4-5	0x80	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by 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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID Number	16-17	0x0	Copied from command
VSI Number	18-19		Returns the assigned VSI number
VSI Used	20-21		Number of VSIs used by this function

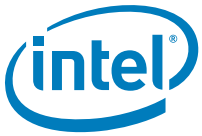


Table 7-88. Update VSI Response (Opcode: 0x0211)

Name	Bytes.Bits	Value	Remarks
Total VSIs un-allocated	22-23		Total number of VSIs still un-allocated and not reserved by any function.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

7.4.9.5.4.3 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 7-89. Get VSI Parameters Response (Opcode: 0x0212)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0212	Command opcode
Datalen	4-5	0x80	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by 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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID Number	16-17		VSI SEID number (copied from command)
VSI Number	18-19		Returns the assigned VSI number
VSIs used	20-21		Number of VSIs used by this function
Total VSIs un-allocated	22-23		Total number of VSIs still un-allocated and not reserved by any function.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

Table 7-90. Get VSI Parameters Command (Opcode: 0x0212)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0212	Command opcode
Datalen	4-5	0x80	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command

Table 7-90. Get VSI Parameters Command (Opcode: 0x0212)

Name	Bytes.Bits	Value	Remarks
SEID Number	16-17		VSI SEID number
Reserved	18-23		Reserved
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

The following table describes the response buffer received when querying a VSI.

Table 7-91. Get VSI Parameters Response Buffer

Byte/Bit	Description
0-95	Same parameters as described in Table 7-83.
96-127	Same parameters as described in Table 7-84.

7.4.9.5.5 Port Virtualizer Commands (Opcode 0x022x)

7.4.9.5.5.1 Add Port Virtualizer

This command is used to instantiate an Port Virtualizer on a port. A Port Virtualizer can be instantiated only when a single VSI is connected between the MAC and a PF. If additional switching elements are active on this port, they must be first disconnected before the Port Virtualizer can be added.

When an Add Port Virtualizer command is received, the internal Firmware checks if all the requested resources are available and allocate them to the Port Virtualizer. If part of the resources are not available, the command returns a list of unavailable resources

Table 7-92. Add Port Virtualizer command (Opcode: 0x0220)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0220	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Command Flags	16-17	0x0	0: Port Virtualizer type: Must be set to zero 1: Forward Unknown S-tag. If set, packets with unknown S-tags are forwarded to the Default VSI. *2:* Reserved ***Note:*** If bit 1 is set, the port type should be Default, otherwise these bits are ignored. 3: Port type: • 0 - Default • 1 - Control 4:15: Reserved

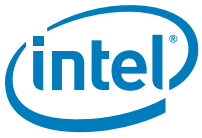


Table 7-92. Add Port Virtualizer command (Opcode: 0x0220)

Name	Bytes.Bits	Value	Remarks
uplink SEID	18-19	0x0	Defines the SEID to which this Port Virtualizer should be connected. This can be only the MAC of the port on which this function resides.
Connected VSI SEID	20-21	0x	Defines the SEID of the first VSI connected to this Port Virtualizer. Port type is defined by the "Port Type" flag. Should point to a valid VSI, connected to the uplink SEID as Uplink.
Reserved	22-31		Reserved

Table 7-93. Add Port Virtualizer Response (Opcode: 0x0220)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0220	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - if the port SEID do not point to a valid element. ENOSPC - if there aren't enough resources to assign a Port Virtualizer. ESRCH - if the operation is not permitted (e.g. MFP mode or virtualization is disabled) EACCES - if the port is not owned by this PF. EINVAL - if a driver tries to create a Port Virtualizer of a type contradicting the previously created Port Virtualizer type.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Port Virtualizer SEID/Error Reasons.	16-17		If the Return value is zero: Bytes 25:24:Returns the SEID of the Port Virtualizer If Return value is ENOSPC, contains a bitmap that indicates which resources are missing: 0: No Port Virtualizer left 1: Not enough Scheduler nodes. 2: Reserved 3: Not enough switching entries. 4-31: Reserved.
Reserved	18-31		Reserved

7.4.9.5.5.2 Update Port Virtualizer Parameters

This command is used to modify the parameters of an Port Virtualizer. In this command, the flags of the Port Virtualizer can be modified. Note that the default VSI and control VSI can not be modified using this command, only the routing of unknown packets.

In order to enable forwarding of unknown packets, a default VSI should be defined beforehand.

Table 7-94. Update Port Virtualizer Command (Opcode: 0x0221)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0221	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Command Flags	16-19	0x0	0: Reserved (The Port Virtualizer type can not be modified on the fly). 1: Forward Unknown S-tag. If set, packets with unknown S-tags are forwarded to the Default VSI. 2-31: Reserved
Reserved	20-31	0x0	Reserved

Table 7-95. Update Port Virtualizer Response (Opcode: 0x0221)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0221	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value	6-7		The following error values can be returned: ENOENT - if the SEID do not point to a Port Virtualizer. EACCES - if the Port Virtualizer is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31		Reserved

7.4.9.5.5.3 Get Port Virtualizer Parameters

This command is used to get the parameters of an Port Virtualizer

Table 7-96. Get Port Virtualizer Command (Opcode: 0x0222)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0222	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command

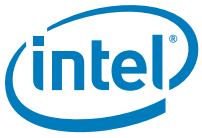


Table 7-96. Get Port Virtualizer Command (Opcode: 0x0222)

Name	Bytes.Bits	Value	Remarks
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID	16-17		The SEID of the Port Virtualizer
Reserved	18-31		Reserved

Table 7-97. Get Port Virtualizer Response (Opcode: 0x0222)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0222	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		The following error values can be returned: ENOENT - if the SEID do not point to a Port Virtualizer. EACCES - if the Port Virtualizer is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-17		The SEID of the requested Port Virtualizer (reflects the command)
Default tag	18-19		If Forward Unknown E-tag is set, the S-tag is used to forward packets with unknown E-tag.
Command Flags	20-21	0x0	0: Port Virtualizer type: • 0 - S-comp • 1 - Port Extender 1: Forward Unknown S-tag. If set, packets with unknown S-tags are forwarded to the Default VSI. 2: Forward Unknown E-tag. If set, packets with unknown E-tags are forwarded as if received with Default S-tag. 2:15: Reserved
Reserved	22-29	Reserved	Reserved
Default Port SEID	30-31		Returns the SEID of the default port.

7.4.9.5.6 VEB/VEPA Commands (Opcode 0x023x)

7.4.9.5.6.1 Add VEB

This command is used to instantiate an VEB on a port/Port Virtualizer. A VEB can be instantiated either as a floating element or can be inserted between an existing VSI and its uplink.

When an Add VEB command is received, the internal Firmware checks if all the requested resources are available and allocate them to the VEB. If part of the resources are not available, the command returns a list of unavailable resources.

Note: After a VEB is added, the *allow loopback* flag of the original VSI should be set using the Update VSI command.

.

Table 7-98. Add VEB command (Opcode: 0x0230)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0230	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
uplink SEID	16-17	0x0	Defines the SEID to which this VEB should be connected. If the *Floating VEB* flag is set, the uplink element should be Null (0).
Downlink SEID	18-19	0x0	Defines the SEID of the downlink VSI to which this VEB is connected. Port type is defined by flags. Should point to valid VSI, currently connected to uplink SEID. Should be Null(0) for Floating VEB
Flags	20-21	0x0	0: Floating VEB: If set it can only send packets to attached VSIs; if cleared it can send and receive packets from the LAN. 2:1: Port type: Defines the type of the VSI defined by the Downlink SEID field. 00 - Reserved 01 - Default 10 - Data Port 11 - Reserved Should be Default (01) for Floating VEB 3: Enable L2 filtering. If set, the L2 filter table should pass in addition to the regular forwarding decision - relevant only for "cloud" and "UDP cloud" NVM images. Reserved otherwise. 15:4 Reserved.
Enabled TCs	22	0x0	A bitmap of the TCs that should be enabled in this VEB. The TCs enabled should be already enabled in the uplink element (should already have a node in the scheduler in the uplink element).
Reserved	23-31	0x0	Reserved

The Enable L2 filtering flag should be set to the same values for all VEB instances in the device. The firmware will use the value set by the first *Add VEB* AQ and will reject subsequent *Add VEB* AQ with a conflicting setting with an EINVAL error.

Table 7-99. Add VEB Response (Opcode: 0x0230)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0230	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - if the uplink SEID do not point to valid elements. ENOSPC - if there aren't enough resources to assign a VEB. EACCES - if the uplink element is not owned by this PF. ERANGE - a TC not enabled in the uplink element was requested. EPERM - if the command is not permitted (for example virtualization is disabled).
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command

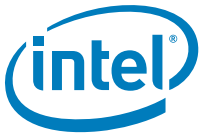


Table 7-99. Add VEB Response (Opcode: 0x0230)

Name	Bytes.Bits	Value	Remarks
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-21		Reserved
Switch ID	22-23		Assigned switch ID. If connected to an S-channel, this is equal to the S-tag, otherwise, it is the switch ID assigned internally by the firmware.
VEB SEID/Error Reasons.	24-25		If the Return value is zero: Bytes 25:24:Returns the SEID of the VEB If Return value is ENOSPC, contains a bitmap that indicates which resources are missing: 0: No VEB left 1: Not enough Scheduler nodes. 2: Not enough statistics counters. 3: Not enough switching entries. 4-31: Reserved.
Statistic counter	26-27	Statistic counters index	Returns an index of the statistics counters block assigned to this VEB. This number is in the 0-15 range and points to the VEB statistics set.
VEBs used	28-29		Number of VEBs used by this function
Total VEBs un-allocated	30-31		Total number of VEBs still un-allocated and not reserved by any function.

7.4.9.5.6.2 Get VEB parameters (0x0232)

This command returns the parameters of the VEB/VEPA

Table 7-100. Get VEB Parameters command (Opcode: 0x0232)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0232	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID	16-17	0x0	Defines the SEID of the VEB
Reserved	18-31	0x0	Reserved

Table 7-101. Get VEB Parameters Response (Opcode: 0x0232)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0232	Command opcode
Datalen	4-5	0x0	Length of buffer

Table 7-101. Get VEB Parameters Response (Opcode: 0x0232)

Name	Bytes.Bits	Value	Remarks
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - if the SEID do not point to a valid VEB/VEPA element. EACCES - if the VEB is not owned by this PF. EPERM - if the command is not permitted (for example virtualization is disabled).
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID	16-17		The SEID requested in the command.
Switch ID	18-19		Assigned switch ID. If connected to an S-channel, this is equal to the S-tag, otherwise, it is the switch ID assigned internally by the firmware.
Flags	20-21	0x0	0: Floating VEB: If set it can only send packets to attached VSIs; if cleared it can send and receive packets from the LAN. 2:1 Reserved 3: L2 filtering is Enabled. If set, the L2 filter table should pass in addition to the regular forwarding decision - relevant only for "cloud" and "UDP cloud" NVM images. Reserved otherwise. 15:4 Reserved.
Statistic counter	22-23	Statistic counters index	Returns an index of the statistics counters block assigned to this VEB.
VEBs used	24-25		Number of VEBs used by this function
Total VEBs un-allocated	26-27		Total number of VEBs still un-allocated and not reserved by any function.
Reserved	28-31		Reserved

7.4.9.5.7 Switch Connectivity Commands (Opcode 0x024x)

7.4.9.5.7.1 Delete Element

This command is used to remove a switching element. A function can remove only an element it controls. The driver should make sure every queue in a VSI is disabled before a VSI is removed, however, queue groups tied to the VSI may be removed together with the VSI. It should also make sure the removed element is not tied to any other element before removing it. An exception to this rule is the case of a VEB, that when removed with a single VSI tied to it, will be replaced with this VSI.

Table 7-102. Delete Element Command (Opcode: 0x0243)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.1 for details.
Opcode	2-3	0x0243	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, will be copied by the FW into the completion of this command

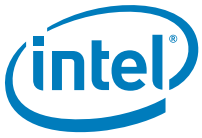


Table 7-102. Delete Element Command (Opcode: 0x0243)

Name	Bytes.Bits	Value	Remarks
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID	16-17	SEID	The SEID of the element to remove
Reserved	18-31	0x0	Must Be zero

Table 7-103. Delete Element Response (Opcode: 0x0243)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.1 for details.
Opcode	2-3	0x0243	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 - if SEID do not point to a valid element. EACCES - if the element is not owned by this PF. EBUSY - if the element to remove has more than one uplink element tied to it.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31		Reserved

7.4.9.5.8 Forwarding Table Configuration Commands (Opcode 0x025x)

Note: All the MAC addresses in the forwarding table configuration commands should be given in big endian format.

7.4.9.5.8.1 Add MAC, VLAN pair (0x0250)

This command is used to add a set of MAC or MAC, VLAN pairs to a set of VSIs. If one of the allocation fails due to lack of resources, the *Set VSI Promiscuous modes* command (Section 7.4.9.5.8.5) may be used to allow forwarding of all the packets of a given type to this VSI.

All the VSIs must have the same SwitchID.

To allow reception of untagged packets only, a MAC, VLAN = 0 filter should be added.

This command should not be used to forward Broadcast packets using hash based filtering. The *Set VSI Promiscuous Modes* command (Section 7.4.9.5.8.5) should be used if broadcast forwarding without VLAN filtering is required or using this command with "Use Hash" flag cleared if broadcast filtering with VLAN filtering is needed.

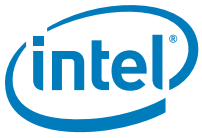
Note: The ToQueue action may be ignored if the packet is forwarded due to multiple rules match (for example, exact match and promiscuous unicast).

Table 7-104. Add MAC, VLAN pair Command (Opcode: 0x0250)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0250	Command opcode
Datalen	4-5		Length of buffer - should be equal to Num_Adresses * 16 bytes
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Num_Adresses	16-17		The number of MAC, VLAN pairs to add
SEID 0	18-19	0x0	Bit 15: Valid SEID Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI.
SEID 1	20-21	0x0	Bit 15: Valid SEID Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI.
SEID 2	22-23	0x0	Bit 15: Valid SEID Bit 14:8: Reserved Bit 9:0: SEID Number of the VSI.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

The command buffer of this command contains the details of the MAC, VLAN pairs to add. It contains a set of *Num_addresses* 16 bytes structures as defined below

Field	Offset	Description
MAC Address	0-5	MAC Address to add
VLAN tag	6-7	VLAN tag to add. To allow reception of untagged packets only, a VLAN ID of zero should be set and the Ignore VLAN flag should be cleared. 23:12: Reserved 11:0: VLAN ID



Field	Offset	Description
Flags	8-9	0: Use perfect match: If set, an perfect match MAC address may be used to create this MAC, VLAN pair 1: Use Hash: If set, a hash MAC address may be used to create this MAC, VLAN pair 2: Ignore VLAN: If set, the VLAN tag is ignored and the MAC address is used to forward packets from all VLANs 3: ToQueue: Use MAC, VLAN to point to a queue. This flag should be set only if the filter points to a single VSI. If a filter points to multiple VSIs, this flag is ignored. 15:4: Reserved ***Note:*** At least one of the Use perfect match or Use Hash flags should be set. If both are set, the Firmware will attempt to use perfect match and will fallback to using hash if the exact match table is full. ***Note:***
Queue Number	10-11	Bit 15:11: Reserved Bit 10:0: Queue number - Valid only if the Flags.ToQueue bit is set. The queue number is relative to the VSI.
Reserved	12-15	Reserved for response part.

Table 7-105. Add MAC, VLAN pair Response (Opcode: 0x0250)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0250	Command opcode
Datalen	4-5		Length of buffer - equals to Num_Adresses * 16 bytes
Return value/VFID	6-7		Return value. The following error values can be returned: ENOSPC - if there aren't enough resources to assign all the MAC, VLAN pairs. The return buffer details which of the allocations failed ENOENT - if the SEID does not point to a valid VSI. EACCES - if the VSI is not owned by this PF. EEXIST - if a queue is assigned to a multicast filter (more than one VSI)
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Perfect match MAC used	16-17		Number of perfect match MAC used by this function
Perfect match MAC un-allocated	18-19		Total number of perfect match MAC still un-allocated and not reserved by any function.
Unicast Hash MAC un-allocated	20-21		Total number of unicast hash MAC still un-allocated.
Multicast Hash MAC un-allocated	22-23		Total number of multicast hash MAC still un-allocated.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

The response buffer of this command contains the results of the MAC, VLAN pairs allocation. It contains a set of *Num_addresses* 16 bytes structures as defined below.

Field	Offset	Description
Reserved	0-11	Reserved for Command part
Matching Method	12	0x0: Perfect Match match was used 0x1: Hash match was used 0x2 - 0xFE: Reserved 0xFF: Request failed due to lack of resources.
Reserved	13-15	Reserved

7.4.9.5.8.2 Remove MAC,VLAN pair (0x0251)

This command is used to remove a set of MAC or a MAC, VLAN pairs from up to 3 VSIs.

All the VSIs must have the same SwitchID.

If a hash MAC was used, the address should be removed only if this hash value is not needed for this VSI(s) anymore.

Table 7-106. Remove MAC, VLAN pair Command (Opcode: 0x0251)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0251	Command opcode
Datalen	4-5		Length of buffer - should be equal to Num_Adresses * 16 bytes
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Num Addresses	16-17		The number of addresses in the command buffer.
SEID 0	18-19	0x0	Bit 15: Valid SEID Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI.
SEID 1	20-21	0x0	Bit 15: Valid SEID Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI.
SEID 2	22-23	0x0	Bit 15: Valid SEID Bit 14:8: Reserved Bit 9:0: SEID Number of the VSI.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

The command buffer of this command contains the details of the MAC, VLAN pairs to remove. It contains a set of *Num_addresses* 16 bytes structures as defined below

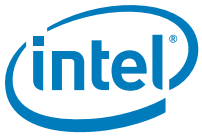


Table 7-107. Remove MAC, VLAN pair Response (Opcode: 0x0251)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0251	Command opcode
Datalen	4-5		Length of buffer. Indicate the number of entries filled by the Firmware (16*Num Addresses) even if the Datalen in the command was larger than that.
Return value/VFID	6-7		Return value. The following error values can be returned: EINVAL - if all the VSIs do not point to the same switchID. ENOENT- if the MAC, VLAN pair does not exist or if one of the SEID does not point to a valid VSI. The details of which remove request failed is found in the response buffer. ***Note:*** If a VSI is not connected to one of the MAC, VLAN to remove, it is silently ignored. EACCES - if one of the VSI is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Perfect match MAC used	16-17		Number of perfect match MAC used by this function
Perfect match MAC un-allocated	18-19		Total number of perfect match MAC still un-allocated and not reserved by any function.
Unicast Hash MAC un-allocated	20-21		Total number of unicast hash MAC still un-allocated.
Multicast Hash MAC un-allocated	22-23		Total number of multicast hash MAC still un-allocated.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

Field	Offset	Description
MAC Address	0-5	MAC Address to remove VSI from
VLAN tag	6-7	VLAN tagto remove VSI from
Flags	8	0: perfect match: If set, an perfect match MAC address was used to create this MAC, VLAN pair 1: Hash: If set, a hash MAC address was used to create this MAC, VLAN pair 2: Reserved 3: Ignore VLAN: If set, the forwarding is currently based only on MAC 4: Remove from all VSIs. This will remove the filter from all the VSIs owned by this PF. 7:5: Reserved
Reserved	9-15	Reserved

The response buffer of this command contains the results of the MAC, VLAN pairs removal. It contains a set of *Num_addresses* 16 bytes structures as defined below.

Field	Offset	Description
Reserved	0-11	Reserved for Command part
Error code	12	0x0: Successful removal 0x1 - 0xFE: Reserved 0xFF: Request failed (The MAC, VLAN pair does not exist).
Reserved	13-15	Reserved

7.4.9.5.8.3 Add VLAN (0x0252)

This command is used to add a set of VLANs to the VLAN table or to add a set of VLAN filters to up to 3 VSIs. All the VSIs must have the same SwitchID.

Note: In order to support egress (Transmit) VLAN filtering, the "Enable VLAN anti spoof" flag should be set in the relevant Add VSI command. In order to support ingress (Receive) VLAN filtering, the Promiscuous VLAN flag in the Set VSI Promiscuous Modes Command should be cleared.

For Private VLAN functionality both ingress and egress VLAN filtering are needed.

Table 7-108. Add VLAN Command (Opcode: 0x0252)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0252	Command opcode
Datalen	4-5		Length of buffer - should be equal to Num_VLAN * 8 bytes
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Num_VLAN	16-17		Number of VLANs to add
SEID 0	18-19	0x0	Bit 15: Valid SEID Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI.
SEID 1	20-21	0x0	Bit 15: Valid SEID Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI.
SEID 2	22-23	0x0	Bit 15: Valid SEID Bit 14:8: Reserved Bit 9:0: SEID Number of the VSI.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

The command buffer of this command contains the details of the VLAN IDs to add. It contains a set of *Num_VLAN* 8 bytes structures as defined below

Table 7-109. Add VLAN Response (Opcode: 0x0252)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0252	Command opcode
Datalen	4-5		Length of buffer - equals to Num_VLAN * 8 bytes
Return value/VFID	6-7		Return value. The following error values can be returned: EINVAL - if all the VSIs do not point to the same switchID. ENOSPC - if there aren't enough resources to assign the VLANs. The response buffer details which of the allocations failed. ENOENT - if one of the VSIs is not valid. EACCES - if one of the VSIs is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-19		Reserved
VLAN used	20-21		Total Number of VLAN used (VLAN is a shared resource).
VLAN un-allocated	22-23		Total number of VLAN still un-allocated.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

Field	Offset	Description
VLAN tag	0-1	VLAN tag to add
Flags	2	0: Local VLAN 2:1: Private VLAN type: • 00: Regular VLAN (not private) • 01: Primary VLAN • 10: Secondary VLAN • 11: Reserved 4:3: Private VLAN Port type (valid only if Private VLAN type is Primary or Secondary VLAN): • 00: Regular VLAN • 01: Promiscuous VSIs • 10: Community VSIs • 11: Isolated VSIs 7:5 Reserved ***Note:*** If Primary or Secondary VLAN is set, the port type can not be zero. If both are not set, it must be zero. ***Note:*** The definition of a VLAN ID should be consistent within a VEB. Thus subsequent commands should set the same Private VLAN type an local VLAN flags when adding VSIs to the same VLAN ID. The Private VLAN Port type can be different for various VSIs within the same VLAN.
Reserved	3-7	Reserved for response part.

The response buffer of this command contains the results of the VLANs allocation. It contains a set of *Num_VLAN* 8 bytes structures as defined below.

Field	Offset	Description
Reserved	0-3	Reserved for Command part
Result	4	0x0: Allocation success. 0x1 - 0xFD: Reserved 0xFE: Request failed due to inconsistent VLAN definition. 0xFF: Request failed due to lack of resources.
Reserved	5-7	Reserved

7.4.9.5.8.4 Remove VLAN (0x0253)

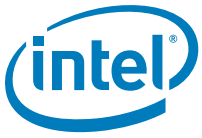
This command is used to remove a set of VLANs from the VLAN table or to remove VLAN filters from up to 3 VSIs.

All the VSIs must have the same SwitchID

Table 7-110. Remove VLAN Command (Opcode: 0x0253)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0253	Command opcode
Datalen	4-5	0x0	Length of buffer - should be equal to Num_VLAN * 8 bytes
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Num_VLAN	16-17		Number of VLANs to add
SEID 0	18-19	0x0	Bit 15: Valid SEID Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI.
SEID 1	20-21	0x0	Bit 15: Valid SEID Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI.
SEID 2	22-23	0x0	Bit 15: Valid SEID Bit 14:8: Reserved Bit 9:0: SEID Number of the VSI.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

The command buffer of this command contains the details of the VLAN pairs to remove. It contains a set of *Num_VLAN* 8 bytes structures as defined below



Field	Offset	Description
VLAN tag	0-1	VLAN tag to remove VSI(s) from
Flags	2	0: remove entire VLAN 7:1 Reserved
Reserved	3	Reserved
Reserved	4-7	Reserved for response part.

Table 7-111. Remove VLAN Response (Opcode: 0x0253)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0253	Command opcode
Datalen	4-5		Length of buffer - equals to Num_VLAN * 8 bytes
Return value/VFID	6-7		Return value. The following error values can be returned: EINVAL - if all the VSIs do not point to the same switchID. ENOENT - if the VLAN does not exist or if one of the SEID does not point to a valid VSI. The response buffer details which of the removal failed. ***Note:*** If a VSI is not connected to one of the MAC, VLAN to remove, it is silently ignored. EACCES - if one of the VSIs is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-19		Reserved
VLAN used	20-21		Total Number of VLAN used (VLAN is a shared resource).
VLAN un-allocated	22-23		Total number of VLAN still un-allocated.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

The response buffer of this command contains the results of the MAC, VLAN pairs removal. It contains a set of *Num_VLAN* 8 bytes structures as defined below.

Field	Offset	Description
Reserved	0-3	Reserved for Command part
Error code	4	0x0: Successful removal 0x1 - 0xFE: Reserved 0xFF: Request failed (the VLAN do not exist or if one of the VSIs is not valid).
Reserved	5-7	Reserved

7.4.9.5.8.5 Set VSI Promiscuous Modes (0x0254)

This command is used to allow a VSI to set various promiscuous mode and other generic filtering options.

Note: If the Default VSI flag is set for a VSI within a VEB, this command will change the default VSI of the VEB to the VSI pointed by this command.

Table 7-112. Set VSI Promiscuous Modes Command (Opcode: 0x0254)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0254	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Flags	16-17		0: Promiscuous Unicast 1: Promiscuous Multicast 2: Promiscuous Broadcast 3: Default VSI - accept packets within the switch ID not matching any specific address to this VSI. 4: Promiscuous VLAN 15:5 Reserved Multiple flags may be set. ***Note:*** Default VSI is supposed to be used only for VMDq1 scenarios where there is no VEB. If this command is used in presence of a VEB, it will change the default VSI of the VEB.
Valid flags	18-19		0: Promiscuous Unicast flag is valid 1: Promiscuous Multicast is valid 2: Promiscuous Broadcast is valid 3: Default VSI is valid 4: Promiscuous VLAN is valid 15:6 Reserved Multiple valid bits may be set.
SEID Number	20-21		Bit 15: Valid SEID - ignored in this field Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI.
VLAN ID	22-23		15: VLAN is valid 14:12: Reserved 11:0: VLAN ID ***Note:*** If bit 15 is set, the Promiscuous Unicast, Multicast, and Broadcast flags applies only to this VLAN, otherwise, these modes applies to all VLANs. ***Note:*** If VSI is in promiscuous VLAN mode, the VLAN ID should not be used.
Reserved	22-23		Reserved
Reserved	24-31	0x0	Reserved

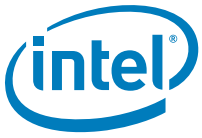


Table 7-113. Set VSI Promiscuous Modes Response (Opcode: 0x0254)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0254	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - If the SEID doesn't point to a valid VSI. ENOSPC - if there aren't enough resources to apply the promiscuous rules. EACCES - if the VSI is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31		Reserved - should contain the parameters from the command.

7.4.9.5.8.6 Add S-tag (0x0255)

This command is used to associate an S-tag with a VSI. This command can be given only by the port controlling the Port Virtualizer. This command should be used for cascaded Port Virtualizer VSIs where more than one S-tag points to the same VSI. For regular channels, the *Add VSI* command already adds the single S-tag of the VSI.

An S-tag may be directed to a specific queue in the VSI. This is done by setting the ToQueue flag and providing a valid queue in the QueueNumber field. In order to assign a queue to the default VSI, this command should be used with the default S-tag as tag value.

Note: The first S-tag added to the cascaded Port Virtualizer VSI via the *Add VSI* command is considered as the switch ID for this VSI and should not be manipulated by the Add S-tag and Remove S-tag commands.

Note: A VEB or L2 filters can not be applied to a cascaded VSI on any S-tag. THe VSI should be in default mode for its initial S-tag. The VSI should be defined with a *Connection Type* of *Default Port*.

Table 7-114. Add S-tag command (Opcode: 0x0255)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0255	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7	0x0	Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Flags	16-17		0: ToQueue 15:1: Reserved

Table 7-114. Add S-tag command (Opcode: 0x0255)

Name	Bytes.Bits	Value	Remarks
SEID	18-19	0x0	Bit 15: Valid SEID - ignored in this field Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI.Defines the VSI that uses this S-tag. This VSI should belong to the Port Virtualizer through which this command is received
Tag	20-21	0x0	The value of the S-tag
Queue Number	22-23	0x0	Bit 15:11: Reserved Bit 10:0: Queue number - Valid only if the Flags.ToQueue bit is set. The queue number is relative to the VSI.
Reserved	24-31		Reserved

Table 7-115. Add S/E-tag Response (Opcode: 0x0255)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0255	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - if the SEID do not point to a valid element ENOSPC - if there aren't enough resources to assign a tag. EACCES - if the VSI is not owned by this PF. EEXIST - if this tag already points to another VSI. EPERM - Attempt to add tags to a VSI which is not a cascaded type.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-27	0x0	
S/E-tags used	28-29		Number of tags used by this PF.
S/E-tags un-allocated	30-31		Total number of tags still un-allocated.

7.4.9.5.8.7 Remove S-tag (0x0256)

This command is used to remove an S-tag from the forwarding table of a VSI.

Note: This command should be used only to remove tags in a cascaded Port Virtualizer VSIs.

Table 7-116. Remove S/E-tag Response (Opcode: 0x0256)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x0256	Command opcode
Datalen	4-5	0x0	Length of buffer

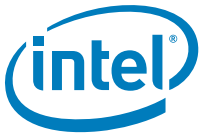


Table 7-116. Remove S/E-tag Response (Opcode: 0x0256)

Name	Bytes.Bits	Value	Remarks
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - if the SEID do not point to a valid element EACCES - if the VSI is not owned by this PF. ENXIO - this S-tag do not point to this VSI. EPERM - Attempt to remove tags from a VSI which is not a cascaded type.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-27		Reserved
S/E-tag used	28-29		Number of S/E-tags used by this function
S/E-tag un-allocated	30-31		Total number of S/E-tags still un-allocated.

Table 7-117. Remove S/E-tag Command (Opcode: 0x0256)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0256	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
VSI SEID	16-17	0x0	Bit 15: Valid SEID - ignored in this field Bit 14:10: Reserved Bit 9:0: SEID Number of the VSI. Defines the VSI SEID that uses this tag. This VSI should belong to the Port Virtualizer through which this command is received
S/E-tag	18-19	0x0	The value of the tag
Reserved	20-31	0x0	Reserved

7.4.9.5.8.8 Update S-tag (0x0259)

This command is used to update the S-tags associated with a VSI. This command can be given only by the port controlling the Port Virtualizer. This command can be used only for cascaded port virtualizer VSIs.

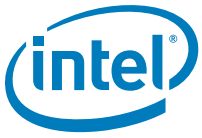
If a queue was associated with the previous tag, the association is kept for the new tag.

Table 7-118. Update S/E-tag command (Opcode: 0x0259)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0259	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7	0x0	Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID Number	16-17	0x0	Bit 15: SEID Valid - ignored in this field. Bit 14:10: Reserved Bit 9:0: Defines the SEID of the VSI that uses this tag. This VSI should belong to the PF through which this command is received
Old tag	18-19	0x0	The original value of the S-tag. If the value is zero, it is assumed the VSI didn't had a tag.
New tag	20-21	0x0	The new value of the tag
Reserved	22-31		Reserved

Table 7-119. Update S-tag Response (Opcode: 0x0259)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0259	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - if the SEID do not point to a valid element ENOSPC - if there aren't enough resources to assign a new tag. EACCES - if the VSI is not owned by this PF. EEXIST - if the new tag already points to another VSI. EPERM - Attempt to modify a tags in a VSI which is not a cascaded Port Extender type.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-27	0x0	
S-tags used	28-29		Number of S-tags used by this PF
S-tags un-allocated	30-31		Total number of S-tags still un-allocated.



7.4.9.5.8.9 Add Control Packet Filter (0x025A)

This command is used to add a control filter to forward packets to a control VSI. This function should be used to forward packets to control VSIs, however, there is no enforcement of this rule and the driver may use it to forward control packets to other VSIs. Only packets reaching the switching element will be forwarded by this rule. This means that:

- If the VSI is connected as a control port of the MAC, a Port Virtualizer or a VEB directly connected to the MAC, this filter will apply to untagged packets (no S-tag).
- If the VSI is connected to an S-channel, either directly or as a VEB port, only packet tagged with the S-channel tag will be forwarded.

Note: These filters are exclusive. If a request to set a filter on an existing flow type is received, it will be rejected with an EEXIST reason code. If a filter is set to forward an Ethertype ignoring the MAC address (Ignore MAC = 1), a filter with the same Ethertype and a MAC address should not be applied within the same switch.

Note: The ethertype programmed by this command should not be one of the L2 tags ethertype (VLAN, E-tag, S-tag, etc.) and should not be IP or IPv6.

Table 7-120. Add Control Packet Filter Command (Opcode: 0x025A)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See for details.
Opcode	2-3	0x025A	Command opcode
Datalen	4-5		Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
MAC Address	16-21		The MAC address to use in the filter
Ethertype	22-23		The Ethertype to use: The 16-bit value of the Ethertype
Command Flags	24-25		0: Ignore MAC. If set, forwarding is based only on Ethertype 1: Drop filter. If set, packets received or sent with this Destination MAC address and Ethertype are dropped. 2: ToQueue. If set, the packets matching this filter is sent to a specific queue. ***Note:*** If cleared, the regular queuing mechanism are used. A queue defined by a lower priority switch filter (for example MAC filter) is ignored. 3: Direction: • 0: Apply to Rx traffic. • 1: Apply to Tx traffic. 15:4 Reserved.
SEID Number	26-27		Bit 15: SEID Valid - ignored in this field. Bit 14:10: Reserved Bit 9:0: Defines the SEID of the control VSI that should get the packet.
Queue Number	28-29		Queue to send the packet to if the ToQueue flag is set. The queue number is relative to the VSI.
Reserved	30-31		Reserved

Table 7-121. Add Control Packet Filter Response (Opcode: 0x025A)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x025A	Command opcode
Datalen	4-5		Length of buffer
Return value/VFID	6-7		Return value. The following error values can be returned: ENOSPC - if there aren't enough resources to assign the requested filter. ENOENT - if the VSI is not valid. EACCES - if the VSI is not owned by this PF. EEXIST - if such a filter already exists and is allocated to another VSI.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
MAC, Ethertype used	16-17		Number of perfect match MAC, Ethertype used by the device
Ethertype used	18-19		Number of perfect Ethertype used by the device
MAC, Ethertype not allocated	20-21		Number of perfect match MAC, Ethertype still un-allocated.
Ethertype not allocated	22-23		Number of perfect Ethertype still un-allocated.
Reserved	24-31		Reserved

7.4.9.5.8.10 Remove Control Packet Filter (0x025B)

This command is used to remove a control filter. The filter to remove is defined by the switching element to which the VSI is connected. This means that:

- If the VSI is connected as a control port of the MAC, a Port Virtualizer or a VEB directly connected to the MAC, the removed filter relates to untagged packets (no S-tag).
- If the VSI is connected to an S-channel, either directly or as a VEB port, the removed filter relates to packet tagged with the S-channel tag.

Table 7-122. Remove Control Packet Filter Command (Opcode: 0x025B)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x025B	Command opcode
Datalen	4-5		Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
MAC Address	16-21		The MAC address in the filter
Ethertype	22-23		The Ethertype used: The 16-bit value of the Ethertype

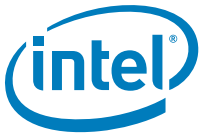


Table 7-122. Remove Control Packet Filter Command (Opcode: 0x025B)

Name	Bytes.Bits	Value	Remarks
Command Flags	24-25		0: Ignore MAC. If set, forwarding is based only on Ethertype 2:1: Reserved 3: Direction: • 0: Apply to Rx traffic. • 1: Apply to Tx traffic. 15:4 Reserved.
SEID Number	26-27		Bit 15: SEID Valid - ignored in this field. Bit 14:10: Reserved Bit 9:0: Defines the SEID of the control VSI to which the filter is associated.
Reserved	28-31		Reserved

Table 7-123. Remove Control Packet Filter Response (Opcode: 0x025B)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x025B	Command opcode
Datalen	4-5		Length of buffer
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT- if the VSI is not connected to the MAC, Ethertype or Ethertype filter to remove. EACCES - if the VSI is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
MAC, Ethertype used	16-17		Number of perfect match MAC, Ethertype used by the device.
Ethertype used	18-19		Number of perfect Ethertype used by the device.
MAC, Ethertype not allocated	20-21		Number of perfect match MAC, Ethertype still un-allocated.
Ethertype not allocated	22-23		Number of perfect Ethertype still un-allocated.
Reserved	24-31		Reserved

7.4.9.5.8.11 Add Cloud Filters (0x025C)

This command is used to add a set of cloud filters to a VSI. The filters set by this commands are the filters specific to cloud formats. To add MAC or MAC, VLAN filters, the *Add MAC, VLAN pairs* (0x0250) command should be used.

Note: As the response can not contain all the cloud resources available, the driver needs to use the *Get Switch Resources Allocation* command (0x0204) to get the resources left after allocation of a filter (Section 7.4.9.5.3.5).

Note: (A0) These filters are available only for VSIs defined as Cloud VSIs in the Add VSI or Update VSI commands (Flags.CloudVSI = 1).

Table 7-124. Add Cloud Filters Command (Opcode: 0x025C)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x025C	Command opcode
Datalen	4-5		Length of buffer - should be equal to 64 * Num_filters bytes.
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Num_filters	16		The number of filters to add
Reserved	17	0x0	Reserved
SEID	18-19	0x0	Bit 15: SEID Valid - ignored in this field (SEID is always valid). Bit 14:10: Reserved Bit 9:0: Defines the SEID of the VSI
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 details of the filters to add. It contains a set of *Num_filters* 64 bytes structures as defined below

Table 7-125. Add Cloud Filters Response (Opcode: 0x025C)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x025C	Command opcode
Datalen	4-5		Length of buffer - equals to 64 * Num_filters bytes.
Return value/VFID	6-7		Return value. The following error values can be returned: ENOSPC - if there aren't enough resources to assign all the filters. The return buffer details which of the allocations failed ENOENT - if the VSI is not valid. EACCES - if the VSI is not owned by the offload PF.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Num_filters	16		The number of filters in the response buffer
Reserved	17-23		
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

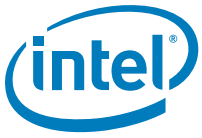


Table 7-126. Add Cloud Filters - command buffer

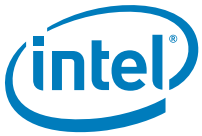
Field	Offset	Description
Outer MAC Address	0-5	Outer MAC Address to add
Inner MAC Address	6-11	Inner MAC Address add
Inner VLAN tag	12-13	Inner VLAN tag to add (only 12 LSBits are relevant to filter).
IP	14-29	IP address to add. For IPv4 addresses, bytes 14-25 are reserved
Command Flags (A0)	30-31	6:0: Filter type: • 0x0: Reserved • 0x1: Outer IP (for GRE packets) • 0x2: Outer IP, GRE Key (for GRE packets) • 0x3: Inner MAC, Inner VLAN (for NVGRE or VXLAN packets) • 0x4: Inner MAC, Inner VLAN, GRE Key (for NVGRE packets) • 0x5: Inner MAC, Inner VLAN, Outer IP (for NVGRE packets) • 0x6: {Inner MAC, GRE} (NVGRE packet) or {Inner MAC, VN Key } (VXLAN packets). • 0x7: Inner MAC, Inner VLAN, VN Key (for VXLAN packets) • 0x8: Reserved • 0x9: Outer MAC L2 filter • 0xA: Inner MAC filter • 0xB - 0x3F: Reserved. ***Note:*** Filter 0x7 is relevant only to MAC in UDP cloud mode (“UDP cloud image”). ***Note:*** Filters 0x1,0x2, 0x4 and 0x5 are relevant only for the other (non MAC in UDP) modes (“cloud” image). ***Note:*** Filters 0x3, 0x6, 0x9 and 0xA are relevant to both modes. ***Note:*** An Outer MAC L2 filter (0x9) should be added only once per MAC relevant MAC address. This filter applies to all the VSI of type cloud. 7: ToQueue: Use MAC, VLAN to point to a queue. Not relevant if filter type = 0x9. 8: IP address type: 0 = IPv4, 1 = IPv6. Relevant only for “cloud” image for filter types 0x1, 0x2, and 0x5. 15:9: Reserved.

Table 7-126. Add Cloud Filters - command buffer

Field	Offset	Description
Command Flags (B0)	30-31	6:0: Filter type: • 0x0: Reserved • 0x1: Outer IP (for GRE packets) • 0x2: Reserved • 0x3: Inner MAC, Inner VLAN (for NVGRE, VXLAN or Geneve packets) • 0x4: Inner MAC, Inner VLAN, Tenant ID (for NVGRE, VXLAN Geneve packets) • 0x5: Reserved Geneve • 0x6: {Inner MAC, Tenant ID} (NVGRE packet or VXLAN Geneve packets). • 0x7: Reserved • 0x8: Reserved • 0x9: Outer MAC L2 filter • 0xA: Inner MAC filter • 0xB: Outer MAC, Tenant ID, Inner MAC • 0xC: Inner IP • 0xD - 0x3F: Reserved. ***Note:*** Filter 0xB should not be used in modes with port extenders . ***Note:*** An Outer MAC L2 filter (0x9) should be added only once per relevant MAC address. This filter applies to all the VSI of type cloud. 7: ToQueue: Use MAC, VLAN to point to a queue. Not relevant if filter type = 0x9. 8: IP address type: • 0 = IPv4, • 1 = IPv6. Relevant only for filter types 0x1, 0x2, and 0x5. 12:9: Tunnel Type: • 0x0: VXLAN • 0x1: NVGRE or other MAC in GRE • 0x2: Geneve0x3: IP in GRE • 0x4: Reserved 0x5-0xF: Reserved Relevant only for filter types 0x4, 0x6, and 0xB. 15:13: Reserved.
Tenant ID	32-35	Key to add: • NVGRE tunnel type: TNI • VXLAN/Geneve tunnel type: VNI • Other GRE tunnel type: GRE Key ***Note:*** The Tenant ID is 3 bytes (32..34) and byte 35 is reserved.
Reserved	36-39	Reserved
Queue Number	40-41	Bit 15:11: Reserved Bit 10:0: Queue number - Valid only if the Command Flags.ToQueue bit is set. The queue number is relative to the VSI.
Reserved	42-55	Reserved
Reserved	56-63	Reserved for response part.

The response buffer of this command contains the results of the filters allocation. It contains a set of *Num_filters* 64 bytes structures as defined below.

Field	Offset	Description
Reserved	0-55	Reserved for Command part
Allocation result	56	0x0: Filter assigned 0x1 - 0xFE: Reserved 0xFF: Request failed due to lack of resources.
Reserved	57-63	Reserved



7.4.9.5.8.12 Remove Cloud filters (0x025D)

This command is used to remove a set of cloud filters from a VSI. The filters removed by this commands are the filters specific to cloud formats. To remove MAC or MAC, VLAN filters, the *Remove MAC, VLAN pairs* (0x0251) command should be used.

Table 7-127. Remove Cloud Filters Command (Opcode: 0x025D)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x025D	Command opcode
Datalen	4-5		Length of buffer - should be equal to 64 * Num_filters bytes.
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Num_filters	16		The number of filters to remove
Reserved	17	0x0	Reserved
SEID	18-19	0x0	Bit 15: SEID Valid - ignored in this field. Bit 14:10: Reserved Bit 9:0: Defines the SEID of the VSI
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 details of the filters to remove. It contains a set of *Num_filters* 16 bytes structures as defined in the *Add Cloud filters* command buffer (Table 7-126).

Note: The ToQueue flag and Queue Number field are not used by the *Remove Cloud Filter* command.

Table 7-128. Remove Cloud Filters Response (Opcode: 0x025D)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.2 for details.
Opcode	2-3	0x025D	Command opcode
Datalen	4-5		Length of buffer - equals to 64 * Num_filters bytes.
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT- if the VSI is not connected to one of the cloud filters or the cloud filter entry does not exist or if one of the VSIs is not valid. The details of which remove request failed is found in the response buffer. EACCES - if the VSI is not owned by the offload PF (VSI is on another port).
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command

Table 7-128. Remove Cloud Filters Response (Opcode: 0x025D)

Name	Bytes.Bits	Value	Remarks
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Num_filters	16		The number of filters in the response buffer.
Reserved	17-23	0x0	Reserved
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

The response buffer of this command contains the results of the filters allocation. It contains a set of *Num_filters* 16 bytes structures as defined below.

Field	Offset	Description
Reserved	0-13	Reserved for Command part
Allocation result	14	0x0: Successful removal 0x1 - 0xFE: Reserved 0xFF: Request failed (one of the VSIs is not connected to this filter or the filter entry does not exist).
Reserved	15	Reserved

7.4.9.5.9 Mirroring Commands (Opcode 0x026x)

The mirroring behavior is described in Section 7.4.6.2.1.

7.4.9.5.9.1 Add Mirror Rule (0x0260)

This command is used to add a mirror rule to a specific switch. Mirror rules are supported for VEBs or VEPA elements only

Table 7-129. Add Mirror Rule Command (Opcode: 0x0260)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID	16-17		Defines the SEID of the switch to which the rule refers.

Table 7-129. Add Mirror Rule Command (Opcode: 0x0260)

Name	Bytes.Bits	Value	Remarks
Rule Type	18-19		2:0 Rule Type: • 000b: Reserved • 001b: Virtual port ingress mirroring • 010b: Virtual port egress mirroring • 011b: VLAN mirroring • 100b: All ingress traffic (to this switch). Includes traffic sent from all VSIs connected to this switch. Does not include traffic received from LAN. • 101b: All egress traffic (from this switch). Includes traffic sent to all VSI in this switch. Does not include traffic sent to the LAN • 11xb: Reserved. 15:3: Reserved ***Note:*** If the Rule Type is 100b (all ingress) or 101b (all egress), then there is no associated buffer. The Flags in bytes 0-1 should be set accordingly.
Number of mirrored entries	20-21		Defines the number of VSI/VLANs that should be mirrored. The values are in the command buffer.
Destination VSI	22-23		Defines the VSI SEID to which the packets matching the mirror rule will be mirrored.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

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 7-130.

Table 7-130. Add Mirror Rule command buffer

Byte	Description
0-1	Mirrored VSI (SEID) or VLAN ID. For all rule types but Ingress VLAN mirroring, the values are SEIDs of VSI. For Ingress VLAN mirroring rule type, the values are VLAN IDs. The VSIs in the list should be part of the switch defined by the SEID.

The following table describes the Add Mirror rule response (with no buffer)

Table 7-131. Add Mirror Rule Response (Opcode: 0x0260)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0260	Command opcode
Datalen	4-5		Length of buffer - equals to 2 * Number of mirrored entries.
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - if the mirror SEID do not point to a valid switch element or the SEID of the mirrored VSI does not point to a valid VSI. ENOSPC - if there aren’t enough resources to assign an mirror rule EACCES - if the VEB is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command

Table 7-131. Add Mirror Rule Response (Opcode: 0x0260)

Name	Bytes.Bits	Value	Remarks
Reserved	16-17		Reserved
Rule ID	18-19		Defines the rule ID that will be 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. The rule ID is not relevant for the Ingress VLAN mirroring rule type (011b):
Mirror rules used	20-21		Number of mirror rules used by this function
Mirror rules un-allocated	22-23		Total number of mirror rules still un-allocated.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

7.4.9.5.9.2 Delete Mirror Rule (0x0261)

This command is used to delete an existing mirror rule. If the rule to remove is of Ingress VLAN mirroring type, a buffer should be provided containing the currently mirrored VLAN to remove.

Table 7-132. Delete Mirror Rule Command (Opcode: 0x0261)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0261	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID	16-17		Defines the SEID of the switch to which the rule refers.
Rule Type	18-19		2:0 Rule Type: • 000b: Reserved • 001b: Virtual port ingress mirroring • 010b: Virtual port egress mirroring • 011b: Ingress VLAN mirroring • 100b: All ingress traffic (to this switch) • 101b: All egress traffic (from this switch) • 11xb: Reserved. 15:3: Reserved
Number of mirrored entries	20-21		Defines the number of VSI/VLANs that should be mirrored. The values are in the command buffer.
Rule ID	22-23		Defines the rule ID that is returned in the receive descriptor. This ID identifies the rule to delete. This ID is the number returned from the Add Mirror Rule response. Relevant only if rule type is not *Ingress VLAN mirroring* (011b).
Data Address high	24-27		Address of buffer - relevant only if rule type is *Ingress VLAN mirroring* (011b)..
Data Address low	28-31		

The structure of the buffer used to indicate the VLAN to remove is as follow:

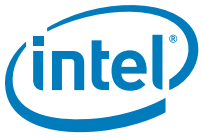


Table 7-133. Delete Mirror Rule command buffer

Byte	Description
0-1	Mirrored VLAN ID to remove. 15:12: Reserved 11:0: VLAN ID

The following table describes the Delete Mirror rule response (with no buffer)

Table 7-134. Delete Mirror Rule Response (Opcode: 0x0261)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0261	Command opcode
Datalen	4-5		Length of buffer
Return value/VFID	6-7		Return value. The following error values can be returned: ENOENT - if the SEID do not point to a valid switch element EINVAL - if the Rule ID doesn't exist. EACCES - if the VEB is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-19		
Mirror rules used	20-21		Number of mirror rules used by this function
Mirror rules un-allocated	22-23		Total number of mirror rules still un-allocated.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

7.4.9.5.10 Storm Control Commands (Opcode 0x028x)

7.4.9.5.10.1 Set Storm Control Configuration

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 7-135. Set Storm Control Configuration (Opcode: 0x0280)

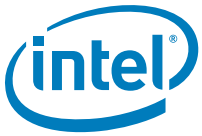
Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0280	Command opcode
Datalen	4-5	0x0	Length of buffer

Table 7-135. Set Storm Control Configuration (Opcode: 0x0280)

Name	Bytes.Bits	Value	Remarks
Return value/VFID	6-7		Return value. Zeroed by driver.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Broadcast Threshold	16-19	0x0	18:0 Traffic Upper Threshold-size: Represents the upper threshold for broadcast storm control. 31:19: Reserved
Multicast Threshold	20-23	0x0	18:0 Traffic Upper Threshold-size: Represents the upper threshold for Multicast storm control. 31:19: Reserved
Storm Control Control	24-27	0x0	0: MDIPW: Drop multicast packets (excluding flow control and manageability packets) if multicast threshold is exceeded in previous window 1: MDICW: Drop multicast packets (excluding flow control and manageability packets) if multicast threshold is exceeded in current window 2: BDIPW: Drop broadcast packets (excluding flow control and manageability packets) if broadcast threshold is exceeded in previous window 3: BDICW: Drop broadcast packets (excluding flow control and manageability packets) if broadcast threshold is exceeded in current window 4: BIDU: BSC Includes Destination Unresolved packets: If bit is set, unicast received packets with no destination pool and sent to the default VSI are included in IBSC 7:5: Reserved 17:8 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). 31:18: Reserved ***Note:***
Reserved	28-31	0x0	Reserved

Table 7-136. Set Storm Control Response (Opcode: 0x0280)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0280	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value	6-7		Return value. There is no specific error code for this command
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31	0x0	Reserved



7.4.9.5.10.2 Get Storm Control Configuration

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 7-137. Get Storm Control Configuration command (Opcode: 0x0281)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.1 for details.
Opcode	2-3	0x0281	Command opcode
Datalen	4-5	0x0	Reserved
Return value	6-7	0x0	Must be Zero
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31		Reserved

Table 7-138. Get Storm Control Response (Opcode: 0x0281)

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0281	Command opcode
Datalen	4-5	0x0	Length of buffer
Return value	6-7		Return value. There are no specific error codes for this command.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Broadcast Threshold	16-19	0x0	18:0 Traffic Upper Threshold-size: Represents the upper threshold for broadcast storm control. 31:19: Reserved ***Note:***
Multicast Threshold	20-23	0x0	18:0 Traffic Upper Threshold-size: Represents the upper threshold for Multicast storm control. 31:19: Reserved ***Note:***
Storm Control Control	24-27	0x0	0: MDIPW: Drop multicast packets 1: MDICW: Drop multicast packets 2: BDIPW: Drop broadcast packets 3: BDICW: Drop broadcast packets 4: BIDU: BSC Includes Destination Unresolved packets: 7:5: Reserved 17:8 Interval - BSC/MSC Time-interval-specification: The interval size for applying Ingress Broadcast or Multicast Storm Control. 31:18: Reserved ***Note:***
Reserved	28-31	0x0	Reserved

7.5 Interrupts

7.5.1 Interrupt signaling

The XL710 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 XL710 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 INTPIN field in the NVM defining the interrupt pin (A,B,C,D or none) per PF. It is the NVM programmer responsibility to follow the PCI rules for allocating orderly interrupt pins for the PCI functions. The legacy interrupt is triggered by the interrupt zero per 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 zero per PF. The MSI capability is enabled per PF by the MSI_En bit in the PFPCI_CNF (loaded from the NVM).
- 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 is loaded from a global MSI_X_PF_N and MSI_X_VF_N parameters in the GLPCI_CNF2 register (loaded from the NVM), shared for all PFs and all VFs respectively. The MSI_X_PF_N and MSI_X_PF_N parameters must be lower or equal than the maximum number of MSI-X vectors per functions as described below. The XL710 supports up to 1168 MSI-X vectors that are allocated to PFs and VFs uniformly as a function of the number of enabled PFs and the number of enabled VFs. The maximum number of MSI-X vectors per function is shown in the Table 7-139 below.

Table 7-139. MSI-X vector allocation per function

Index of max Enabled PF	Max number of MSI-X Vectors per PF	Number of Registers per PF indicated as "INTPF"
1 ... 4	1 + 128	128
5 ... 8	1 + 64	64
9 ... 16	1 + 32	32

Index of max Enabled VF	Max number of MSI-X Vectors per VF	Number of Registers per VF indicated as "INTVF" or "INTVP"
1 ... 32	1 + 16	16
33 ... 64	1 + 8	8
65 ... 128	1 + 4	4

Note: The "Index of max Enabled PF" and the "Index of max Enabled VF" are defined by the values of the PCIPFCNT and PCIVFCNT fields respectively in the GLGEN_PCIFCNCNT register (loaded from the NVM)

Note: All registers with attribute "INTPF", "INTVF" and "INTVP" are reflected in the function's space as 512 registers which is the total number of these registers in the device. Still each function may access only its own registers (starting at register index zero) according to the number of registers indicated in Table 7-139.

The INTVP registers are mapped in the PF space in the following manner:

The register index in the PF space for register 'n' of VF 'm' = "Number of Registers per VF" * 'm' + 'n', while 'm' is the relative VF index within the PF and the "Number of Registers per VF" can be 4, 8 or 16 as indicated in the Table 7-139 above.

The VP registers are mapped in the PF space in the following manner:

The register index in the PF space for VF 'm' = 'm', while 'm' is defined the same as above.

A register address in the PF space = Base address of the specific registers + register index * 4

7.5.1.1 Interrupt Enablement

Interrupts are enabled at 3 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 config space per PF.
  - MSI is enabled by the "MSI Enable" flag in the "MSI Capability" structure in the PCI config space per PF.
  - MSI-X is enabled by the "MSI-X Enable" flag in the "MSI-X Capability" structure in the PCI config 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 xxINT_DYN_CTL0 and xxINT_DYN_CTLN registers (where xx=PF or VF).
  - The software driver sets the INTENA flag to enable the relevant interrupt signal. Upon interrupt assertion on the PCIe bus, the INTENA flag and the interrupt level are auto-cleared.
- Enablement of the interrupts per cause by the CAUSE_ENA flag in cause control registers (see Section 7.5.2).
- Interrupt moderation
  - Interrupt Throttling (ITR) is described in Section 7.5.4.1
  - Interrupt rate limiting (INTRL) is described in Section 7.5.4.2

7.5.1.2 Pending Interrupt Array - PBA

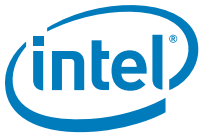
On top of the interrupt signalling on the PCIe bus, the XL710 supports also the standard PBA structure in the MSI-X BAR (see BAR description in Section 12.2.6.1). The PBA is relevant only when MSI-X is enabled. It is described as part of the "MSI-X Capability" structure in Section 12.3.3 for the PF and Section 12.5.3.1 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.

7.5.1.3 Interrupt Sequence

This section describes the interrupt sequence of events starting by an internal events that triggers an interrupt till 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, the hardware executes the steps below in the following order:
   - Process all LAN receive and transmit queues of this interrupt. Write back the status of all completed descriptors that were not reported so far and clear the internal EVENT flags of these queues.



  - — Set the matched bit in the pending interrupt block array (PBA) according to the interrupt moderation policy (represented by the 'ARB' block in the figure below). Note that the PBA is reflected externally only with MSI-X (and not with legacy interrupts nor MSI).
  - — 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 causes. Specific to interrupt zero of the function the software can optionally read the ICR0 register identifying the interrupt causes to be processed. Reading the ICR0 register clears it making it ready to reflect the events of the next interrupt.
5. At the end of the interrupt handler the software re-enables the interrupts by setting the INTENA
  - — On the same register the software sets also the CLEARPBA flag that clears the matched bit in the PBA. In this case it is meaningless since it was already auto-cleared in the previous step.
  - — On the same register the software can update one of three ITRs of this interrupt be setting the ITR_INDX and INTERVAL fields. Setting the ITR_INDX to 11b does not impact the ITRs. See ITR explanation in Section 7.5.4.1.

MSI-X interrupts while interrupts are disabled by the OS

Steps 1 & 2 are the same as above

1. The OS polls the PBA and schedule the interrupt handler

Steps 4 & 5 are the same as above

Legacy interrupts while interrupts are enabled by the OS (mapped to interrupt zero of the PF)

Steps 1 & 2 are the same as above

1. If the interrupt is enabled by the OS (by PCIe setting), the interrupt message is sent to the PCIe.
2. At the very beginning of the interrupt service routing, the software driver clears the internal PBA by setting the CLEARPBA flag in the PFINT_DYN_CTL0 register.
  - — As a result an interrupt de-assertion message is sent on the PCIe bus.
  - — The software driver also reads the PFINT_ICR0 scheduling the matched processes for the active flags in the register
3. During the interrupt handler the software processes each individual interrupt causes. The software can optionally read the ICR0 register identifying the interrupt causes to be processed. Reading the ICR0 register clears it making it ready to reflect the events of the next interrupt.
4. At the end of the interrupt handler the software re-enables the interrupts by setting the INTENA
  - — On the same register the software can update one of three ITRs as explained above.

7.5.2 Interrupt Causes

This section lists all interrupts causes (sources) while mapping these causes to the interrupt vectors is described in Section 7.5.3. Note that only the PF has access to any of the cause registers listed in this subsection. Therefore, VF is required to request its registers programming from the PF by admin command or any other sideband channel.

7.5.2.1 LAN Transmit Queues

The XL710 supports 1536 LAN transmit queues for the whole device while each queue is a potential interrupt cause. A status reporting of a completed transmit 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 the matched QINT_TQCTL[n] register. The queues are mapped to any interrupt vector within the function space by the MSIX_INDX field in the matched QINT_TQCTL[n] register and mapped to any of it's ITRs (or immediate interrupt) by the ITR_INDX in the same register. See ITR description in Section 7.5.4.1. Using interrupt vector zero, the transmit queues are mapped to one of the QUEUE_x flags in the xxINT_ICR0 registers by the MSIX0_INDX field in the matched QINT_TQCTL[n] register. The MSIX0_INDX of PF queues can be set to 0...7 while VF queues can be set only to 0...3.

7.5.2.2 LAN Receive Queues

The XL710 supports 1536 LAN receive queues for the whole device while 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.

7.5.2.3 FCoE DDP Queues

FCoE DDP queues do not trigger interrupts. Instead, FCoE traffic can generate interrupts by FCoE packets or headers that are directed to / from the LAN queues.

7.5.2.4 Other Interrupt Causes

The XL710 supports asynchronous "events" that can generate interrupts for the PFs and the VFs. The "other" interrupt events supported by the PFs are indicated in the PFINT_ICR0 register and enabled by the PFINT_ICR0_ENA register (per PF) as shown in the Table 7-140 below. The "other" interrupt events supported by the VFs are indicated in the *VFINT_ICR0* register and enabled by the *VFINT_ICR0_ENA* register (per VF) as shown in the Table 7-141 below.

The "other" interrupt cause is mapped to MSI-X vector zero of the functions. It is mapped to any of its ITRs (or immediate interrupt) as programmed by the *OTHER_ITR_INDX* field in the *PFINT_STAT_CTL0* register for the PF and the VFINT_STAT_CTL0 register for the VF (for ITR description see Section 7.5.4.1).

During nominal operation it is expected that the *xxINT_ICR0* is used only as a read/clear register by the software. Setting the flags in the *xxINT_ICR0* register (other than the queue flags and the SWINT flag), emulates an interrupt event of the specific cause (if enabled by the *xxINT_ICR0_ENA* register).

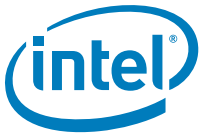


Table 7-140. "Other" interrupt causes of the PFs

PF "other" cause	Description
ECC_ERR	Unrecoverable ECC Error. This bit is set when an unrecoverable error is detected in one of the device memories.
MAL_DETECT	Malicious programming detected
GRST	Global Resets Requested (CORER, GLOBR or EMPR)
PCI_EXCEPTION	The PCI exception is detected as reported in the PFPCI_ICAUSE register.
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.
TIMESYNC	Any of the TimeSync interrupt causes as described in Section 8.5.6.
STORM_DETECT	Indicates a change in 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.
HMC_ERR	HMC error as indicated in the PFHMC_ERRORINFO and PFHMC_ERRORDATA registers.
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.
ADMINQ	Send / Receive admin queues interrupt
SWINT	Software interrupt (detailed in Section 7.5.2.5 below)

Table 7-141. "Other" interrupt causes of the VFs

PF "other" cause	Description
ADMINQ	Send / Receive admin queues interrupt
SWINT	Software interrupt (detailed in Section 7.5.2.5 below)

7.5.2.5 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 may 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 *xxINT_DYN_CTLx* registers ('xx' stands for PF or VF and 'x' stands for '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 *SWINT_TRIG* flag in the *xxINT_DYN_CTLx* registers. Setting the *SWINT_TRIG* flag the software should set also the *INTENA* flag in the same register.

Setting the SW interrupt in vector 0 of the PFs, its status indication is reflected in the *PFINT_ICR0* register (on top of the interrupt triggering).

7.5.2.6 Interrupt Status Registers

During nominal operation the software avoids any possible read accesses. Interrupt zero is used for all the "other" causes which enforce read cycles. The XL710 provides a status indication of all causes of interrupt zero of the PFs and the VFs. The xxINT_ICR0 registers ('xx' stands for PF or VF) provide indication for the following events:

- Any of the "other" interrupt causes
- Up to 8 status flags (QUEUE_0 ... QUEUE_7) in the PFs and up to 4 status flags (QUEUE_0 ... QUEUE_3) in the VFs for any LAN transmit or receive queues associated with interrupt zero
- SWINT indication which is a result of software initiated interrupt

The status indication in the xxINT_ICR0 registers is independent on the xxINT_ICR0_ENA registers setting. The xxINT_ICR0_ENA registers enable interrupt assertion by each of the ICR0 causes. Interrupt on the PCIe is further enabled by the interrupt vector 0 logic (legacy interrupt or MSI or MSI-X 0).

7.5.3 Interrupt Linked List

Once an interrupt is triggered by the ITR logic (described in Section 7.5.4) the hardware process all queues that are associated to the same interrupt. These queues are organized in a linked list as illustrated in Figure 7-53 below. While processing the queues in the linked list, the hardware posts relevant status write back for those descriptors that were not reported already before the interrupt is initiated.

Note 1: The registers that define the linked list are accessible only to the PF. A VF can assign causes to interrupts with the help of the PF using admin command or any other sideband message.

Note 2: The linked list could contain LAN transmit and receive queues. In case that multiple types of interrupt causes are mapped to the same interrupt, it is recommended (for optimized performance) to interleave them in the linked list as possible.

The beginning of the linked list is indicated per interrupt signal by the following parameters in the xxINT_LNKLSTx registers (while 'xx' stands for PF or VP and 'x' stands for '0' or 'N')

- FIRSTQ_TYPE: The type of the first cause in the list assigned to the interrupt signal. The type can be one of the following: Receive Queues or Transmit Queues.
- FIRSTQ_INDX: The queue index of the first cause in the linked list assigned to this interrupt signal. For receive and transmit queues it is the index within the PF space (both PF and its VF queues are defined relative to the PF queue space). The FIRSTQ_INDX could be NULL pointer for no linked list.

An interrupt vector assigned only to the "other" causes, does not have a linked list. In this case, software should set the FIRSTQ_INDX to a NULL value (0x7FF).

The interrupt causes are linked to each other by similar parameters as above in the following cause control registers: QINT_RQCTL; QINT_TQCTL registers ('xx' stands for PF or VP and 'x' stands for '0' or 'N').

- NEXTQ_TYPE: The type of the next cause in the list assigned to the same interrupt signal. It is the same definition as the FIRSTQ_INDX described above.
- NEXTQ_INDX: The queue index of the next cause in the linked list assigned to this interrupt signal. For receive and transmit queues it is the index within the PF space (both PF and its VF queues are defined relative to the PF queue space). The NEXTQ_INDX is a NULL pointer for the last cause in the linked list (equals to 0x7FF).

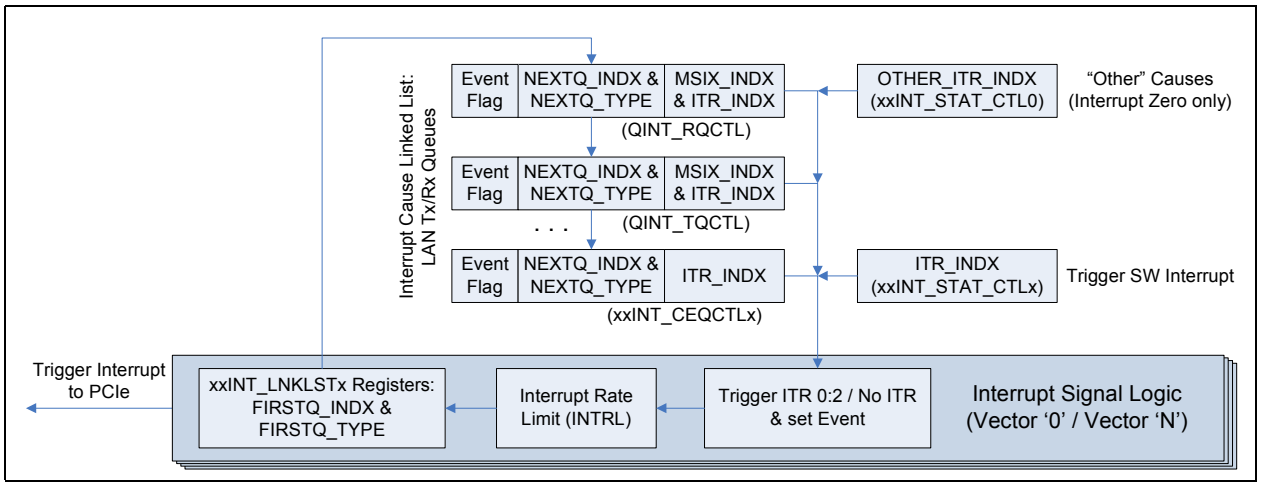


Figure 7-53. Mapping Interrupt Causes to Interrupt Signaling

7.5.3.1 Interrupt Linked List Management

This section describe the required software flow for adding and removing an interrupt cause from an interrupt linked list.

7.5.3.1.1 Initial setting of a linked list

Associating interrupt causes to an interrupt signal can be programmed by the software by any arbitrary order. As long as interrupts are not generated, the linked list is considered as static so any programming order is acceptable.

7.5.3.1.2 Adding an interrupt cause to an active interrupt

Once an interrupt is active, its linked list is considered as dynamic resource. Special programming ordering is required when adding an interrupt cause as follow:

- Program the cause register as required while the NEXTQ_INDX and NEXTQ_TYPE parameters point to the next cause in the linked list.
- Update the NEXTQ_INDX and NEXTQ_TYPE parameters in the previous cause pointing to the added one.
- Note that there is no need to disable the interrupt before these two steps.

7.5.3.1.3 Removing an interrupt cause from an active interrupt

Removing an interrupt cause from an interrupt linked list can be done in one of the following 2 cases:

Case 1 - the interrupt is disabled and it is guaranteed that this interrupt does not have any possible pending interrupts. In this case, the linked list is considered static and there are no special programing ordering. The software should simply update the NEXTQ_INDX and NEXTQ_TYPE parameters in the previous cause to the next cause, skipping the cause that is removed from the linked list. At this point, the software can modify the cause register of the removed interrupt cause if required.

Case 2 - the interrupt is active. In this case, the linked list is considered dynamic resource and the software should follow a strict flow.

- Update the NEXTQ_INDX and NEXTQ_TYPE parameters in the previous cause to the next cause, skipping the cause that is removed from the linked list. In case it is the first cause in the linked list then update the FIRSTQ_INDX and FIRSTQ_TYPE parameters in the matched xxINT_LNKLSTx register, skipping the cause that is removed from the linked list.
- Clear the CAUSE_ENA flag in the matched xINT_xQCTL register.
- The software should wait “long enough” time till it is guaranteed that the hardware fetched the updated value of the previous cause. Waiting “long enough” time could be done by scheduling a software interrupt and waiting for that interrupt that follows.
- At this point, the software can modify the cause register of the removed interrupt cause if required.
- Note that there is no need to disable the interrupt before starting the above sequence.

7.5.4 Interrupt Moderation

The XL710 is able to throttle interrupts in two layered methods: Interrupt Throttling (ITR) and Interrupt Rate limiting (INTRL). These methods are detailed in the following subsections.

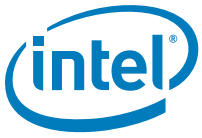
7.5.4.1 Interrupt Throttling (ITR)

Interrupt throttling (ITR) is a mechanism that guarantees a minimum gap between two consecutive interrupts. The XL710 supports 3 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 xxINT_ITRx registers or via the xxINT_DYN_CTLx registers (‘xx’ stands for PF or VF and ‘x’ stands for ‘0’ or ‘N’). It might be useful to set the initial values using the xxINT_ITRx registers and dynamic update by the xxINT_DYN_CTLx registers as explained in step #4 of the interrupt sequence explained in Section 7.5.1.3. When any ITR interval of an interrupt with pending event is expired and the INTRL(*) credit is positive, the hardware follows the following steps:

- Clear the other ITRs of the same interrupt
- Process all causes of the same interrupt (associated to all ITRs) as defined by the linked list (described above in Section 7.5.3).
- (*) See next section for description of the interrupt rate limiting (INTRL)

7.5.4.2 Interrupt rate limiting (INTRL)

Interrupt rate limiting (INTRL) is a credit based mechanism that limits the maximum average number of interrupts per second. The PF controls its interrupt rate limit by the PFINT_RATE0 and PFINT_RATEN registers. The PF also controls its VF’s interrupt rate limit by the VPINT_RATE0 and VPINT_RATEN 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 4 usec units that controls the time gap on which new interrupt credit is gained.

7.6 Virtualization

7.6.1 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 were at its disposal. However, for the limited number of I/O devices, this presents a problem because each operating system might be in a separate memory domain and all the data movement and device management has to be done by a VMM (Virtual Machine Monitor). VMM access adds latency and delay to I/O accesses and degrades I/O performance. Virtualized devices are designed to reduce the burden of VMM by making certain functions of an I/O device 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 OSes using the PCI sig SR-IOV standard (also known as “Native mode” or pass through mode). This mode is called IOV mode in this chapter.
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 XL710 supports fully Next Generation VMDq mode and SR-IOV.

XL710 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.

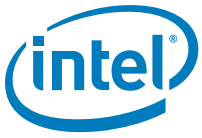
In a virtualized environment, the XL710 serves up to 256 virtual machines (VMs) per device; 128 of these can be directly assigned and any of them can be accessed in Next Generation VMDq mode. See Section 7.6.3 for details of the VFs and VMs resource allocation.

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 XL710, apart from the replication of functions defined in the PCI-sig IOV spec, 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 OSes for which VF drivers are available. Such configurations can benefit from IOV. Others may run older OSes 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 7.4.5.2.

The following section describes the support the XL710 provides for virtualization. This chapter assumes a single-root implementation of IOV and no support for multi-root.



7.6.1.1 Direct Assignment Model

The direct assignment support in the XL710 is built according to the software model defined by the SR-IOV specification.

The PF (Physical function) 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 can not 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 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 the document below refers to the VMM.

A channel is provided between the VF driver and the PF driver through the use of admin commands. See Section 7.10.13.

7.6.1.1.1 Rationale

Direct assignment's purpose 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 in order to minimize the software development effort.

The main time critical operations that require direct handling by the VM are:

1. Maintenance of the data buffers and descriptor rings in host memory. In order 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.
2. Handling of the hardware ring (tail bump and head updates).
3. Interrupt handling.

The capabilities needed to provide independence between VMs are:

1. Per VM reset and enable capabilities.
2. TX QoS control.
3. Allocation of separate CSR space per VM.

The queue context creation, rate control and VF enable capabilities are controlled by the PF.

7.6.1.2 Virtualized System Overview

The following drawings describe the various elements involved in the I/O process in a virtualized system. Figure 7-54 describes the flow in software Next Generation VMDq operation mode and Figure 7-55 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 XL710 or any other means provided by the VMM vendor.

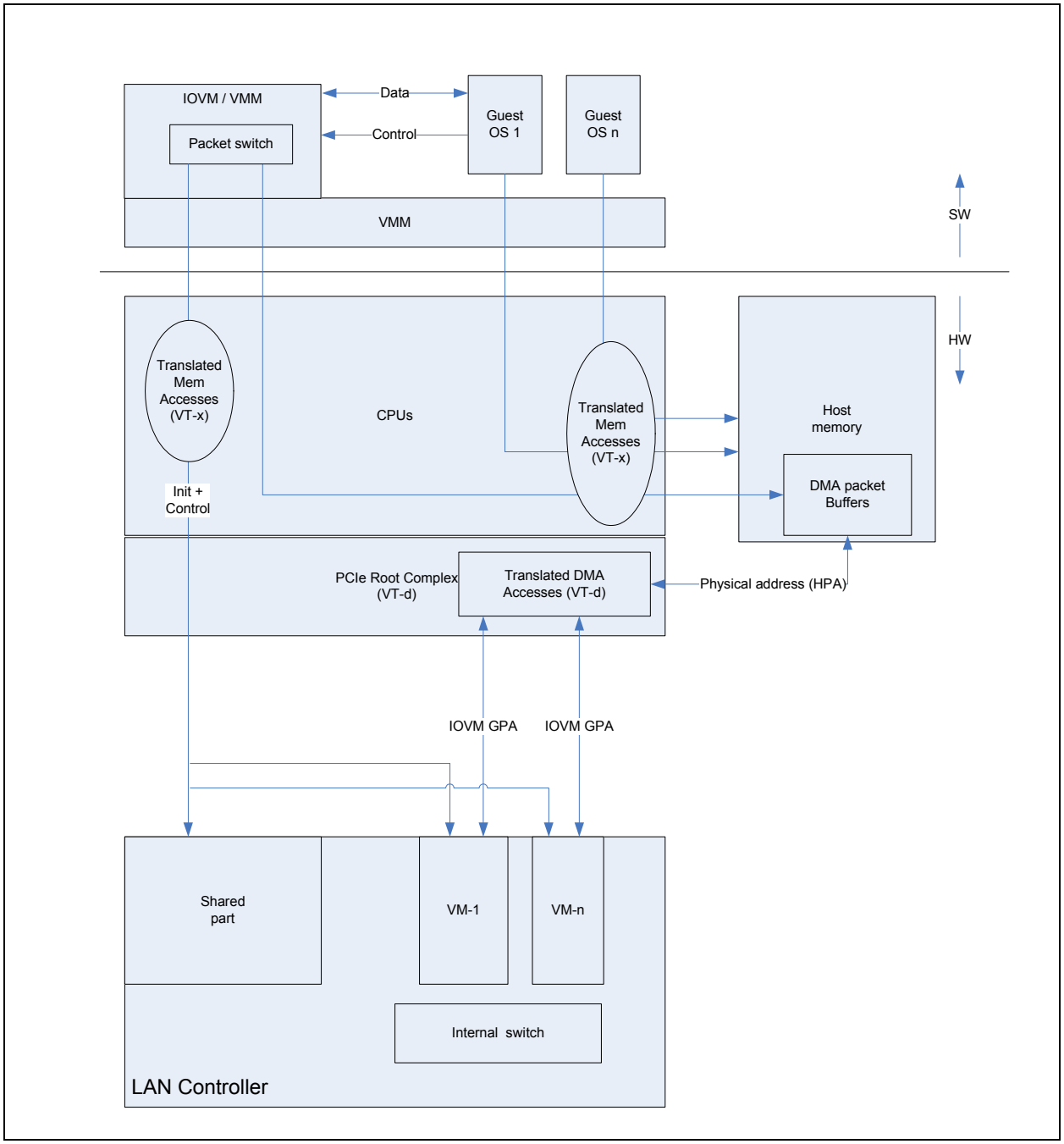


Figure 7-54. VMDq System

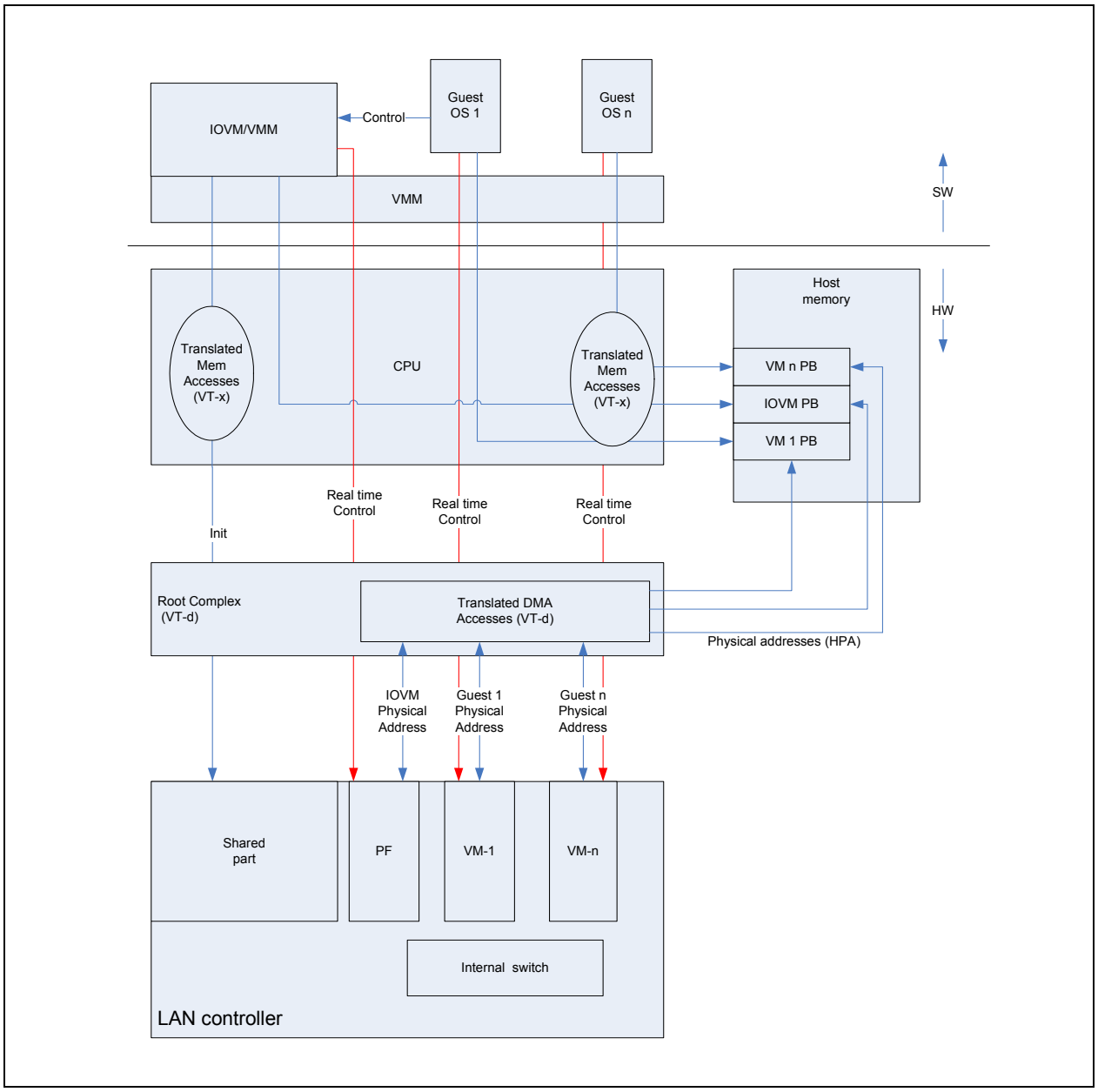


Figure 7-55. SR-IOV Based System

7.6.1.3 Virtualization Supported Features

The XL710 supports a super set of the virtualization features supported in previous products. The following table compares the virtualization features of the XL710 with these of 82599.

Table 7-142. XL710 Versus 82599 Virtualization Support

Feature	XL710 Support	82599 Support
Direct attach (SR-IOV) features		
SR-IOV support	Yes	Yes
ATS support	No	No
VF to PF mailbox	Yes	Yes
Max Number of Virtual functions	128 per device (globally)	64 per port (single queue)
VMDq features		
Max number of Queues allocatable to VMs	1536	128
Max number of queues per VF	16	8
Max number of queues per VMDq2 VSI	16	8
Max number of queues per VMDq1 VM	1 (A VMDq1 VSI may support multiple VMs).	8
Max Number of VMDq2 ports	256 per device (globally)	64 per port (single queue)
MAC addresses	1024 per device (globally)	128 per port
VLAN tags	512 per device (global)	64 per port
Queuing to pool method	SA, VLAN pairs or SA or VLAN	SA or VLAN or (SA and VLAN)
RSS per VF	Yes	No (Single RSS used for all VFs).
DCB in VF	Yes	Yes
Switching modes	VEB, VEPA, EVB	VEB
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 pool	Yes	Yes
per pool statistics	Yes	Yes
Per pool offloads	Yes	Yes
Mirroring	Yes	Yes
Long packet filtering	Global and per pool	Global and per pool
Storm Control	Yes	No
Promiscuous modes per VM	VLAN, Multicast, Unicast	Multicast

For more details about the switching support, see Section 7.4.

7.6.1.3.1 Enablement of Virtualization Features

Virtualization feature enablement can be controlled using a soft SKU. The following table describes the way to enable each of virtualization feature.

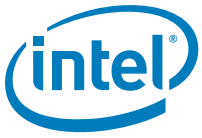


Table 7-143. Virtualization features enablement

Technology	Included features	Enablement
SR-IOV		Set the *GLPCI_CAPSUP.IOV_EN* bit
VEB	VEB, VEPA, storm control, VMDq2, mirroring	Set the *GLGEN_STAT.VTEN* bit
802.1BR/802.1Qbg	Port Extender creation CDCP handling	Set the *GLGEN_STAT.EVBEN* bit

7.6.2 SR-IOV Implementation

7.6.2.1 IOV Concepts

The SR-IOV spec defines the following entities in relation to I/O virtualization:

1. Virtual Machine (VM): A virtual machine to which I/O resources are assigned.
2. A PCIe device: The physical device that might contain a few physical functions - in this case, the XL710.
3. 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.
4. Virtual function (VF): A part of a PF assigned to a VM.

7.6.2.2 IOV Control

In order to control the IOV operation, the physical driver is provided with a set of registers and capabilities. These include:

1. The *PF_VIRT_STATUS* register: Indicates whether SR-IOV is enabled and the number of VFs enabled.
2. Driver to driver communication provided by the Virtualization admin commands (see Section 7.10.13)
3. Switch and filtering control admin commands (described in Section 7.4.9).
4. Reset indications and traffic enables registers per VF using the *GLGEN_VFLRSTAT.VFLRE* bit indicating that a VFLR reset occurred in one of the VFs. When the *GLGEN_VFLRSTAT.VFLRE* bit is set for a given VF, this VF can not send or receive packets. The PF should clear this bit to enable a VF.
5. Malicious driver detection (described below).

The flow used to configure a function when SR-IOV is enabled is described in Section 4.2.3.1.2.

7.6.2.3 PF Only Features

The following features are available only to PFs or controlled solely by the PF:

Note: 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, etc. Flow control packets are sent using the station MAC address stored in the EEPROM. The watermarks of the flow control process and the time-out value are also controllable by the PF only. In a DCB environment, the parameters of per-TC-flow control and the ETS settings are also PF responsibilities.

- 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.

7.6.3 Hardware Resources Assignment to VFs

Table 7-144 describes how the XL710 shared resources are distributed between different VFs. Following is a list of resources that can be assigned to a VF and what is the mechanism used to assign them. Details for each type of resource can be found in the relevant section of this document.

Table 7-144. VF resource allocation

Resource	Allocation method	Detailed description
General resources		
Queues	Dynamic allocation. Each VF can have a different number contiguous of queues (up to 16). Each PF allocates the queues to its VSIs according to the requests of the VMM. The allocation is done using the *Add VSI* (Section 7.4.9.5.4.1) or *Update VSI* (Section 7.4.9.5.4.2) adming commands.	Section 8.2
Traffic classes/Queue Groups	The scheduler nodes are assigned to a VSI. By default 2 nodes are allocated to each VSI. a VM may request more TCs for its VF from the PF when it is created. The allocation is done using the *Add VSI* (Section 7.4.9.5.4.1) or *Configure VSI Bandwidth Limit per Traffic Type* (Section 7.8.4.7) adming commands.	
Interrupt causes and vectors	Fixed according to total number of exposed VFs.The interrupt causes and vectors are allocated to VFs only and not to VMs that are controlled by the PF. These VMs use the PF interrupts. 512 vectors allocated evenly among VFs	TBD
Statistics counters	Each VSI has a set of statistics assigned to it. The allocation is done as part of the VSI creation using the *Add VSI* (Section 7.4.9.5.4.1) admin command.	
Bandwidth	Each VF gets an allocation of transmit bandwidth and is guaranteed it can transmit within the allocation. This arbitration is combined with the Traffic class arbitration. So a rate scheduler is allocated to each TC in the VSI. The allocation is done using the Configure Switching Component Bandwidth Limit per Traffic Type (Section 7.8.4.13) and Configure Switching Component Bandwidth Allocation per Traffic Type (Section 7.8.4.14) adming commands.	Section 7.8.1
FCoE resources		
FCoE DDP DMA contexts	Static allocation - Each function is allocated 4K contexts	
FCoE DDP filter contexts	Static allocation - Each function is allocated 4K contexts	
Switching resources		

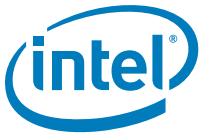


Table 7-144. VF resource allocation (Continued)

Resource	Allocation method	Detailed description
VSIs	Up to 384 VSIs can be directly supported by the XL710. These VSI resources are distributed between the different PFs dynamically. Each VF is guaranteed to receive at least one VSI. The PF may allocate multiple VSIs to a VF. The allocation is done using the using the *Add VSI* (Section 7.4.9.5.4.1) admin command. There is no limit on the number of VSIs that can be assigned to a VF up to the number of Tx/Rx queue pairs the VF is allocated.	
Unicast MAC addresses, Multicast addresses, VLAN tags	The PF driver is responsible for the resource allocation to its VFs. XL710 does not manage the per VF resources of the switch. The PF assigns filters to VFs VSI using the admin commands listed in Section 7.4.9.5.8.	Section 7.4.9.2.2.1
Queueing Resources		
RSS	RSS per VF with 64 entries and up to 16 queues.	

7.6.4 Routing of Traffic

Traffic from and to VFs is handled by the switching elements described in Section 7.4.

7.6.5 VF Driver Interface

VF driver control over the VF functionality can be done either via CSRs directly accessed through the VF BAR or through requests from the PF. This section describes the interface for each type of functionality.

7.6.5.1 Control

There are no dedicated registers that reflect the resources available to a VF. The VF driver can get the needed information through the regular VF to PF channel (admin queues or software based channel).

The status of the entire device is also not apparent to the VF driver, except for the specific VF reset status that can be read from the VFGEN_RSTAT register.

7.6.5.2 Interrupts

Interrupts can be separated into two types:

1. Interrupts relevant to the behavior of each VM. These include Rx & Tx packet sent indications and admin queue events.
2. Interrupts relevant only to the handling of the shared resources. These are mainly error indications, such as packet buffer full and parity errors.

The first type of interrupts are provided directly to the VF driver and the second type are handled by the PF driver. The PF driver may then propagate the interrupt to the VFs via admin queues or some other mechanism.

VFs work only with MSI-X interrupts. Each VF is assigned 5, 9 or 17 MSI-X vectors, depending on the number of exposed VFs (as described in Section 7.5.1).

A VF is responsible for the allocation of the different causes to interrupt vectors and of the interrupt moderation configuration.

Usage of interrupts for VFs is described in Section 7.5.

7.6.5.3 L2 Transmit and Receive

7.6.5.3.1 Queues

Each VF can be assigned up to 16 queue pairs (see Section 8.2.1.2). The allocation of queues is dynamic (see Section 8.2.1.2.3).

The initialization of a queue is not done directly by the VF. It should request a queue initialization from the PF. After the queue is initialized by the PF, the creation of descriptor in the ring and the tail bump via QRX_TAIL/QTX_TAIL registers are done by the VF driver.

7.6.5.3.2 TLP Processing Hints (TPH)

The TPH mechanism is described in Section 3.1.2.6.2.

Given a TPH enabled device, each VM might decide for each queue, on which type of traffic (data, headers, Tx descriptors, Rx descriptors) TPH should be asserted and what is the CPU ID assigned to this queue.The TPH configuration is done as part of the queue initialization via the PF.

7.6.5.3.3 LAN Receive Capabilities

Each VF can decide to queue packets it receives according to different mechanisms:

- RSS
- Traffic Classes.FCoE Exchange ID.

Details of the different queueing modes are described in Section 7.1.1.2.

Configuration of the queuing filters is not done directly by the VF and it should require the PF to do the programming as part of the VSI creation request.

All the regular receive offloads like checksum, header split, etc. described in Section 8.3.4 are available to VFs.

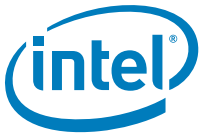
The enablement of these offloads is done by the PF as part of the queue initialization process.

7.6.5.3.4 LAN Transmit Offloads

The following Transmit offloads are exposed through a VF:

- Checksum offloads
- LSO

Details of the different offloads can be found in Section 8.4.4.



7.6.5.4 FCoE specific Functionality

A VF has access to all the FCoE offloads (described in Section 9.0).

There are no registers exposed to the VF for FCoE offloads. The VF may use the special transmit descriptors used to program FCoE DDP contexts (described in Section 9.4.1).

7.6.5.5 Misc. Capabilities

7.6.5.5.1 Statistics

The VF is not directly exposed to statistics counters. If a VF needs to get statistics for its traffic, it should request the information from the PF. Alternatively, the PF driver may notify the VF when significant statistical event occurs (for example, if the number of packets received since last update passes a given threshold).

7.6.5.5.2 IEEE 1588

IEEE 1588 is a per link function and thus is controlled by the PF driver. VMs have access to the real time clock register via the *PRTTSYNTIME_H* and *PRTTSYTIME_L* registers.

7.6.5.5.3 Free Running Timer

The free running timer is a PF driver resource VMs can access. This register is read only to all VFs. It is reset only by PCI reset (see Section 11.2.2.1.27).

7.7 Data Center Bridging (DCB)

This chapter assumes the reader is familiar with the following specifications:

- IEEE P802.1Qbb-2011 a.k.a. Priority-based Flow Control (PFC) spec
- IEEE P802.1Qaz-2011 a.k.a. Enhanced Transmission Selection for Bandwidth Sharing Between Traffic Classes (ETS) spec
  - DCB Center Bridging Exchange Protocol (DCBX) spec refers to Clause 38 in ETS spec
- IEEE Std 802.1AB-2009 a.k.a. Link Layer Discovery Protocol (LLDP) spec

7.7.1 Receive Path DCB

7.7.1.1 Receive Path Enhanced Transmission Selection (ETS)

7.7.1.1.1 Identifying Low Latency (LL) Traffic in Receive

Only two types of traffic are identified along the receive data path, Low Latency (LL) and Bulk (B). This approach requires an a priori differentiation between low latency and bulk traffic. Traffic Class (TC) indexes are used for that purpose according to a setting made in PRTDCB_RETSC.LLTC bitmap.

The entity that runs DCBX shall configure all non-ETS TCs as low latency traffic and only these TCs.

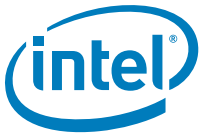
7.7.1.1.2 User Priority (UP) to Traffic Classes (TC) in Receive

7.7.1.1.2.1 Mapping of UP to TC in Receive

Register PRTDCB_RUP2TC controls the mapping of incoming packets to TCs according to the UP field they carry. 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. Refer to Section 7.7.1.1.5.

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.

The GL_SWT_L2TAGCTRL.HAS_UP bit indicates per tag type if its UP is candidate for DCB usage. The UP for DCB is taken from the first tag encountered in the packet that has this bit set. The same method is used both for Receive and loopback traffic. This bit should be set for S-tags and VLAN (internal and external) EtherTypes.



7.7.1.1.2.2 Remapping of the UP Field in Receive

Refer to Section 7.4.6.1.6.2.

The remapping scheme applies also to the traffic destined to EMP (or BMC), which is represented in the tables by its own four VSI numbers.

7.7.1.1.3 Receive Flow for ETS

The XL710 does not implement a true IEEE802.1Qaz standard ETS scheduler on its Rx path. Instead it does some basic arbitration across the TCs (and UPs inside a TC) as described below.

Receive Linked Lists

The total available PCIe bandwidth allocated for traffic reception is *firstly* allocated amongst the LAN ports, and *secondly* allocated amongst traffic classes (and amongst the different UPs attached to it) for each LAN port separately. Bandwidth allocated to a Traffic Class (TC) over a LAN port may differ to the traffic allocated to the same TC index over another LAN port. Since the number of TCs to be supported over a port may vary from 1 to 8 as per DCBX exchange, for simplicity, the Rx packet buffer is organized in a fixed manner into 32 receive linked lists, one for each UP over each LAN port.

Receive Arbitration Levels

The two levels of bandwidth allocation described above, one across LAN ports and one across the traffic classes of a LAN port, induce two levels of arbitration. One Port Round Robin arbiter (PRR) arbiter which is used to select the Rx LAN port to be handled; and one ETS-like arbiter which is used to select the traffic class to be handled within the selected LAN port. Since both arbitration schemes do not deal with absolute throughput allocation, no PCIe bandwidth should be wasted in case for some time some LAN ports or traffic classes offer less workload than allocated to them. Unused bandwidth by a traffic class is proposed first to other traffic classes over the LAN port, and secondly to other LAN ports.

7.7.1.1.4 Receive ETS Arbiter

XL710's receive ETS-like arbiters determine the order in which the per UP linked lists of a port are serviced at the Rx packet buffer’s exit. Note that within each linked list, packets are drained in the order they arrived, indifferently whether they arrived from the Rx port or from the Tx loopback path of the port.

The arbitration algorithm between the linked lists is either Strict Priority or packet-based Round Robin.

Two operating modes are supported according to setting made in NON_ETS_MODE bit in PRTDCB_RETSC register:

1. **Strict Priority (SP) mode** - TCs are served in a strict priority order between them starting from the TC with the higher index until it has no workload, and so forth down to the TC with the lowest index.
2. **Round Robin (RR) mode** - TCs are served in a packet-based round-robin manner between them until no workload is left to any of them.

When several UPs are mapped (by DCBX) to the same TC index in PRTDCB_RUP2TC register, two operating modes are supported:

1. **Strict Priority (SP) mode** - This mode is selected by setting UPINTC_MODE field to 0b for the corresponding TC index in PRTDCB_RETSTCC register array. UPs are served in a strict priority order

between them within the total bandwidth allocated to the TC. Whenever the TC is selected by ETS arbiter, the highest UP is served first until it offers no traffic, and so forth down to the lowest UP of the TC. This is the default operating mode.

2. **Round Robin (RR) mode** - This mode is selected by setting UPINTC_MODE field to 1b for the corresponding TC index in PRTDCB_RETSTCC register array. UPs are served in a packet-based round-robin manner between them within the total bandwidth allocated to the TC. Whenever the TC is selected by ETS arbiter, a different UP which offers workload is served among the UPs mapped to the TC, and so forth cyclically.

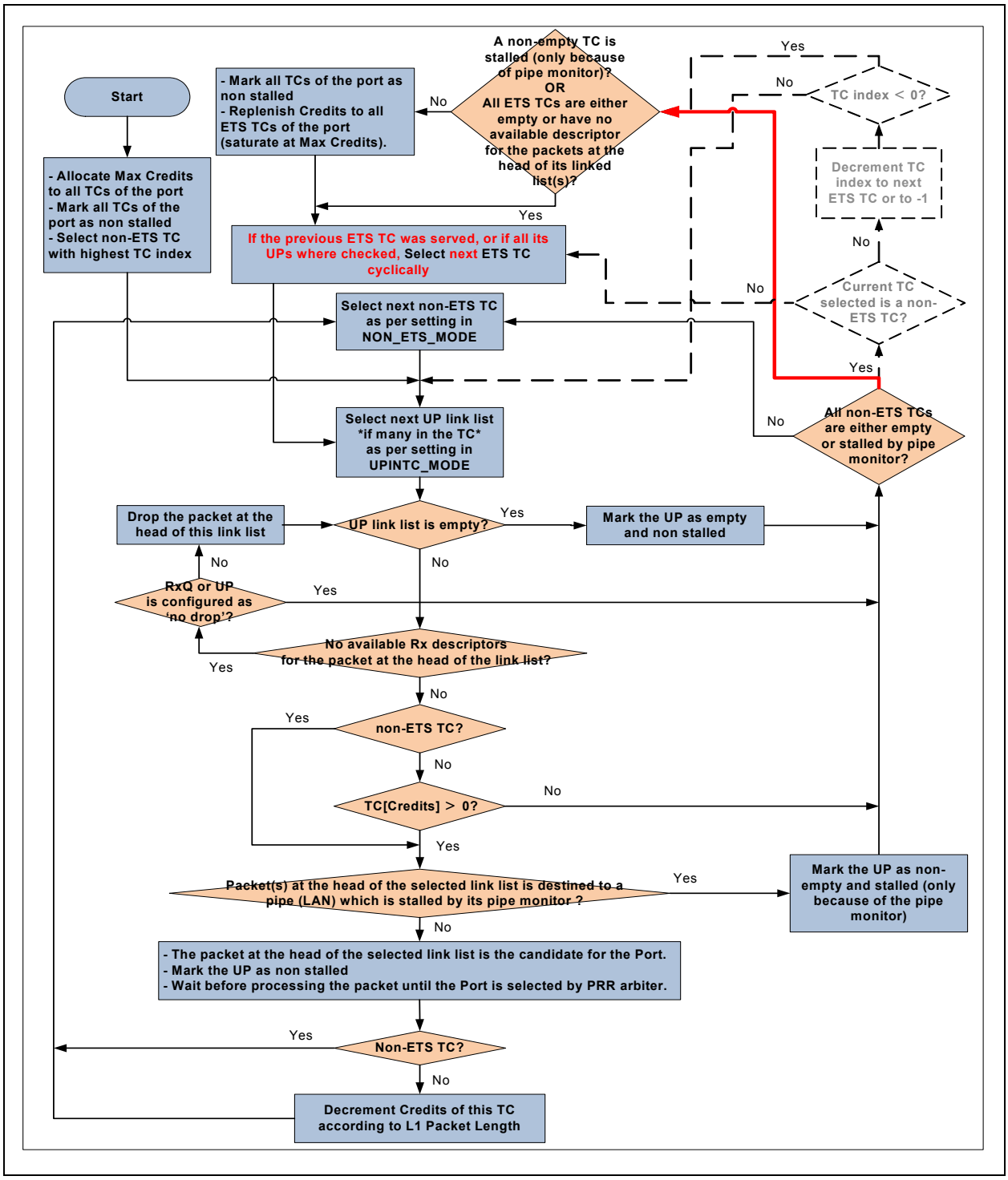


Figure 7-56. Receive ETS Arbiter

According to the total number of enabled ports and to the number of TCs of the port, these fields are configured by the DCBX agent as follow:

- Quad port configuration
  - Ports with 1 to 4 TCs:
    - PRT_SWR_PM_THR.THRESHOLD = 0x9
    - PRTDCB_RPPMC.RX_FIFO_SIZE = 0x8
  - Ports with 5 to 8 TCs:
    - PRT_SWR_PM_THR.THRESHOLD = 0x6
    - PRTDCB_RPPMC.RX_FIFO_SIZE = 0x4
- Dual port configuration
  - Ports with 1 to 4 TCs:
    - PRT_SWR_PM_THR.THRESHOLD = 0xF
    - PRTDCB_RPPMC.RX_FIFO_SIZE = 0x10
  - Ports with 5 to 8 TCs:
    - PRT_SWR_PM_THR.THRESHOLD = 0xC
    - PRTDCB_RPPMC.RX_FIFO_SIZE = 0x8
- Single port configuration
  - For all ports:
    - PRT_SWR_PM_THR.THRESHOLD = 0xF
    - PRTDCB_RPPMC.RX_FIFO_SIZE = 0x10

Depending on the locality of Rx traffic, it may that in quad port configuration, ports with 5 to 8 TCs will not achieve the Rx performance goals.

1. Packets from either pipe (LAN) that belong to a LL TC are always strictly prioritized whenever they compete with Bulk packets on the other pipe.
2. Whenever the competing LAN head of pipe packets belong to the same type, either both LL or both Bulk, they are serviced in a 50%/50% round-robin ratio relatively to the amount of bytes already serviced per LAN type.

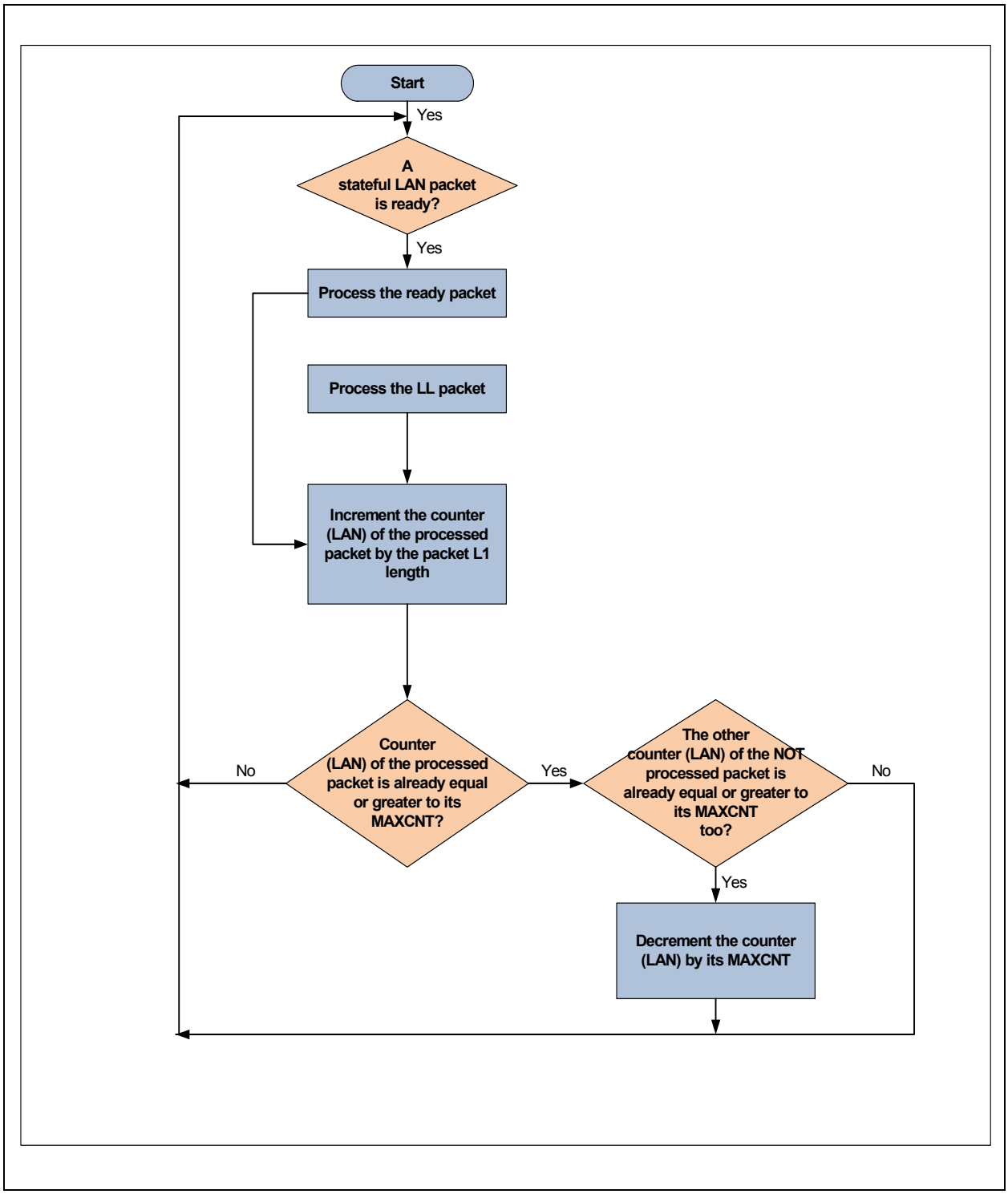


Figure 7-57. LAN counter

7.7.1.1.5 Rx ETS Configuration Rules

PRTDCB_RPRRC are dynamics registers which are auto-programmed by the device. The device retrieves different default values from NVM, one for each the link speed. Whenever a port changes its link speed, the corresponding default value is loaded by hardware.

PRTDCB_RUP2TC, PRTDCB_GENC,PRTDCB_RPPMC.RX_FIFO_SIZE, and PRT_SWR_PM_THR are also dynamic registers. Refer to the flow described in Section 7.7.3.1.1 for the way they are modified.

All other Rx-ETS settings described in Section 7.7.1.1 are static. They shall not be modified at run time. They are loaded from NVM only at initialization/reset time.

7.7.1.2 Receive Path Priority Flow Control (PFC) and Rx Packet Buffer (RPB)

This section deals with guaranteeing no packet loss inside the XL710 for the PFC-enabled traffic, whether it is received from the LAN wires or from the internal switch. Since this feature is tightly related to the allocation of receive packet buffer to the traffic classes, the partitioning scheme of the Rx packet buffer will be described first.

RPB has a max byte capacity of 968 KB and a max packet capacity of 7744 packets (i.e. 968 KB / 128B).

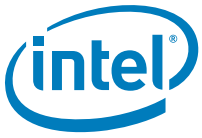
TCs are classified in two types relatively to PFC: no-drop (a.k.a. loss less) TCs, and drop TCs (a.k.a. best effort delivery). PRTDCB_TC2PFC bitmap (and PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE for 40G links) controls the use of PFC by a TC over a link. The setting applies for both Tx and Rx directions (while for 40G links PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE controls Tx direction).

Refer to Section 3.2.1.5 for the basic operation of Ethernet Flow Control, which is relevant whether or not the device is operated in DCB mode.

7.7.1.2.1 Requirements on Rx Packet Buffer

DCB requires that separate queuing resources be allocated to each traffic class along the whole path from the source to the destination node. This will guarantee the good functioning of the ETS scheme, and mainly when it comes to prioritize low latency traffic classes over others.

- Handle one linked list of packets per each UP, per port. The list maintains arriving order of packets inside a flow. UPs are attached to TCs in Rx according to settings made in PRTDCB_RUP2TC register.
- No crosstalk be sensed between the LAN ports, at least on regular basis and if the average incoming packet size is greater or equal to 128B
- Be capable to allocate fully independent buffers for up to five ‘basic’ TCs per port, as follow:
  - 2 no-drop TCs, one only for FCoE traffic, and one for any other traffic type that requires PFC-enabled and 9.5 KB Jumbo frames (e.g. iSCSI traffic).
  - 3 best effort delivery TCs (a.k.a. drop TCs) for which PFC is disabled and which carry 9.5 KB Jumbo frames.
- Be capable to allocate fully independent buffers for up to 8 best effort TCs per port
- 3 ‘extra’ TCs beyond the 5 ‘basic’ TCs described above, or any other TC settings up to 8 TCs per port are supported. It can however be handled via some level of buffer sharing between them.
- Partitioning of Rx packet buffer across the enabled LAN ports is set equally at init time, according to NVM settings. Even in MFP mode, no redistribution of Rx-PB occurs further to a link state change or



to a link speed change, i.e. all ports are assumed to be operated at 10G. This will avoid crosstalk between ports.

- Partitioning of the amount of Rx packet buffers allocated to a port is done dynamically according to the following parameters:
  - Number of TCs supported by the port, as per DCBX
  - Number of PFC-enabled TCs over the port, as per DCBX
  - MFS over the TC, as per DCBX for at least the FCoE TC
- Support LPI, and especially the worst case Tx LPI exit time of 17.38 us while Rx is not in LPI state.

7.7.1.2.2 Normal Partitioning of Rx Packet Buffer

- Eight dedicated pools per port, one per TC. The size of each dedicated buffer is set by PRTRPB_DPS array of registers. Setting a null size to a dedicated packet buffer is allowed (e.g. for non used TCs or for the 'extra' TCs), it is equivalent to say that no dedicated buffer is allocated to the TC.
  - Normal partitioning assumes at least the 5 'basic' TCs per port will get dedicated buffers.
  - Eight dedicated filling counters per port, one per TC.
  - Eight high watermarks per port, one per TC. It is set by PRTRPB_DHW array of registers.
  - Eight low watermark per port, one per TC. It is set by PRTRPB_DLW array of registers. At any time, it shall be set to a lower value than the corresponding high watermark. Setting a null low watermark is allowed.
- One shared pool per port. The size of the shared buffer is set by PRTRPB_SPS register. Setting a null size to the shared buffer of a port is allowed, it is equivalent to say that no shared buffer is allocated to the port. This may be the case when only the 5 'basic' TCs are supported on a port.
  - One shared pool filling counter per port.
  - One high watermark per shared buffer. It is set by PRTRPB_SHW register.
  - One low watermark per shared buffer. It is set by PRTRPB_SLW register. At any time, it shall be set to a lower value than the corresponding high watermark. Setting a null low watermark is allowed.
  - Eight occupancy counters per port, one per TC, to track the amount of bytes of the TC stored on the account of the shared buffer of the port.
  - Eight high thresholds per port, one per TC. It is set by PRTRPB_SHT register. The threshold is relative to the corresponding occupancy counter.
  - Eight low thresholds per port, one per TC. It is set by PRTRPB_SLT register. At any time, it shall be set to a lower value than the corresponding high threshold. Setting a null low threshold is allowed. The threshold is relative to the corresponding occupancy counter.
- One packet buffer is allocated per port. Setting a null size to the port packet buffer is allowed (e.g. for disabled ports), it is equivalent to say that no packet buffer is allocated to the port.
  - Normal partitioning assumes the sum of the four per port buffer sizes equals to the total size of the Rx packet buffer in the XL710, i.e. 968 KB.
  - The size of the per port packet buffer is computed internally as the shared pool size plus the sum over all the dedicated pools size of the port.
  - One per port filling counter.
- One global filling counter for the whole XL710's receive packet buffer.
  - One global high watermark for the whole device, relative to the global filling counter. It is set by GLRPB_GHW register.
  - One global low watermark for the whole device, relative to the global filling counter. It is set by GLRPB_GLW register. Normal partitioning assumes it is set to 0.

Note: The receive packet buffer accounting granularity is 16B, a physical memory line. To all buffer sizes, threshold, and watermarks registers defined in bytes, the device will discard the 4 least significant bits written by software.

The maximum capacity in bytes of the receive packet buffer is 968 KB, provided that the average size of the packets stored is equal or greater than 128B.

- One global packet counter for the whole XL710's receive packet buffer.
  - — One packet high watermark for the whole device, relative to the global packet counter. It is set by GLRPB_PHW register. It is used to reflect the implementation limit, and shall normally be set equal or lower than full Rx packet buffer size (i.e 968 KB) divided by 128B.
  - — One packet low watermark for the whole device, relative to the global packet counter. It is set by GLRPB_PLW register.

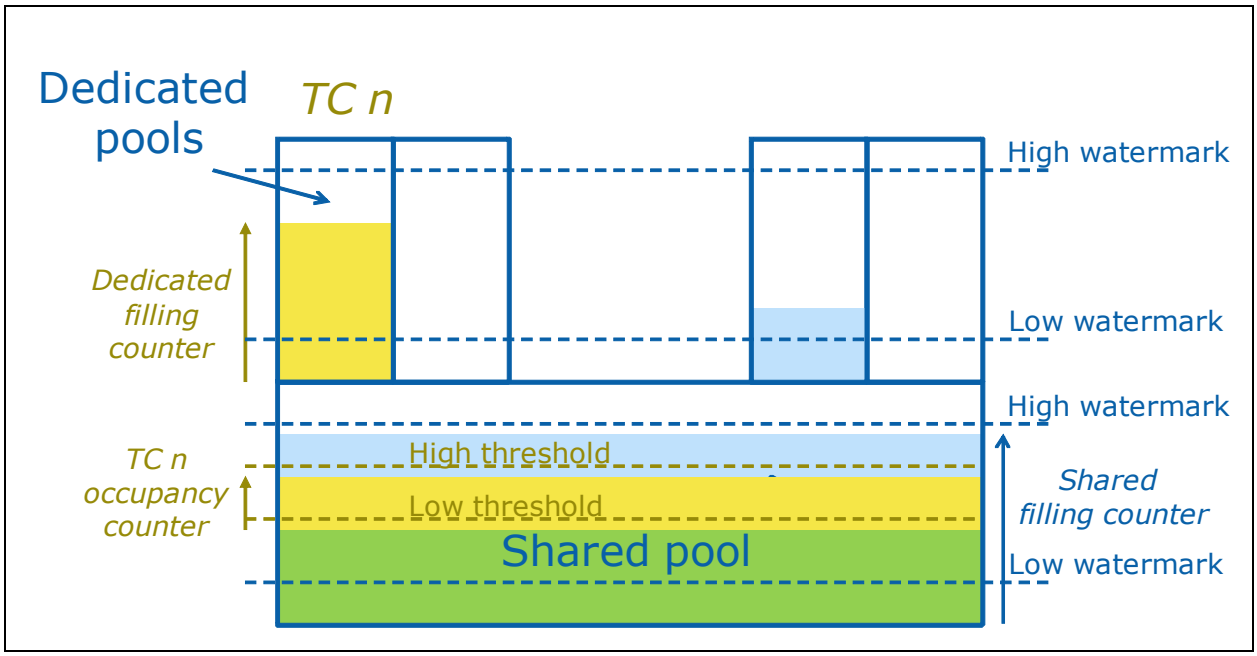
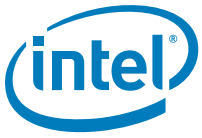


Figure 7-58. Normal Partitioning of the Per Port Rx Packet Buffer

7.7.1.2.3 Receive Drop Policy

Whenever receiving a packet attached to a drop TC, either from the LAN wire or from the internal loopback path, the packet is dropped if ANY of the following condition would occur when storing it into the Rx packet buffer:

a. The dedicated pool (if anyPRTRPB_DFC) is filled above or equal to its high watermark (PRTRPB_DHW), and the TC occupancy counter has reached its high threshold .
b. The dedicated pool (if anyPRTRPB_DFC) is filled above or equal to its high watermark (PRTRPB_DHW), and the shared pool filling counter has reached its high watermark and has not returned to the low watermark yet.
c. The dedicated pool (if anyPRTRPB_DFC) is filled above or equal to its high watermark , and the global filling counter has passed above its global high watermark (GLRPB_GHW) and has not returned to the global low watermark yet .
d. The global packet counter has passed above its packet high watermark .



Note: The hysteresis between high and low watermarks is required to avoid starving the no-drop TCs which can use only the shared pool account with drop TC traffic.

Whenever receiving a packet - even for a no-drop TC, either from the LAN wire or from the internal loopback path, the packet is dropped if there would be no place for storing it entirely in the global Rx packet buffer.

7.7.1.2.4 Policy for Issuing XOFF

Whenever receiving a packet attached to a no-drop TC, either from the LAN wire or from the internal loopback path, a PFC XOFF notification is issued onto the wire and internally to the loopback path for all the UPs mapped to the TC - if ANY if the following condition occurs:

a. The dedicated pool (if anyPRTRPB_DFC) is filled above or equal to its high watermark (PRTRPB_DHW).
b. No dedicated pool for the TC (PRTRPB_DPS=0), and the TC occupancy counter (which is relative the shared pool only) has reached its high threshold (PRTRPB_SHT).
c. No dedicated pool for the TC (PRTRPB_DPS=0), and shared pool filling counter is above its high watermark (PRTRPB_SHW). In this case XOFF will concern all the no-drop TCs of the port that do not have a dedicated pool.
d. Global filling counter has reached its global high watermark (GLRPB_GHW). In this case XOFF will concern all the no-drop TCs of all ports.
e. The global packet counter has passed above its packet high watermark (GLRPB_PHW). In this case XOFF will concern all the no-drop TCs of all ports.

Next time such a condition occurs, no XOFF will be issued until the congested condition disappeared (i.e. XON sent) or until the XOFF timer expired.

7.7.1.2.5 Accounting Packet Reception

Whenever a packet is stored into the receive packet buffer (i.e the packet has not been dropped), the global filling counter is increased by the number of bytes required to store the packet. Also increment the FIRST counter for which the condition is satisfied when checking it in the following order:

1. Shared pool filling counter - Unless one of the following conditions is met:
   a. No shared pool for the port (i.e. null shared pool size, PRTRPB_SPS=0)
   b. Shared pool filling counter has reached its high watermark (PRTRPB_SHW) and there is a dedicated pool for the TC (PRTRPB_DPS>0).
   c. TC occupancy counter (which is relative to the shared pool only) has reached its high threshold (PRTRPB_SHT) and there is a dedicated pool for the TC (PRTRPB_DPS>0).
2. Dedicated pool filling counter – Unless there is no dedicated pool for the TC (PRTRPB_DPS>0).

7.7.1.2.6 Accounting the Servicing of a Receive Packet

Whenever a packet is fetched from the receive packet buffer (e.g., the packet is sent to the host), the global filling counter is decremented by the number of bytes used to store the packet in the receive packet buffer. Also decrement the counter(s) for which the condition is satisfied when checking them in the following order - until the amount of served bytes has been reached:

1. Dedicated pool filling counter – If both conditions are fulfilled:
   a. There is a (non-null) dedicated buffer attached to the TC (PRTRPB_DPS>0).
   b. The dedicated pool filling counter has still not reached zero (non-empty)

Note that it may that the shared pool is no longer full. However, there is no hurry to send XON for a TC that has still not been served internally by Rx-ETS since it gets into XOFF state. This approach may encounter some tolerable fairness issues across the TCs.

2. Shared pool filling counter – When no or empty dedicated pool and as long as shared pool filling has not reached zero

Note: When a port is disabled or disconnected, its Rx packet buffer is drained in such a way that all its associated filling counters get down to zero within a short and bounded time.

When a packet is served, it may that 2 x 48B = 96B are not decremented from the dedicated pool filling counter and from the shared pool occupancy counter. This will happen if the packet is not aligned with the 64B memory blocks and if packets in the TC are not served as per their insertion order.

7.7.1.2.7 Policy for Issuing XON

Whenever servicing a packet attached to a no-drop TC (i.e the packet is sent to the host), a PFC XON notification is issued onto the wire and internally to the loopback path if ALL the below conditions are satisfied:

1. XOFF was issued for this UP
2. TC occupancy counter is below or equal to its low threshold.
3. There is no pool (shared or dedicated) for any port which is currently in an overflow situation (i.e. filling counter above the pool size)
4. The global packet counter is below or equal its packet low watermark (GLRPB_PLW).
5. AND the relevant condition is met:
   a. If there is a dedicated pool for the TC and it has a non-null low watermark (PRTRPB_DLW>0): when the dedicated pool filling is below or equal its low watermark.
   b. If there is a dedicated pool for the TC (PRTRPB_DPS>0) and it has a null low watermark (PRTRPB_DLW=0): when the shared pool filling is below or equal its low watermark.
   c. If there is no dedicated pool for the TC : when the shared pool filling is below or equal its low watermark.
   d. When the global filling counter is below or equal its global low watermark.

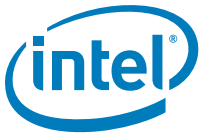
7.7.1.2.8 Prevention of PFC Crosstalk between PCIe Functions

Though Rx queues are allocated per TC in RSS mode; in the XL710, no TC index is stored in the Rx queue context. It is recommended to avoid mixing drop and no-drop traffic over the same Rx queue, as the no-drop traffic might cause head of line blocking of the drop traffic.

There is a need to prevent a malfunctioning function, a paused function, or a non-trusted VM from blocking an entire PFC-enabled TC if it does not provide Rx descriptors. For this purpose, a per UP timer is started each time a packet located at the head of the per UP linked list is waiting for software to free Rx descriptors in the Rx queue to which the packet is destined. A timer starts counting on if the UP is associated with a TC of type "No Drop" (if the TC is of type "Drop", head-of-line packets are dropped when descriptors are not available).

The timer is reset when Rx descriptors are freed.

The Ports/UPs for which the timer timed out can be read via GLDCB_RUPTI bitmap register. Writing 1b to a bit in the bitmap will restart the corresponding timer. The Rx queue that blocked the UP is reported to the PRTDCB_RUPTQ array of registers. 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 and Rx queue registers.



When a timer expires, the EMP is interrupted and takes the following steps:

1. EMP identifies the port and UP of the offending queue by reading the GLDCB_RUPTI register
2. EMP reads the PRTDCB_RUPTQ register to identify the offending queue
3. EMP clears the timer by writing a 1b to the corresponding bit in GLDCB_RUPTI
4. EMP reads the QTX_CTL[Q] register, where Q is the index of the offending RX queue. QTX_CTL identifies the following:

Note: This flow is based on the assumption that software programs the QTX_CTL register for any matched transmit queues of all enabled receive queues. If this rule is not addressed by software, the reported LAN Queue Overflow Event admin command is sent to PF0.

   a. Whether the queue belongs to a PF or VF (the PFVF_Q field).
      - A VM queue is considered to be a PF queue for this section
   b. The PF that owns the queue (or the parent PF in case the queue belongs to a VF) (the PF_INDX field)
   c. The VF that owns the queue (for the case that the queue belongs to a VF) (the VFVM_INDX field)
5. If the queue belongs to a PF
   a. If the NVM bit "PF reset on queue overflow" is set
      - The EMP issues a PFR to the function. Note that the PFR also resets the PF's VFs
   b. If the NVM bit "PF reset on queue overflow" is cleared
      - The EMP sends a "LAN Queue Overflow" AQ event to the PF, notifying the event
6. The EMP indicates to the BMC that the PF driver is not presentIf the queue belongs to a VF
   a. The EMP sends a "LAN Queue Overflow" AQ event to the respective PF that owns the VF, notifying the event
   b. The PF issues a VFR to the function

Note: It is advisable to assign manageability traffic to UPs associated with "Drop" TCs, therefore avoiding the risk of stalling due to host queue that overflows.

7.7.1.2.9 Rx Packet Buffer Configuration Flow

All RPB registers are dynamic registers. They can be loaded from NVM at initialization/reset time, and/or they can be modified at run time further to a change in DCBX resolution. Re-configuring RPB while working may lead to the issuing of XOFF/XON notifications to several TCs at once.

Whenever a change has been made to one of the input parameters listed in Section 7.7.1.2.10, the RPB settings have to be updated.

The modified Rx-PB setting shall be loaded to the device according to the following order:

1. Low thresholds and low watermarks that require to be decreased - registers PRTRPB_SLT, PRTRPB_SLW, PRTRPB_DLW
2. High thresholds and high watermarks that require to be decreased - registers PRTRPB_SHT, PRTRPB_SHW, PRTRPB_DHW
3. Dedicated pools size that require to be decreased
4. Shared pool size - register PRTRPB_SPS
5. Dedicated pools size that require to be increased
6. High thresholds and high watermarks that require to be increased - registers PRTRPB_SHT, PRTRPB_SHW, PRTRPB_DHW
7. Low thresholds and low watermarks that require to be increased - registers PRTRPB_SLT, PRTRPB_SLW, PRTRPB_DLW

7.7.1.2.10 Configuration Rules for RPB Partitioning

The computing flow described in this section is run by the entity responsible to run DCBX (refer to Section 7.7.3). The flow is run for each port independently over the amount of memory allocated to the port, as part of the Rx-PB configuration flow described in Section 7.7.1.2.9.

It makes use of the following parameters as input:

- Number of enabled ports (static parameter after initialization time), used to determine the amount of memory allocated to a port. It is set to 968 KB divided by the number of enabled ports.
- Tx LPI enabled/disabled over the port (as per PRTPM_EEER.TX_LPI_EN register bit). When Tx LPI is enabled over the port, the amount of memory usable for PFC by the port is reduced by [Tx LPI Exit Time - Max MFS over all the TCs of the port]. This reduced effective packet buffer size of the port will be referred as RPB_ESize(Port). For simplicity, worst case Tx LPI Exit Time value is taken regardless of the port type or speed, which corresponds to Tw_phy of 10GBASE-KR Case-2 (see table 78-4 in EEE spec): 14.25 us @ 10G, equivalent to 17.4 KB.
- Number of supported TCs in receive (PRTDCB_GENC.NUMTC)
- Number of PFC-enabled TCs (TC2PFC field in PRTDCB_TC2PFC and for 40G links PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE and PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE)
- MFS per TC table (refer to Section 7.7.3.1.3, it is maintained by the DCBX handling entity, and can be also loaded from NVM). MFS over a TC is referred as MFS(TC), while maximum MFS over all TCs of a port is referred as MFS(max).
- PCIe Round Trip Time (a.k.a. PCIRTT). Refer to Section 7.7.3.2.2.2
- PFCLinkDelayAllowance (a.k.a. DV, standing for Delay Value) over the port. Since MFS is a per TC value, there is a different DV value per TC, referred as DV(TC). Refer to Section 7.7.3.2.2.1

The flow makes use of DV(TC) extended by one MFS(TC) because of a 'last minute' packet from the same TC that can be looped back into the RPB. It is noted as **Std DV(TC) = DV(TC) + MFS(TC)**.

Note:

1. **Allocate 'fully independent' buffers to all TCs**; and allocate the remaining buffer space (if any) to the shared pool of the port.
   a. Step 1 dedicated no-drop TC:
      - Dedicated no-drop low watermark = 2 x MFS(TC) + PCIRTT
      - Dedicated no-drop high watermark = Dedicated no-drop low watermark + MAX{MFS(max), 4.5KB}; the last term is referred as MFS'
      - Dedicated no-drop pool size = Dedicated no-drop high watermark + Std DV(TC)
   b. Step 1 dedicated drop TC:
      - Dedicated drop low watermark = 0
      - Dedicated drop high watermark = Dedicated drop pool size
      - Dedicated drop pool size = 2 x MFS(TC) + PCIRTT
   c. Step 1 shared:
      - Shared pool size = RPB_ESize(Port) – Sum of Dedicated pools' sizes
      - Shared high watermark = Shared pool size
      - Shared low watermark = 2 x MFS(max) + PCIRTT
      - Shared low threshold = 2 x MFS(max) + PCIRTT
      - Shared high threshold = Shared pool size
      - Exit condition: If Shared pool size is not negative, then exit the flow

Note: Ports with only a single TC represent an exception to the flow above, where no dedicated pools are defined, but only a shared pool as follow:

- Shared pool size = RPB_ESize(Port)
- Shared low threshold = 2 x MFS(TC) + PCIRTT
- Shared low watermark = 2 x MFS(TC) + PCIRTT

- Shared high threshold = RPB_ESize(Port)
- If PFC is enabled over the unique TC or if transmitting LFC is enabled for the port (i.e. no-drop behavior over the port Rx path), then
    - Shared high watermark = RPB_ESize(Port) - Std DV(TC)
- Otherwise (i.e. drop behavior over the port Rx path), then
    - Shared high watermark = RPB_ESize(Port)

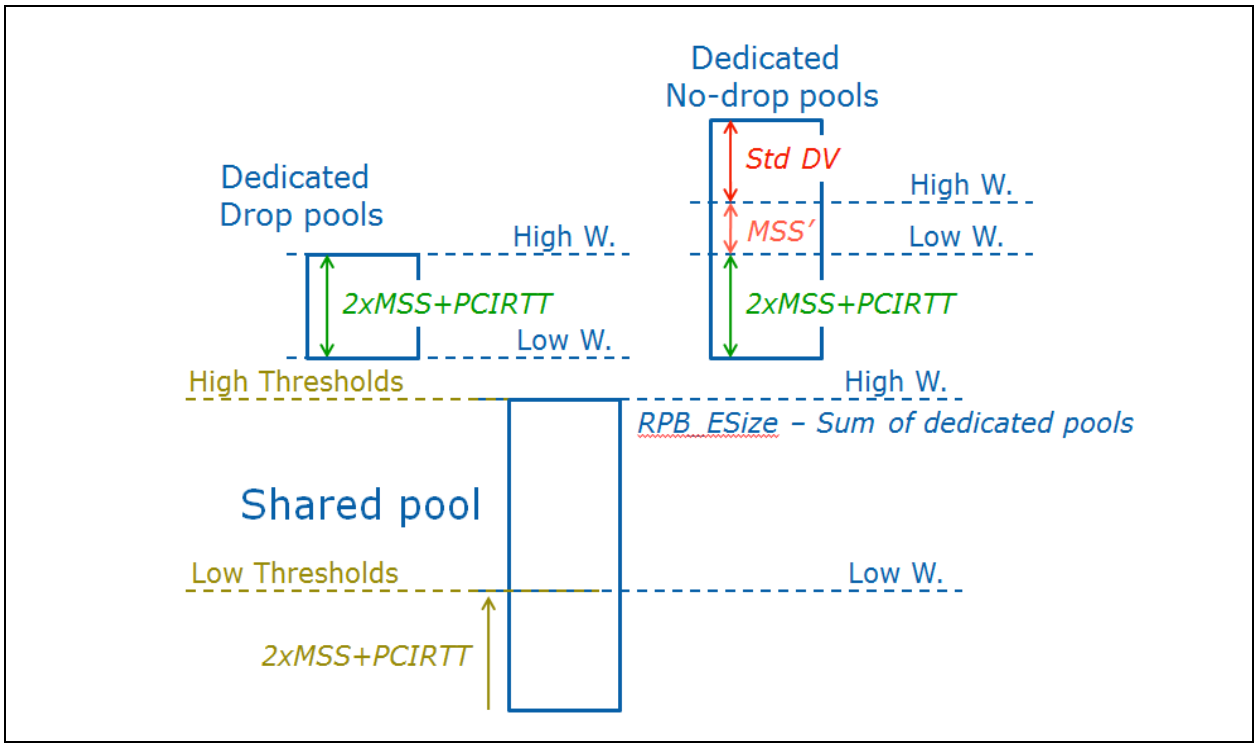


Figure 7-59. Normal Configuration at Step 1

2. **Downgrade TCs to 'semi independent' TCs** - Downgrade a TC to a 'semi independent' PB. Try a drop TC first, but if it is not enough to fulfill the exit condition below, then downgrade a no-drop TC instead, and so forth starting from lowest TCID and up.
    a. Step 2 no-drop TC:
        - Dedicated no-drop low watermark = 64B
        - Dedicated no-drop high watermark = MFS(TC) + 64B
        - Dedicated no-drop pool size = Dedicated no-drop high watermark + Std DV(TC)
        - Shared no-drop low threshold = 2 x MFS(TC) + PCIRTT
        - Shared no-drop high threshold = 2 x MFS(TC) + PCIRTT
    b. Step 2 drop TC:
        - Dedicated drop low watermark = 0
        - Dedicated drop high watermark = MFS(TC)
        - Dedicated drop pool size = Dedicated drop high watermark
        - Shared drop low threshold = 0
        - Shared drop high threshold = MFS(TC) + PCIRTT
    c. Step 2 shared:
        - Shared pool size = RPB_ESize(Port) - Sum of Dedicated pools' size of TCs left unchanged from Step 1 - Sum of Dedicated pools' size of TCs downgraded at Step 2.

- Shared low watermark = 2 x MFS(max) + PCIRTT
- Shared high watermark = Shared pool size
- For TCs left unchanged from Step 1: Shared low threshold = Shared high threshold =0
- Exit condition: If Shared pool size > Sum over n1 of MFS(TC) + Sum over n2 of (MFS(TC)/ 2) + PCIRTT + MFS(max), where n1 = # of Step 2 no-drop TCs, n2 = # of Step 2 drop TCs; then exit the flow with no new iteration of Step 2

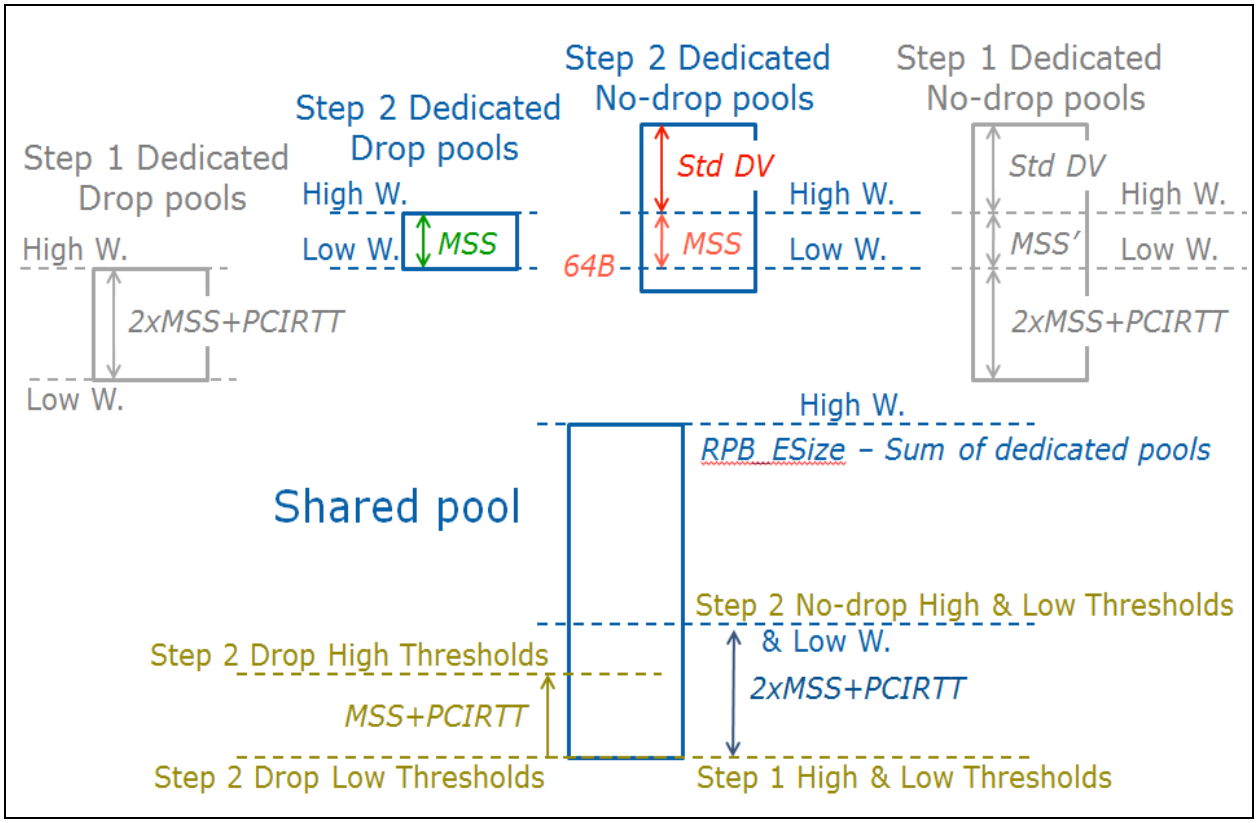


Figure 7-60. Normal Configuration at Step 2

3. **Downgrade TCs to 'shared pool only'** - Downgrade a TC to 'shared pool only', meaning it does not have a dedicated pool. Try a drop TC first, but if it is not enough to fulfill the exit condition below, then downgrade a no-drop TC instead, and so forth starting from lowest TCID and up.
   a. Step 3 no-drop TC:
      - No Dedicated pool
      - Shared no-drop low threshold = 2 x MFS(TC) + PCIRTT
      - Shared no-drop high threshold = Shared no-drop low threshold + MFS(TC)
   b. Step 3 drop TC:
      - No Dedicated pool
      - Shared drop low threshold = 0
      - Shared drop high threshold = 2 x MFS(TC) + PCIRTT
   c. Step 3 shared:
      - Shared pool size = RPB_ESize(Port) - Sum of Dedicated pools' size of TCs left unchanged from Step 2
      - Shared low watermark = 2 x MFS(max) + PCIRTT

- Shared high watermark = Shared pool size – Max Std DV across no-drop TCs downgraded at step 3 (if any)

  Exit conditions:

  If Shared pool size > Sum over all no-drop TCs of MFS(TC) + Sum over all drop TCs of (MFS(TC)/2) + PCIRTT + MFS(max), AND

  If Shared pool size > Max {Std DV + 3 x MFS(TC) + PCIRTT} across no-drop TCs downgraded at step 3 (if any),

then exit the flow with no new iteration of Step 3

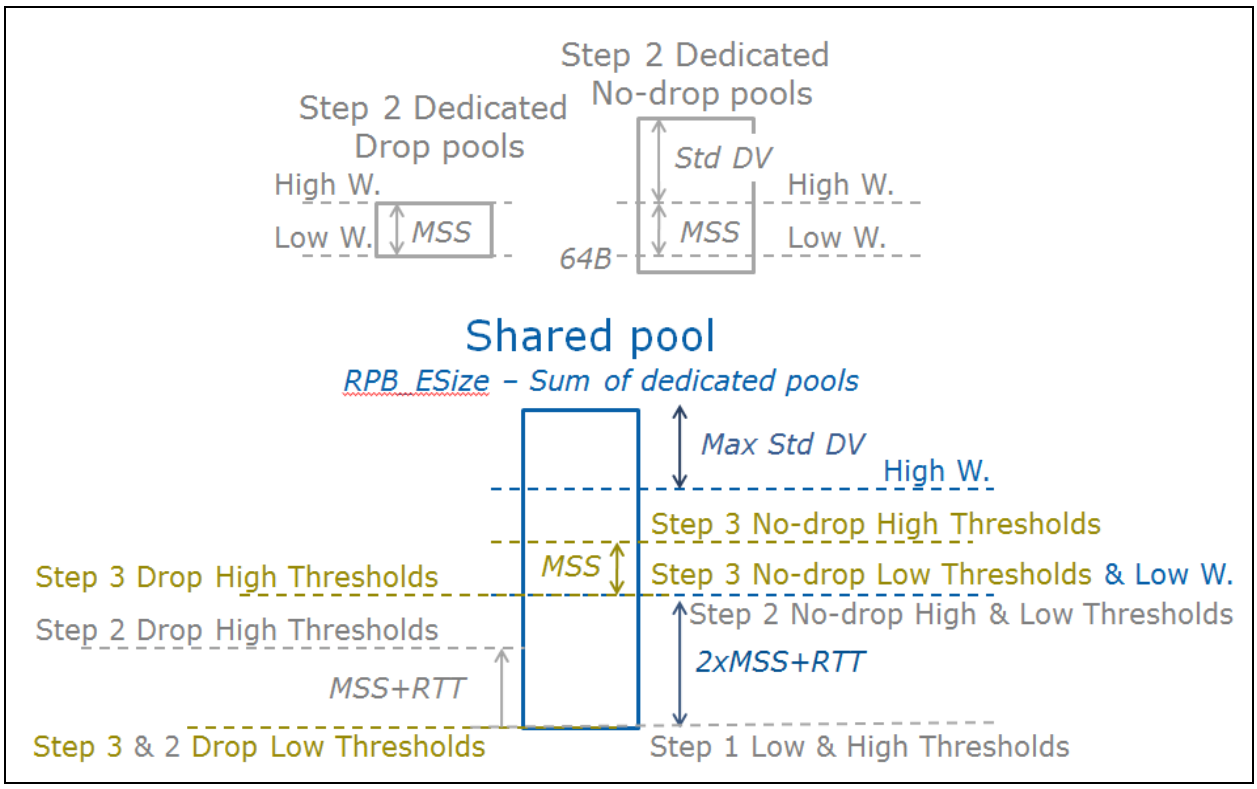


Figure 7-61. Normal Configuration at Step 3

7.7.1.2.11 Configuration Rules for the Global Watermarks

Refer to Section 7.7.1.2.10 for the definition of Std DV(TC).

- The global high watermark (GLRPB_GHW) = [968 KB – Sum over the 4 ports of (Max Std DV(TC) over all no-drop TCs of a port) - n x (Tx LPI Exit Time - Max MFS over all the TCs of the port), where n is the number of ports for which EEE is enabled - 16 KB for 2x LLDP packets per port].
- The global low watermark (GLRPB_GLW) = [968 KB – 2 x (Sum over the 4 ports of (Max Std DV(TC) over all no-drop TCs of a port)) - n x (Tx LPI Exit Time - Max MFS over all the TCs of the port), where n is the number of ports for which EEE is enabled - 16 KB for 2x LLDP packets per port].
- The packet high watermark (GLRPB_PHW) = (968 KB / 128 B) - [(Sum over the 4 ports of (Max Std DV(TC) over all no-drop TCs of a port) / 64 B] - [n x (Tx LPI Exit Time - Max MFS over all the TCs of

the port) / 64 B, where n is the number of ports for which EEE is enabled] - 8 for the 2x LLDP packets per ports.

- The packet low watermark (GLRPB_PLW) = (968 KB / 128 B) - 2 x [(Sum over the 4 ports of (Max Std DV(TC) over all drop TCs of a port) / 64 B] - [n x (Tx LPI Exit Time - Max MFS over all the TCs of the port) / 64 B, where n is the number of ports for which EEE is enabled] - 8 for the 2x LLDP packets per ports.

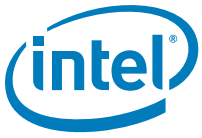
7.7.1.2.12 Examples of RPB Configuration Flows

- EEE enabled over all the 4 ports
- 9.5KB Jumbo used over all the TCs excepted to the FCoE TC
- 10GBASE-T PHY is assumed, which includes the following Interface Delay contributors:
  - XGMII MAC/RS and XAUI interface: 8 192 + 2 * 2 048 = 12 288 bit times
  - 10GBASE-T Delay: 25 600 bit times
- All MFS shall be rounded up to 16B
- All pool sizes or watermarks shall be rounded up to 64B

7.7.1.2.12.1 Example #1: 1 FCoE TC + 1 no-drop TC + 3 drop TCs

10G Analysis	bit time	KB	usec
MSS(FCoE TC)	18048	**2.2**	1.8
MSS(TCs other than FCoE)	77824	**9.5**	
MSS(max)	77824	9.5	7.8
PFC frame	**672**	0.1	0.1
Cable delay	**5556**	0.7	0.6
Interface delay	**37888**	4.6	3.8
Higer layer delay	**6144**	0.8	0.6
Std DV(FCoE TC)	207744	**25.4**	20.8
Std DV(other no-drop TCs)	327296	**40.0**	32.7
PCIRTT (1us=10,000 bit times)	**20000**	2.4	1
Dedicated pool size (FCoE TC) at Step 1	342016	**41.8**	34.2
Dedicated pool size (other no-drop TCs) at Step 1	581120	**70.9**	58.1
Dedicated pool size (drop TCs) at Step 1	176128	**21.5**	17.6
Number of FCoE TCs at Step 1	**1**		
Number of other no-drop TCs at Step 1	**1**		
Number of drop TCs at Step 1	**3**		
Shared pool size at Step 1	466432	**57**	

Figure 7-62. Example #1: 1 FCoE TC + 1 no-drop TC + 3 drop TCs



7.7.1.2.12.2 Example #2: 2 FCoE TC + 2 no-drop TCs + 3 drop TCs

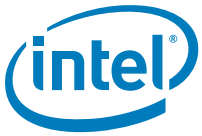
10G Analysis	bit time	KB
MSS(FCoE TC)	18048	**2.2**
MSS(TCs other than FCoE)	77824	**9.5**
MSS(max)	77824	9.5
PFC frame	**672**	0.1
Cable delay	**5556**	0.7
Interface delay	**37888**	4.6
Higer layer delay	**6144**	0.8
Std DV(FCoE TC)	207744	**25.4**
Std DV(other no-drop TCs)	327296	**40.0**
PCIRTT (1us=10,000 bit times)	**20000**	2.4
Dedicated pool size (FCoE TC) at Step 1	342016	**41.8**
Dedicated pool size (other no-drop TCs) at Step 1	581120	**70.9**
Dedicated pool size (drop TCs) at Step 1	176128	**21.5**
Number of FCoE TCs at Step 1	**2**	
Number of other no-drop TCs at Step 1	**2**	
Number of drop TCs at Step 1	**3**	
Shared pool size at Step 1	-457216	**-56**
Dedicated pool size of an FCoE TC dowgraded at Step 2	225792	**27.6**
Dedicated pool size of another no-drop TC downgraded at Step 2	405504	**49.5**
Dedicated pool size of a drop TC downgraded at Step 2	77824	**9.5**
Number of FCoE TCs downgraded at Step 2	**2**	
Number of other no-drop TCs downgraded at Step 2	**2**	
Number of drop TCs downgraded at Step 2	**3**	
Minmum Shared pool size required at Step 2	406528	**49.6**
Shared pool size at Step 2	421683	**51**

Figure 7-63. Example #2: 2 FCoE TC + 2 no-drop TCs + 3 drop TCs

7.7.1.2.12.3 Example #3: 8 no-drop TCs

10G Analysis	bit time	KB
MSS(FCoE TC)	18048	**2.2**
MSS(TCs other than FCoE)	77824	**9.5**
MSS(max)	77824	9.5
PFC frame	**672**	0.1
Cable delay	**5556**	0.7
Interface delay	**37888**	4.6
Higer layer delay	**6144**	0.8
Std DV(FCoE TC)	207744	**25.4**
Std DV(other no-drop TCs)	327296	**40.0**
PCIRTT (1us=10,000 bit times)	**20000**	2.4
Dedicated pool size (FCoE TC) at Step 1	342016	**41.8**
Dedicated pool size (other no-drop TCs) at Step 1	581120	**70.9**
Dedicated pool size (drop TCs) at Step 1	176128	**21.5**
Number of FCoE TCs at Step 1	**0**	
Number of other no-drop TCs at Step 1	**8**	
Number of drop TCs at Step 1	**0**	
Shared pool size at Step 1	-2731520	**-333**
Dedicated pool size of an FCoE TC dowgraded at Step 2	225792	**27.6**
Dedicated pool size of another no-drop TC downgraded at Step 2	405504	**49.5**
Dedicated pool size of a drop TC downgraded at Step 2	77824	**9.5**
Number of FCoE TCs downgraded at Step 2	**0**	
Number of other no-drop TCs downgraded at Step 2	**8**	
Number of drop TCs downgraded at Step 2	**0**	
Minmum Shared pool size required at Step 2	720896	**88.0**
Shared pool size at Step 2	-1326285	**-162**
Number of FCoE TCs downgraded at Step 3	**0**	
Number of other no-drop TCs downgraded at Step 3	**6**	
Number of drop TCs downgraded at Step 3	**0**	
Minimum Shared pool size required at Step 3	720896	**88.0**
Shared pool size at Step 3	1106739	**135**

Figure 7-64. Example #3: 8 no-drop TCs



7.7.1.2.12.4 Example #4: 2 FCoE TC + 4 no-drop TCs + 2 drop TCs

10G Analysis	bit time	KB
MSS(FCoE TC)	18048	**2.2**
MSS(TCs other than FCoE)	77824	**9.5**
MSS(max)	77824	9.5
PFC frame	**672**	0.1
Cable delay	**5556**	0.7
Interface delay	**37888**	4.6
Higer layer delay	**6144**	0.8
Std DV(FCoE TC)	207744	**25.4**
Std DV(other no-drop TCs)	327296	**40.0**
PCIRTT (1us=10,000 bit times)	**20000**	2.4
Dedicated pool size (FCoE TC) at Step 1	342016	**41.8**
Dedicated pool size (other no-drop TCs) at Step 1	581120	**70.9**
Dedicated pool size (drop TCs) at Step 1	176128	**21.5**
Number of FCoE TCs at Step 1	**2**	
Number of other no-drop TCs at Step 1	**4**	
Number of drop TCs at Step 1	**2**	
Shared pool size at Step 1	-1443328	**-176**
Dedicated pool size of an FCoE TC dowgraded at Step 2	225792	**27.6**
Dedicated pool size of another no-drop TC downgraded at Step 2	405504	**49.5**
Dedicated pool size of a drop TC downgraded at Step 2	77824	**9.5**
Number of FCoE TCs downgraded at Step 2	**2**	
Number of other no-drop TCs downgraded at Step 2	**4**	
Number of drop TCs downgraded at Step 2	**2**	
Minmum Shared pool size required at Step 2	523264	**63.9**
Shared pool size at Step 2	-311501	**-38**
Number of FCoE TCs downgraded at Step 3	**2**	
Number of other no-drop TCs downgraded at Step 3	**1**	
Number of drop TCs downgraded at Step 3	**1**	
Minimum Shared pool size required at Step 3	581120	**70.9**
Shared pool size at Step 3	623411	**76**

Figure 7-65. Example #4: 2 FCoE TC + 4 no-drop TCs + 2 drop TCs

7.7.1.2.12.5 Example #6: 40G Link w/ Active/Passive Support with 1 FCoE TC + 1 no-drop TC + 3 drop TCs per port

10G Analysis	bit time	KB	usec
MSS(FCoE TC)	18048	**2.2**	1.8
MSS(TCs other than FCoE)	77824	**9.5**	
MSS(max)	77824	9.5	7.8
PFC frame	**672**	0.1	0.1
Cable delay of 100m with ? = 0.6	**22224**	2.7	2.2
Interface delay	**57344**	7.0	5.7
Higer layer delay fixed to 614.4 ns for all speeds	**24576**	3.0	2.5
Std DV(FCoE TC)	298368	**36.4**	29.8
Std DV(other no-drop TCs)	417920	**51.0**	41.8
PCIRTT (1us=40,000 bit times)	**80000**	9.8	1
Dedicated pool size (FCoE TC) at Step 1	492544	**60.1**	49.3
Dedicated pool size (other no-drop TCs) at Step 1	731648	**89.3**	73.2
Dedicated pool size (drop TCs) at Step 1	236032	**28.8**	23.6
Number of FCoE TCs at Step 1	**1**		
Number of other no-drop TCs at Step 1	**1**		
Number of drop TCs at Step 1	**3**		
Shared pool size at Step 1 - assuming 2x 40G ports	1968128	**240**	

Figure 7-66. Example #6: 40G Link w/ Active/Passive Support with 1 FCoE TC + 1 no-drop TC + 3 drop TCs per port

7.7.2 Transmit Path DCB

7.7.2.1 Transmit Path Enhanced Transmission Selection (ETS)

Transmit path ETS is basically handled in the Tx-scheduler. Refer to Section 7.8.

This section deals with the requirements on Tx data path to avoid distortions on the ETS scheme performed by Tx-scheduler.

The approach relies on two principles:

1. Maintain the order of requests issued by the Tx-scheduler, as most as possible.
2. Avoid misleading the scheduler decisions by inexact reports.

Other ETS requirements on Tx data path:

1. Serve low latency traffic as fast as possible in any condition, as long as it is not overtaking its allocated bandwidth.
2. Guarantee that best effort TCs can be served at full blown, regardless to the XOFF/XON events history over loss less TCs.
3. Achieve the per port throughput performance goals.
4. Avoid starvation of one traffic class by another, even in the case it is dripping packets while the other TCs offer sustained workload.

7.7.2.1.1 Identifying Low Latency Traffic in Transmit

In order 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, Low Latency (LL) and Bulk (B). This approach requires an a priori differentiation between low latency and bulk traffic in order to decide in which buffer a Tx data or request for data has to be stored. TC indexes are used for that purpose according to a setting made in LLTC bitmap in PRTDCB_TETSC_TCB and PRTDCB_TETSC_TPB registers.

The entity that runs DCBX will configure all non-ETS TCs as low latency traffic, and only those TCs.

7.7.2.1.2 User Priority (UP) to Traffic Classes (TC) in Transmit

7.7.2.1.2.1 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 TC is paused/released when a UP bit is set in a PFC XOFF/XON frame received (or in internal notification).

Note: PRT_TCTUPR, PRTDCB_TUP2TC, and PRTDCB_RUP2TC registers shall be programmed identically.

Tx manageability traffic issued by the BMC is bound to the lowest indexed drop TC, or to TC0 if all TCs are no-drop. EMP posts traffic into the Tx path with the TC index specified per packet in the Tx descriptor. It is recommended that EMP use separate internal Tx queues for drop and no-drop traffic to allow keep on sending drop traffic in case some TCs are paused in Tx.

7.7.2.1.2.2 Remapping of the UP field in Transmit

Refer to Section 7.4.6.1.6.1.

The remapping scheme applies also to the traffic issued by EMP (or BMC), which is represented in the tables by its own four VSI numbers.

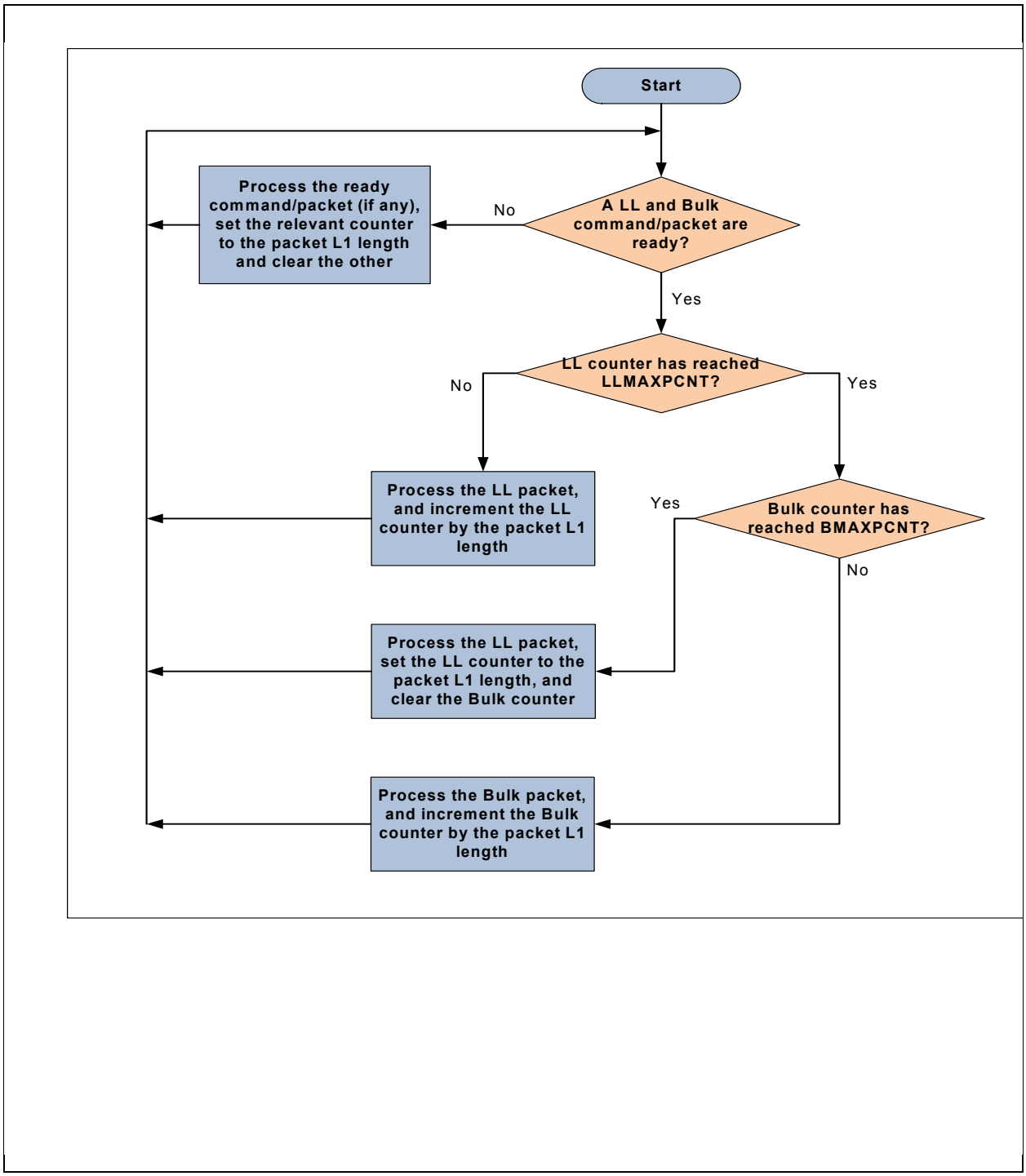


Figure 7-67. Bulk packet processing

7.7.2.2 Transmit Path Priority Flow Control (PFC)

This section deals with guaranteeing no packet loss at the link partner and inside the XL710 further to receiving PFC pause frames from the link partner or further to PFC notifications received from the internal switch.

TCs are classified in two types relatively to PFC: no-drop (a.k.a. loss less) TCs, and drop TCs (a.k.a. best effort delivery). PRTDCB_TC2PFC and PRTDCB_MFLCN.RPFCE bitmaps (and PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE for 40G links) control the use of PFC by a TC over a link.

7.7.2.2.1 Performance Goals for Transmit Path PFC

Basic assumptions:

1. The XL710 shall support optimally up to 2 TCs per port which are PFC-enabled. It may support more, but under sub-optimal performing concerning throughput.
2. One PFC-enable TC supports 9.5 KB jumbo packets, and the other PFC-enabled TC is for FCoE traffic and therefore have its MFS limited to 2.2 KBytes.

XOFF performance goal - Once an internal/external XOFF notification is received on a TC:

1. Transmission from the TC is stopped on the wires within the Delay Value (DV) specified in Annex O of PFC spec. Refer to Section 7.7.1.2.10.
2. Other TCs (PFC-enabled or not) can be served at full blown. Some limitations may be tolerated for other PFC-enabled TCs if 2 PFC-enabled TCs are already paused for the port, depending on the history of PFC XOFF events. Worst case is when the accumulated amount of traffic (commands or data) that was in the data path at the moment XOFF events were received on a port has reached twice the Tx command/data path depth. In this case, all PFC-enabled TCs of the port are paused.
3. PFC XOFF events received on one port shall not impact performance of other PFC-enabled TCs on other ports.

XON performance goal:

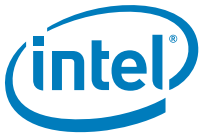
1. **Over the wires** - Once an external XON notification is received for a TC or once the XOFF timer has ended for it, resuming transmission over the wires for this TC shall be made possible within the *Higher Layer Delay* plus the *Interface Delay* specified in Annex O of PFC spec. This corresponds to half of the Delay Value (DV) used for XOFF performance goal above minus the Cable Delay.
2. **On the internal switch** - Once an internal XON notification is received for a TC, resuming transmission at the internal switch ingress port for this TC shall be made possible within only the *Higher Layer Delay* specified in Annex O of PFC spec. Internal switch ingress port is located at the entry of the Rx packet buffer.

Interface Delay shall take in account the layers present and active inside the X710/XL710 and those present in an external PHY connected to it on the board (if any).

7.7.2.2.2 Transmit PFC

The TC(s) concerned by an XOFF/XON notification (which is per UP) are identified by the setting made in PRTDCB_TUP2TC register.

7.7.2.2.3 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 condition is detected autonomously by the device 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 8 Tx PFC timers attached to a port, one per TC (PRTDCB_TPFCTS.PFCTIMER). Each timer (per TC) 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 offset 0x15 of EMP setting module in the NVM), the PF(s) software should take the following actions:

- Post a PFC Ignore admin command (refer to Section 7.7.5.1) for the TC, requesting EMP to set IGNORE_FC bit(s) (32 bits - 1 per TC and per port) in the GLDCB_TFPFCI register. It will cause 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 PRTDCB_TCWSTC and PRTDCB_TCMSTC arrays of registers), the PF will clear the IGNORE_FC bit by posting a PFC Ignore admin command with Ignore Flag cleared.

Note: In case the port is operated in LFC, the functionality is controlled by the bit corresponding to TC0.

The 8 Tx PFC timers of a port shall be always operative, even if a TC has no more UP mapped to it (further to a UP to TC mapping change driven via DCBX).

7.7.2.3 Tx Path DCB Configuration Rules

Tx-Scheduler registers are dynamic registers. They can be loaded from NVM at initialization/reset time, and/or they can be modified at run time further to a change in DCBX resolution. The order and the rules by which the registers have to be written are described in Section 7.8.4.1.

PRTDCB_TUP2TC and PRTDCB_GENC are also dynamic registers. Refer to the flow described in Section 7.7.3.1.1 for the way they are modified.

All other Tx DCB settings described in Section 7.7.2.1 and in Section 7.7.2.2 are static. They shall not be modified at run time. They are loaded from NVM only at initialization/reset time.

7.7.3 Data Center Bridging eXchange Protocol (DCBX)

DCBX protocol relies on the exchange of untagged LLDP packets with the peer over the physical link. It is by default handled by EMP, though when the device is operated in SFP mode, the host can decide to handle it once system boot has been completed. SFP mode software initiates the transition on a per port basis, by posting the Stop LLDP Agent command. By default, EMP handles DCBX for covering the boot time. If the "LLDP Admin Status" word in the NVM "LLDP Configuration" module is cleared, the

LLDP Agent is disabled by default on Port n, and DCBX (as well as any other LLDP-based protocol like EEE) will not be handled by EMP on this port. Similarly, for soft SKUs where DCB is disabled, i.e. GLGEN_STAT.DCBEN bit is cleared, no DCBX or DCB handling is performed by EMP for all ports.

In SFP mode, once software has taken DCBX ownership from EMP, it can give it back to EMP via the 'Start LLDP Agent' AQ command. Also, whenever the host that handles DCBX goes to sleep mode, EMP will not cover for DCBX, and it is the host's responsibility to reset DCB configuration to its default settings (i.e. single traffic class, PFC disabled) prior to entering the sleep state. When software handles DCBX, it is also responsible to configure the DCB settings of the port, via setting the DCB registers. It shall use the relevant flow. In such case, the end-station may be made by SW as the DCBX 'master', i.e. the entity that propagates its DCB settings to the peer.

When LLDP/DCB is handled by EMP, the XL710 behaves always as a DCBX 'slave', retrieving its DCB settings from recommendations or configurations received from the peer.

In any case, the XL710 shall comply with IEEE 802.1Qaz standard for DCBX resolution.

It is assumed that in MFP mode, PFs will not issue untagged LLDP frames in transmit, and in receive the device is configured to capture them to the EMP. If LLDP frames are sent by the PF, they will be tagged with an outer VLAN tag by the device before issuing them on the lines. In receive, LLDP frames with an outer VLAN tag will be directed to appropriate PF.

Refer to Section 7.12 for the general handling of LLDP packets. The LLDP Agent embedded in EMP (including its DCBX agent) is reset by GLOBR. Further to these 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 shall reconfigure the Tx/Rx paths (and any relevant HW which was reset) with the DCB configuration that prevailed before the reset occurred.

DCBX offload made by EMP concerns only the IEEE802.1Qaz DCBX protocol, and the DCB configuration that results is always retrieved from the peer (no EMP support to DCBX 'master' mode).

7.7.3.1 DCBX Offload Flow

DCBX TLVs embedded in the LLDP packets are identified by a TLV type value of 127 followed by the IEEE 802.1 OUI field value of 0x0080C2. There are four types of DCBX TLVs which are handled by the XL710, and they are classified into three categories as follow:

- ETS Configuration TLV and ETS Recommendation TLV
- Priority-based Flow Control Configuration TLV
- Application Priority Configuration TLV

The generic LLDP handling flow is described in Section 7.12, the below steps represent the specific handling required for DCBX:

1. **Wait for first DCBX TLV is received or for a change in received DCBX TLV**.
2. **EMP identifies LLDP/DCBX peer** - The LLDP peer identifier is made by the concatenation of its chassis ID and port ID fields. Compare these fields to a saved copy of the previous LLDP peer identifier received (if there is any and if it is not aged out).
   a. If the LLDP peer identifier is not the same, then store the new LLDP peer identifier into the Remote DCBX parameters saved in firmware data RAM, and start a timer with the longest 802.1AB TTL value of any of the peers. Go to next step even before waiting for timer's end.
   b. If the number of LLDP peers that run DCBX is greater than 1 at the timer ends, then a multiple peer condition is detected and reported via the PRTDCB_GENS.DCBX_STATUS field set to MULTIPLE_PEERS (as well as via a LLDP MIB Change Event if configured for). Restart the timer with the longest 802.1AB TTL value of any of the peers and exit the flow with updating the LLDP DCBX peer and treating DCBX parameters as NULL (i.e. no DCBX peer).

3. **EMP resolves DCBX and reconfigures DCB** - Update the RPB settings according to the flow described in Section 7.7.1.2.9 and to the rules described in Section 7.7.1.2.10 for the normal RPB settings. Update the Local DCB Management Objects of Section 7.7.3.2 according to the flows described in Section 7.7.3.1.1 and Section 7.7.3.1.3.
   a. If LLDP TLVs were aged out, EMP reconfigures the device with the default DCB settings, excepted for LFC which should be reverted to the previously known LFC mode that prevailed before DCBX was resolved.
   b. If some DCBX TLVs are missing or malformed, the concerned DCB settings are returned to their default configuration.

7.7.3.1.1 ETS TLVs Resolution

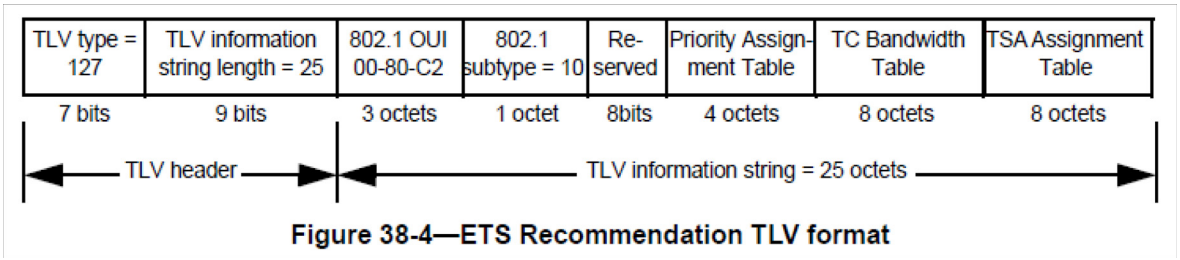


Figure 7-68. ETS Recommendation FLV format

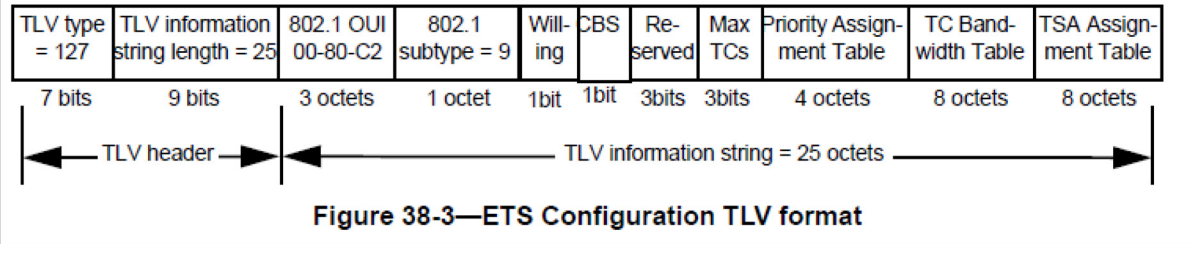


Figure 7-69. ETS Configuration FLV format

The ETS TLV sent by a node concerns its ETS setting in transmit direction. ETS parameters resolution uses the Asymmetric attribute passing state machine described in section 38.4.1 of DCBX standard. It allows a different setting for each Tx direction between two peers. When DCBX is handled by EMP, the XL710 acts as a DCBX 'slave' and therefore the local Willing bit issued to the peer is always set.

If no ETS TLV is received from the peer, or if they are aged out, the default ETS setting is recovered, which assumes all the UPs are mapped to a single non-ETS TC, TC 0.

Otherwise, two scenarios whether or not an ETS Recommendation TLV has been received from the peer:

1. If the ETS Recommendation TLV is present, local ETS settings (both Tx and Rx) are extracted from the ETS Recommendation TLV received. It will be referred as the Remote ETS TLV.
2. If no ETS Recommendation TLV or if it is aged out, local ETS settings (both Tx and Rx) are extracted from the ETS Configuration TLV of the peer. It will be referred as the Remote ETS TLV.

EMP extracts the number of TCs from the ETS TLV. It corresponds to the number of different TC indexes identified in the Priority Assignment Table. Then it performs the following tasks sequentially, according to the case:

1. If a there is a change in a PFC policy or in UP to TC mapping, then go to the port draining flow in Section 7.8.5.6.1.3
   a. Reconfigure DCB settings according to the following order: Load the number of TCs into PRTDCB_GENC.NUMTC.
   b. Configure the Rx commands FIFOs as described in Section 7.7.1.1.4
   c. For all TCs, clear the PRTDCB_RETSTCC .ETSTC bit, regardless to ETS TLVs. Non-ETS TCs are those TC for which the TSA Assignment field value is different than 2. These TCs shall be marked as Low Latency TCs in the LLTC field of PRTDCB_RETSC register.
   d. Update the PFC settings to TCs according to the new Priority Assignment Table (if it was modified), using the flow described in Section 7.7.3.1.2
   e. Load the Priority Assignment Table extracted from the Remote ETS TLV into PRTDCB_RUP2TC
   f. Load the Priority Assignment Table extracted from the Remote ETS TLV into PRT_TCTUPR (referring to Section 7.4.6.1.6.1)
   g. Mark non-ETS TCs as Low Latency TCs in the LLTC field of PRTDCB_TETSC_TCB and PRTDCB_TETSC_TPB registers.
   h. Assign BMC pass-through traffic to the lowest indexed drop TC, or to TC0 if all TCs are no-drop
2. Go to the Tx-scheduler configuration flow in Section 7.8.5.6.1.3, where TC Bandwidth Table and TSA Assignment Table extracted from the Remote ETS TLV will be loaded there into Tx-Scheduler.

Note: When changing the UP to TC mapping and/or PFC policy, it may that PFC and ETS behaviors be perturbed during the transition time until traffic in the pipes are emptied.

7.7.3.1.2 PFC Configuration TLV Resolution

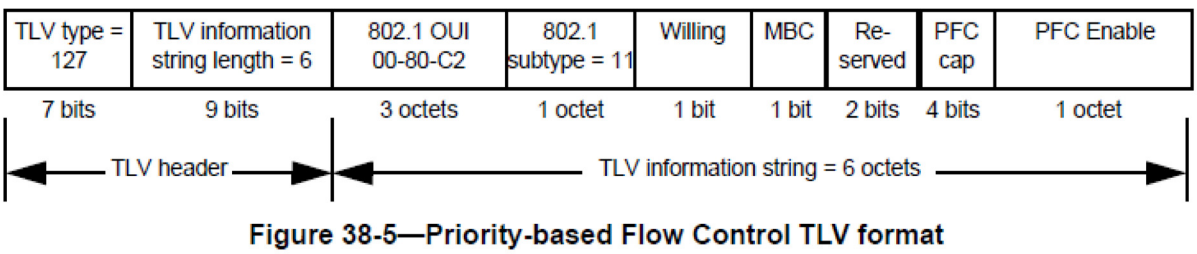


Figure 38-5—Priority-based Flow Control TLV format

Figure 7-70. Priority-based Flow Control TLV format

PFC parameters resolution uses the Symmetric attribute passing state machine described in section 38.4.2 of DCBX standard. It means that the same setting is expected in both directions Tx/Rx, up to the edge of the DCB network.

Two scenarios - assuming local Willing bit is always set:

1. Remote Willing bit is cleared. The local PFC configuration is copied from the peer.

2. Remote Willing bit is set as well. The local PFC configuration is taken from the link partner with the lower numerical MAC address.

In case of scenario 1., and of scenario 2. if the peer has the lower numerical MAC address, the following is performed by EMP:

- Load the PFC Enable bit vector of the peer into TC2PFC field of PRTDCB_TC2PFC and into RPFCE field in PRTDCB_MFLCN (and into PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE and PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE for 40G links), making use of the PRTDCB_RUP2TC.UP2TC settings as follow:
  a. If one of the UPs attached to a TC has its bit set in the PFC Enable bit vector, then set to 1b the TC2PFC bit that corresponds to the TC
  b. If all the UPs attached to a TC have their bit cleared in the PFC Enable bit vector, then clear to 0b the TC2PFC bit that corresponds to the TC
- Set the PRTDCB_RUP.NOVLANUP field with the lowest indexed no-drop UP.

7.7.3.1.3 Application Priority Configuration TLV

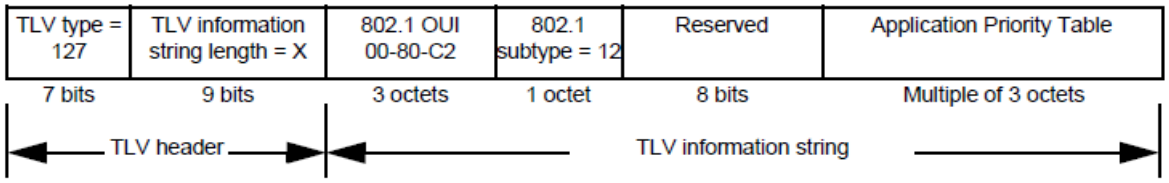


Figure 7-71. Application Priority Table TVL format

Content of this TLV is mainly informational. Two sets are stored by firmware in the DCBX database, one local set and one remote set received from the peer.

It is assumed that the peer always sends the Application TLV and that the local set is copied from the remote set. Otherwise, if no Application TLV has been received from the peer, or if it has been aged out, the default local set contains a unique entry which assign UP value 3 to FCoE traffic.

The local/remote tables stored by firmware are accessible to software via the Get LLDP MIB command (Refer to Section 7.7.3.2.2.1).

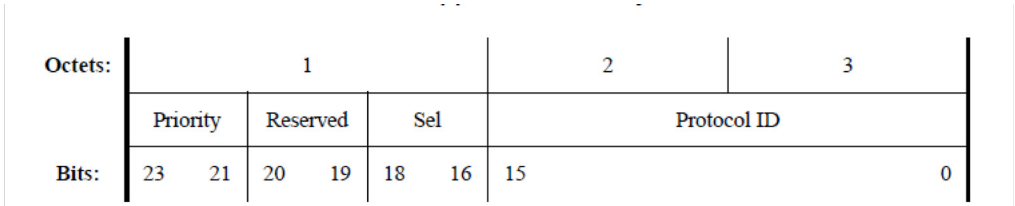


Figure 7-72. Application Priority Table

The UP attached to FCoE traffic is the Priority field in the Application Priority Table row for which

- Sel = 1
- Protocol ID = FCoE Ether Type (0x8906)

This information is stored by EMP into PRTDCB_GENC.FCOEUP, FCOEUP_VALID fields, and it is also used to update an MFS (i.e. Max Frame Size) per TC Table maintained internally by EMP as follow: the MFS of the TC to which the FCoE UP is mapped is set to 2.2 KBytes. For all other TCs, default MFS is 9.5 KB.

Once the Application Priority TLV has been processed, and as long as it is not aged out, EMP sets the PRTDCB_GENS.DCBX_STATUS to DONE. If FCoE protocol was not present in a Protocol ID field of the Application TLV, EMP clears the PRTDCB_GENC.FCOEUP_VALID bit. This is used by BIOS to determine the validity of the FCOEUP field.

7.7.3.2 DCB Managed Objects

7.7.3.2.1 DCBX Managed Objects

When for 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 shall restrain themselves from any write access to the DCB registers listed in Table 7-145. RW in the table refers to the EMP capability to write the object.

Table 7-145. Local DCBX Managed Objects

Object Name	Data Type	Width in bits	Admin	Default	Related Local Configuration Register
ETS Configuration TLV					
Willing	boolean	1	RO	1	Firmware only
Credit-based Shaper (CBS)	boolean	1	RO	0	Firmware only
Max TCs	unsigned integer	3	RW	0[1]	PRTDCB_GENC.NUMTC
Priority Assignment Table	unsigned integer [0..7]	8x4	RW	0	PRTDCB_TUP2TC, PRT_TCTUPR, PRTDCB_RUP2TC
TC Bandwidth Table	unsigned integer [0..7]	8x7	RW	0	See Section 7.8.4.1
TSA Assignment Table	unsigned integer [0..7]	8x8	RW	0[3]	See Section 7.8.4.1
PFC Configuration TLV					
Willing	boolean	1	RO	1	Firmware only
MBC	boolean	1	RO[2]	0	Hardcoded
PFC Cap	unsigned integer	4	RO	0x8[4]	Firmware only
PFC Enable	boolean [0..7]	8x1	RW	0	TC2PFC in PRTDCB_TC2PFC; RPFCE in PRTDCB_MFLCN; PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE; PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE; PRTMAC_HSEC_CTL_RX_ENABLE_GPP; PRTMAC_HSEC_CTL_RX_ENABLE_PPP

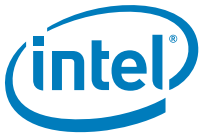


Table 7-145. Local DCBX Managed Objects

Object Name	Data Type	Width in bits	Admin	Default	Related Local Configuration Register
Application Priority Configuration TLV					
Application Priority Table	unsigned integer	32x24	RW	0	Firmware only
MFS per TC Table	unsigned integer	8x15[5]	RW	1536	Firmware only

[1] 0 is the encoding for 8.

[2] MACsec support.

[3] 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.

[4] XL710 is always able to support up to 8 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.

7.7.3.2.2 PFC Managed Objects

The PFC Managed objects are PFC parameters which do not concern DCBX. RW in the table below refers to the Software capability to write the object.

Table 7-146. PFC Managed Objects

Object Name	Data Type	Width in bits	Admin	Default	Spec Reference	Related Local Configuration Register
PFC Control Objects					Table 12.1	
PFCLinkDelayAllowance[2]	unsigned integer	16[1]	RW	0	12.18	PRTDCB_GENC.PFCLDA
PCIRTT[2]	unsigned integer	16	RW	0	N/A	GLDCB_GENC.PCIRTT
PFCRequests	unsigned integer	16	RO	0	12.18	GLPRT_PXOFFTXCNT
PFCIndications	unsigned integer	16	RO	0	12.18	GLPRT_PXOFFRXCNT

1. In PFC spec the parameter is expressed in link bits, but in the XL710 it is expressed in 16 bytes units. SNMP Agent handled in OS is responsible to convert the parameter in bits units when returning it in the corresponding MIB object.
2. These parameters are relevant only when DCBX is handled by EMP. Any modification made by the PF driver to these register fields will take effect on the RPB settings only once a 'DCB Updated' admin command is posted.

7.7.3.2.2.1 PFCLinkDelayAllowance

The value of PFCLinkDelayAllowance is configurable by software per port via the PRTDCB_GENC.PFCLDA register field. It is expressed in 16 Bytes time units.

Firmware uses this parameter to compute the Rx packet buffer settings. Refer to Section 7.7.1.2.10.

Referring to Annex O in PFC Std, PFCLinkDelayAllowance is also referred as the Delay Value (DV), which is computed per TC as follow:

DV= 2*(Max Frame) + (PFC Frame) + 2*(Cable Delay) + 2*(Interface Delay) + (Higher Layer Delay)

- *2*(Max Frame) = MFS (TC) + MFS (max)*. The term is in fact formed by the sum of the MFS (TC) over the TC for which DV is computed and the maximum MFS (max) over all TCs. Refer to Section 7.7.3.1.3 for the MFS per TC table to be used. The PFCLinkDelayAllowance loaded to the PFCDLA register field shall assume MFS is 9.5KB for both. Firmware is responsible to handle a different PFCLinkDelayAllowance per each TC according to the MFS per TC table it handles. It will be referred as DV(TC).
- *(PFC Frame)* duration is equal to 672 bit times
- *2*(Cable Delay)* term is proportional to the cable length and inversely proportional to the bit time duration (i.e. link speed). A 100m Cat6 cable operated at 10G link speed is the worst case for all the supported cable length, medium type, and link speeds across 10G, 1G, and 100M. Under this worst case conditions the term equals to 2*(5,556) bit times. Firmware will not be responsible to optimize the term contribution if the link speed is below 10G. However, it is under FW responsibility to multiply this value internally by 4 for 40G link.
-
- *2* (Interface Delay) + (Higher Layer Delay)* term shall take in account the Interface Delay at each side of the link, and Higher Layer Delay at the peer side. The PFCLinkDelayAllowance value loaded to the PFCLDA register field shall be computed assuming at the peer the worst case values tolerated by the standard over the medium type. Table O-1 in Annex O of PFC Std gives the Interface Delay contributors for the different layers that may be present between the controller and the physical medium.

Sublayer	Maximum RTT (bit times)	Maximum RTT (pause quanta)	Reference (subclause of 802.3)
10G MAC Control, MAC, and RS	8 192	16	46.1.4
XGXS and XAUI	2 048	4	48.5
10GBASE-X PCS	2 048	4	49.2.15
10GBASE-R PCS	3 584	7	50.3.7
LX4 PMD	512	1	53.2
CX4 PMD	512	1	54.3
Serial PMA and PMD	512	1	52.2
10GBASE-T	25 600	50	55.11

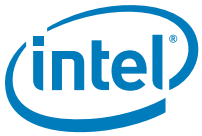
Figure 7-73. IEEE 802.3 Interface Delays

The *(Higher Layer Delay)* at 10G is 614.4 ns, which is equivalent to 6,144 bit times. As before, firmware will not be responsible to optimize the contribution of this term if the link speed is below 10G. Consequently, it is required that the PFCLinkDelayAllowance value loaded by software (or from NVM) to the PFCLDA register field be multiplied by a factor of 4 for a 40G link.

7.7.3.2.2.2 PCIRTT

The value of PCIRTT is configurable by software for the whole device via the GLDCB_GENC.PCIRTT register field. It is expressed in 16 Bytes time units.

Firmware uses this parameter to compute the Rx packet buffer settings. Refer to Section 7.7.1.2.10.



It represents the maximum PCIe round trip time supported with no performance penalty, i.e. 2 us by default. A 10G link speed is the worst case for all the supported link speeds across 10G, 1G, and 100M. At 10G speed the term equals to 20,000 bit times. Firmware will not be responsible to optimize the term contribution if the link speed is below 10G. Consequently, it is required that the PCIRTT value loaded by software (or from NVM) to the PCIRTT register field be multiplied by a factor of 4 for a 40G link.

7.7.4 Initialization of the DCB Functionality

7.7.4.1 DCB Initialization Flow

DCB initialization flow is always handled by EMP, starting from pre-boot time, further to any of the following events: GLOBR, PCIR, PERST, EMPR, and POR.

1. **Auto-load** - The device auto-loads the default configuration of the ports from NVM, which may include some changes to the DCB/RPB registers' defaults, especially for a device operated at 40G. Refer to Table 7-147 table for the default DCB/RPB settings to be made for 40G link speed, since the registers hardware default are tuned to four ports enabled at 10G (or lower) link speed. In any case, DCB/RPB defaults loaded to registers at this stage shall guarantee basic connectivity for the enabled ports.

   It shall equally partition RPB across the enabled ports by setting an identical value into their shared pool buffer (PRTRPB_SPS register) so that their sum equals 968 KB. For disabled ports, zero is expected to be loaded from NVM into PRTRPB_SPS. DCB defaults shall assume one single (non-ETS) TC for the port, i.e. TC0, to which all UPs are mapped. No dedicated pools are used at this stage. PFC is disabled for all UPs. Control port of the internal switch shall by default be addressed to EMP. It guarantees untagged LLDP packets are forwarded to EMP in Rx, and in Tx, untagged LLDP packets (wrongly) issued by the host are filtered out by the device before reaching the wire.
2. **Auto-negotiation** - If auto-negotiation is enabled by default, the device performs auto-negotiation with the peer, which may result in a link speed change and/or in a LFC status change. Any DCB/RPB configuration change required further to auto-negotiation is handled by the device at pre-boot stage, as follow:
   a. *Further to a link speed change* - Update Receive Port Arbiter. The device updates the credits allocated to the port in the Rx Port Arbiter. Other link speed registers settings are handled by EMP as described in Section 3.2.3. FW is responsible to multiply internally the value of PFCLinkDelayAllowance by 4 for 40G link.
   b. *Further to a LFC state change* **-** Update RPB and LFC registers. EMP computes and loads the high/low watermark of the shared pool allocated to the port as if it was a single 'fully independent' buffer. Refer to Section 7.7.1.2.10. Other LFC registers settings are handled by EMP as described in Section 3.2.1.5.
3. **EMP runs DCBX** - Once EMP has (re-)loaded the LLDP filters in the internal switch, and if LLDP Agent was not disabled via NVM settings or via Stop LLDP Agent admin command (the later case is not relevant for POR triggering events), EMP performs DCBX exchange with the link partner from scratch, and updates the device registers related to DCB managed objects as per the flows described in Section 7.7.3.1.
   a. This step is started as soon as the link is up (i.e. auto-negotiation has completed) and any time the link will go up again later on.
4. **BIOS get the UP used for FCoE** - BIOS polls the PRTDCB_GENS.DCBX_STATUS until it is set to DONE, then it can read the PRTDCB_GENC.FCOEUP field to identify on which UP it shall do the FCoE boot load.
   a. In MFP mode, once the link is up, SMASH/CLP runs and determines PF to UP association.
   b. This step is not performed by BIOS if no FCoE boot is needed.

5. **PFs read the DCB configuration** - PFs issue a Get LLDP MIB command to the device. From the response posted by EMP, it identifies the UP used for storage as well as the UP to TC mapping used at the link level.
   a. In MFP mode, a PF can read the UPs it was assigned by BIOS in two steps: First issue the Get Switch Configuration AQ command to get the list of VSIs assigned to it, and then issue the Get VSI Parameters command for each assigned VSI to get the UPs assigned to the VSI.
6. **PFs configure VEBs and VSIs according to the flow described in** Section 7.8.5.6.1.4
   a. If there are several UPs attached to a TC, and if there is at least one Rx Queue for each, EMP may attach one corresponding Tx Queue Set per each UP under the TC. This is done upon availability of the Tx Queue Set resources. In this case, a Tx Queue Set belongs to a single UP and to a single TC. Otherwise, it belongs to a single TC. In any case, an Rx Queue may not carry traffic from a single UP or a single TC, as Rx Filters assign traffic to Rx Queues according to many decision parameters which may not take in account only UP/TC.
   b. EMP set the per VSI UP translation tables used in Tx and Rx, VSI_TUPR and VSI_RUPR tables respectively.

Note: On CORER events, EMP shall reconfigure the core blocks with the DCB settings saved from the last DCBX resolution.

7.7.4.2 Transfer of DCB Ownership between EMP and OS

7.7.4.2.1 Transfer of DCB Ownership to OS

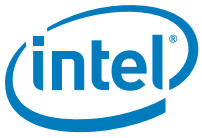
When in SFP mode, it may that the OS decides to take ownership over DCBX from EMP. This is done per port according to the following flow:

1. **OS disables DCBX offload** - OS posts a Stop LLDP Agent command. This notifies EMP to stop handling DCBX.
   a. In case the Shutdown LLDP Agent command variant is used, the EMP LLDP Agent will issue a last LLDP packet to the peer with TTL=0 before returning the port to its default DCB setting (one single non-ETS TC, i.e. TC 0). This variant is used by the PF when it does not plan to take LLDP ownership. It has the advantage to gracefully shutdown LLDP with the peer without requiring the host to get involved in the last LLDP packet transmission.
2. **EMP completes the Stop LLDP Agent command**.
3. **OS redirects LLDP to the host** - OS moves the LLDP forwarding rules of the internal switch to the control VSI.
4. **OS runs DCBX** - OS performs DCBX exchange with the link partner, and updates the device registers related to DCB managed objects as per the flows described in Section 7.7.3.1. It may that OS will (re-)run DCBX in the ‘master’ mode, propagating to the peer the local DCB settings which were made to its DCBX Agent via some DCB Application.
5. **Whenever OS detects a DCBX change**, it posts a DCB Updated event in the admin command queue to notify the EMP that the untagged traffic mapping shall be modified.

Note: If Stop LLDP Agent command is issued when the LLDP agent is already off, the command is silently dropped.

LLDP Agent may be disabled by default via clearing the "LLDP Admin Status" word in the NVM "LLDP Configuration" module. This mode may be useful on nodes that provide NAT and other network edge services.

On any reset event other than POR, EMP will not retake LLDP ownership on ports where ownership has been moved to the PF. It means that the PF shall not perform the transfer of DCB ownership to OS again further to such reset events.



The Stop LLDP Agent is ignored when the device is operated in MFP mode.

7.7.4.2.2 Return of DCB Ownership to EMP

In case of OS reboot, it is possible that the system goes through complete initialization cycle, including a BIOS pre-boot phase. In such a case, it is required that LLDP ownership returns to EMP so that it supports pre-boot activities such as OS boot over network. Pre-boot software will have to first initiate a GLOBR to re-initiate the device hardware, and then to ask EMP to take over the LLDP agent. The later is done by posting a Start LLDP Agent command. Refer to Section 7.12.5.2.3.8.

Further to receiving the Start LLDP Agent, EMP starts the LLDP agent from scratch, making use of the hardware defaults of registers.

7.7.4.3 Initial DCB Settings

The table below describes the changes in DCB registers that shall be made via NVM settings when the port is operated at 40G link speed. The hardware default settings for DCB registers correspond to ports are operated at 10G or lower speeds.

1. The value put in the NVM image shall be optimized to reflect the PHY type and the internal delays in the XL710.

Table 7-147. Changes in DCB registers made via NVM settings when port operated at different link speed

Register	Field	1x 10G	2x10G	4x10G	1x 40G	2x 40G
PRTDCB_GENC	PFCLDA	0x079D[1]	0x079D[1]	0x079D[1]	0x079D	0x079D[1]
GLDCB_GENC	PCIRTT	0x009C	0x009C	0x009C	0x0270	0x0270
PRTDCB_RPPMC	RX_FIFO_SIZE	0x10	0x10	0x08	0x10	0x10
PRT_SWR_PM_THR	THRESHOLD	0x0F	0x0F	0x09	0x0F	0x0F
GLRPB_PHW	PHW	0x1246	0x1246	0x1246	0x16E3[2]	0x16E3
GLRPB_PLW	PLW	0x0846	0x0846	0x0846	0x0FA9[2]	0x0FA9
PRTRPB_SPS	SPS	0x3C800	0x3C800	0x3C800	0xF2000[3]	0x79000

1. The value is taken from the 40G link w/ active/passive support, and is valid also for the single 40G port mode.
2. The value is 0 for the disabled ports
3. The computed value is very close to the 10G case, and therefore the 10G values are taken for simplicity

Table 7-148. Changes in DCB registers made via NVM settings when port operated at different link speed

GLDCB_TGENC_RLPM	TCPM_DIS	0x0	0x0	0x0
	CWLB_MODE	0x0	0x0	0x0
PRTDCB_TCPFCPC_RLPM	PORTOFFTH	0xc64	0x65c	0x34a
PRTDCB_ TCPFCTCC_RLPM	TCOFFTH	0x624	0x320	0x19e

Table 7-148. Changes in DCB registers made via NVM settings when port operated at different link speed

PRTDCB_TCPMC_RLPM	CPM	0x624	0x320	0x19e
	TCPM_MODE	0x0	0x0	0x0
PRTDCB_TCPFCTCC_RLPM	LL_PRI_EN	0x1	0x1	0x1
	LL_PRI_TRESH	0x608	0x304	0x182

7.7.5 DCB Admin Commands

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

7.7.5.1 PFC Ignore

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. EMP shall read-modify-write the GLDCB_TFPFCI register for handling a PFC Ignore request/release command.

Table 7-149. PFC Ignore Response

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.2 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, will be copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value, will be copied by the EMP into the completion of this command.
Completion_flags	16		Bits 7:0 - 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	17-19		Reserved, must be zeroed.
Reserved	20-23		
Reserved	24-27		
Reserved	28-31		

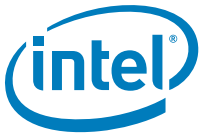


Table 7-150. PFC Ignore Command

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.1 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, will be copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value, will be copied by the EMP into the completion of this command.
Command_flags	16-17		Byte 16, bits 7:0 - 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. Byte 17, bits 6:0 - Reserved, must be zeroed. Byte 17, bit 7 - Ignore Flag: When set to 1b, the PF requests to ignore PFC conditions on the TC indexes set to 1b in the TC Bitmap. When clear to 0b, the PF requests to release any ignore PFC condition request issued for the TC indexes set to 1b in the TC Bitmap.
Reserved	18-19		Reserved, must be zeroed.
Reserved	20-23		
Reserved	24-27		
Reserved	28-31		

7.7.5.2 LLDP/DCBX Admin Commands

Refer to the LLDP Protocol commands described in Section 7.12.5.2.3.

7.7.5.3 DCB Updated Command

When LLDP is handled by the PF, this command is used to notify EMP that a DCB setting has been modified.

When LLDP is handled by EMP, it is used by PF to notify EMP that one of the following parameters has been modified:

— PFCLinkDelayAllowance set by PRTDCB_GENC.PFCLDA

— PCIRTT set by PRTDCB_GENC.PCIRTT

In return, EMP may modify the mapping of untagged traffic (via PRTDCB_RUP). A command completion is posted once the mapping has been changed.

Table 7-151. DCB Updated Response

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.2 for details.
Opcode	2-3	0x0302	Command opcode.
Datalen	4-5		Must be zeroed.
Return value	6-7		Return Value: 0x0 - if needed, RPB or untagged traffic mapping was modified accordingly.
Cookie High	8-11	Cookie	Opaque value, will be copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value, will be copied by the EMP into the completion of this command.
Reserved	16		Reserved
Reserved	17-19		Reserved, must be zeroed.
Reserved	20-23		
Reserved	24-27		
Reserved	28-31		

Table 7-152. DCB Updated Command

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.1 for details.
Opcode	2-3	0x0302	Command opcode.
Datalen	4-5		Must be zeroed by driver.
Return value	6-7		Must be zeroed by driver.
Cookie High	8-11	Cookie	Opaque value, will be copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value, will be copied by the EMP into the completion of this command.
Reserved	16-19		Reserved, must be zeroed.
Reserved	20-23		
Reserved	24-27		
Reserved	28-31		

7.7.5.4 LAN Queue Overflow Event

Refer to Section 7.7.1.2.8 for more details. This event is sent from the device to a PF. It does not generate a response.

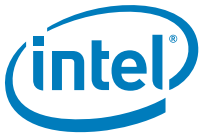


Table 7-153. LAN Queue Overflow Event

Name	Bytes.Bits	Value	Remarks
Flags	0-1	0	See Section 7.10.5.1.1 for details.
Opcode	2-3	0x1001	Command opcode.
Datalen	4-5	0x00	N/A
Return value	6-7	0x00	N/A
Cookie High	8-11	Cookie	Opaque value, will be copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value, will be copied by the EMP into the completion of this command.
PRTDCB_RUPTQ	16-19	See text in Section 7.7.1.2.8	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	See text in Section 7.7.1.2.8	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.

7.8 Transmit Scheduling

7.8.1 Bandwidth Management Hierarchy

7.8.1.1 Hierarchy of Switching Elements

The XL710 contains a number of "switching elements" which may be arranged in a hierarchical manner. Depending on how the XL710 is configured, the hierarchy may consist of zero, one or two layers. Each switching element contains a single uplink (egress) port and one or more virtual (ingress) ports. The egress port of the switching element may be connected to physical port or to the virtual ingress port of another switching element. For a detailed description of the switching components, terminology and hierarchy rules supported by the XL710, see Chapter 7.4.2.1.

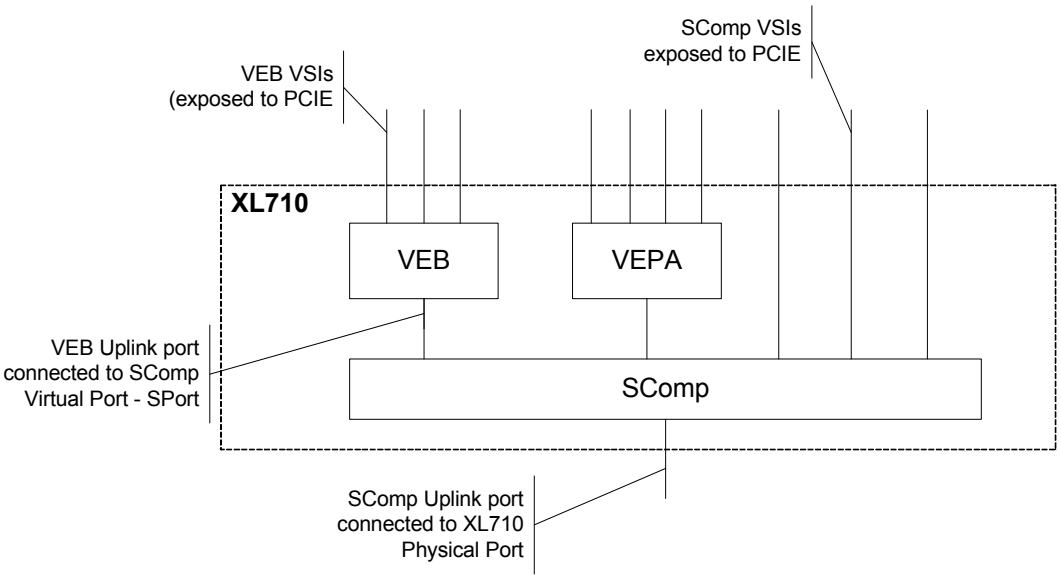


Figure 7-74. Hierarchy of Switching Elements

Figure 7-74 above shows an example of a two level switching hierarchy. The S-comp's ingress port is connected to the physical port. Several of the S-comp's virtual ports (S-channels) are exposed as PCIe functions. Two of the S-comp's egress ports are connected to VEB and VEPA switching elements. Both the VEB and VEPA switching elements expose multiple virtual ports as PCIe functions or VSIs.

A virtual ports which is not connected to the egress of a switching element is exposed as a virtual port to the PCIe interface and called a Virtual Station Interface (VSI). In this chapter, we will use the term virtual port to refer to all ports of a switching component, and we will use the term VSI for virtual ports which are exposed as PCIe functions. Each of the VSIs is assigned to either the physical or virtual function and connected to the virtual egress port of the respective PCIe function. A VSI may be owned by a single PCIe function at any one time, but may transition from one PCIe function to another. A single PCIe function may own multiple VSIs.

By default, a virtual function owns a single VSI. A virtual machine which needs connectivity to multiple virtual ports or switching elements can either use an emulated path through a VMM (using VSIs owned by the physical function) or use multiple virtual functions, one for each VSI.More details on association of various switching elements and VSIs and PCIe functions can be found in Chapter 7.4.2.1.

?

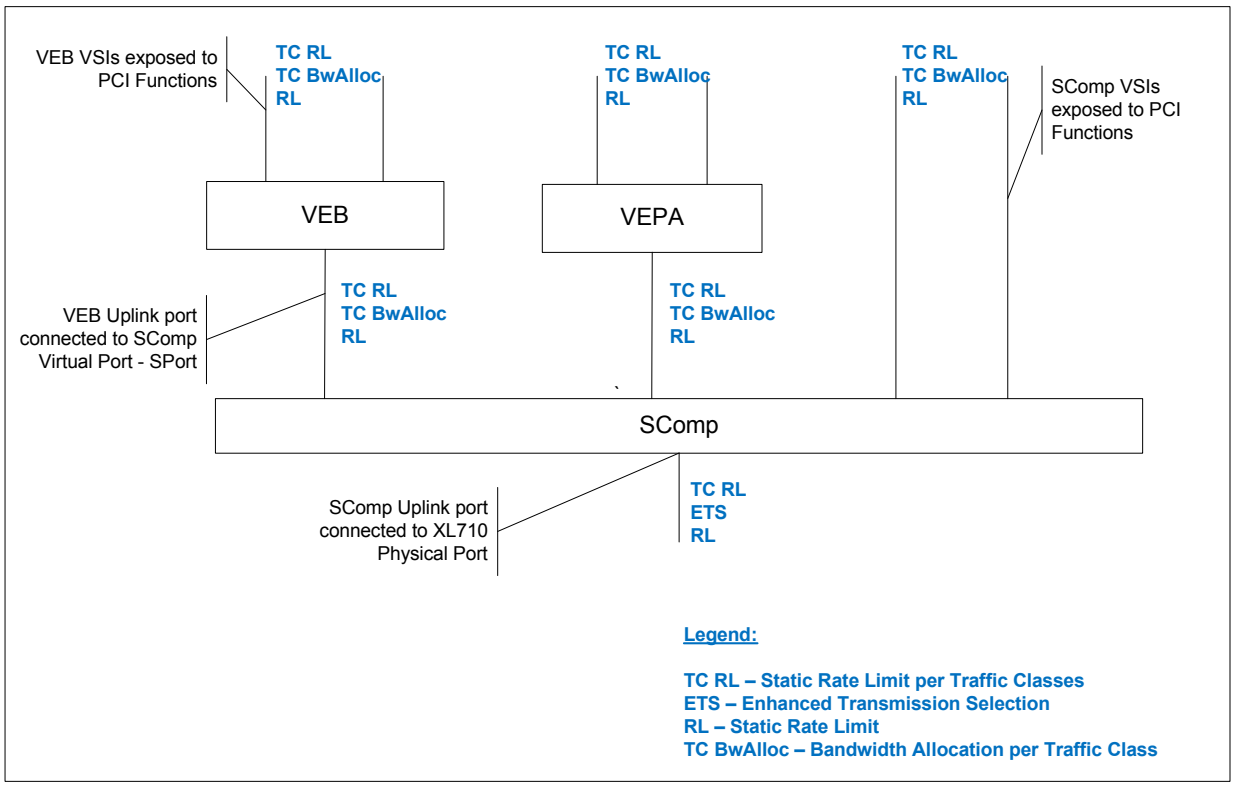


Figure 7-75. XL710 Bandwidth Management Attributes

The XL710 Transmit Scheduler configuration follows topology of the internal switching components. Figure 7-75 shows various bandwidth management attributes and their association with the switching components. A subset of bandwidth management controls shown on this diagram is available depending on the Transmit Scheduler configuration scheme. See Section 7.8.2 for description of supported transmit scheduler configuration schemes.

- VSI/Switching Component bandwidth limit - each VSI and switching component can be configured to limit maximum bandwidth used by that VSI and switching component. Limiting bandwidth of the switching component effectively limits bandwidth all VSIs associated with that switching component. For details, see Section 7.8.1.3. Bandwidth limit of VSIs and Switching Components can be configured in all supported transmit scheduler configuration schemes. See Section 7.8.2.
- ETS Configuration of Physical Port - The XL710 Physical Port can be configured with ETS configuration.This allows to specify distribution of bandwidth among Traffic Classes enabled for the Physical Port. Each Physical Port can be configured with independent ETS. TCs configured to the Physical Port can be enabled for the Switching Components and VSIs allocated for the Port. Allocation, association and bandwidth distribution within each one of the TCs must be consistent with Physical Port ETS configuration and bandwidth distribution within the switching hierarchy. For example: Traffic Classes enabled for VSI must be enabled for respective switching components and

physical ports, and a total bandwidth allocated to all VSIs for particular TC must be equal to the bandwidth allocated for that TC on the switching component. For details on ETS configuration, see Section 7.8.1.4.

- Per Traffic Type Bandwidth Allocation - Each VSI and Switching Component can have relative bandwidth allocated per TC. Such bandwidth allocation is hierarchical, similar to the VSI/Switching Component bandwidth allocation described above. But it is done within particular Traffic Class. I.e. each VSI of the switching component can be configured with bandwidth share with respect to other VSIs of the same switching component for the particular Traffic Class. Such bandwidth allocation is mutually exclusive with the VSI Bandwidth Allocation described above and depends on the transmit scheduler configuration scheme. See Section 7.8.2. The bandwidth allocation principles are described in Section 7.8.1.2.
- Per Traffic Type bandwidth limit - Each VSI that has an ETS enabled can be configured to limit the maximum bandwidth available for one or more TCs configured for that port. For details, see Section 7.8.1.4.

Section 7.8.1.2.1Configuration of the the XL710 scheduler is done by firmware. Software can use the Admin Queue interface to modify a default allocation of bandwidth attributes. See Section 7.8.3.1.

Table 7-154 shows bandwidth management attributes supported for each switching component and VSI depending on the transmit scheduler configuration scheme. For a description of both transmit scheduler configuration schemes, see Section 7.8.2.

Table 7-154. XL710 Bandwidth Management Attributes

Configuration Scheme	Switching Element Name	Bandwidth Attribute	Description
ETS-Based Scheme	SComp	Bandwidth Limit	This applies to any Switching Component or VSI directly attached to the Physical Port Bandwidth limit for entire physical port regardless type of the traffic (TC)
		ETS	Relative bandwidth allocation between TCs within Switching Component
		Traffic Type Bandwidth Limit	Bandwidth limit for individual TC
	VEB/PA	Bandwidth Limit	Bandwidth limit for entire Switching Component regardless type of traffic (TC)
		Traffic Type Bandwidth Allocation	Relative bandwidth allocation of Switching Component within each traffic type (TC)
		Traffic Type Bandwidth Limit	Bandwidth limit for Switching Component per traffic type (TC)
	VSI	Bandwidth Limit	Bandwidth limit for entire VSI regardless type of traffic (TC)
		Traffic Type Bandwidth Allocation	Relative bandwidth allocation of VSI within each traffic type (UP)
		Traffic Type Bandwidth Limit	Bandwidth limit for VSI per traffic type (TC or UP)

7.8.1.2 Bandwidth Allocation

The XL710 supports configurable allocation of bandwidth to the switching elements and their virtual (ingress) ports. Each switching element gets certain portion of bandwidth allocated to its uplink port. The total bandwidth available for the switching component is shared across its ingress virtual ports.

This effectively allows distribution of the egress switch port bandwidth between ingress ports (VSIs in case of virtual port exposed to PCIe functions). Each virtual port is configured with its share of the switching element bandwidth.

The default configuration assumes equal bandwidth distribution between ingress ports. This default configuration can be changed using admin queue commands defined in Chapter 7.8.4. By default, bandwidth distribution between switch virtual ports is relative, and if some of virtual ports did not use their bandwidth allocation, the remaining bandwidth can be used by other virtual ports relatively to their original bandwidth share. For example, if bandwidth allocation of three VSIs is configured to 10%, 30% and 60% respectively; and if VSI_2 (configured to 60%) does not use its bandwidth; then VSI_0 and VSI_1 can share the remaining 60% of wire bandwidth relative to their original bandwidth allocation (which was 10% for VSI_0 and 30% for VSI_1). The new bandwidth distribution would be 25% for VSI_0 and 75% for Port1.

Note that bandwidth ratio between two VSIs remains the same (1:3).In the relative bandwidth allocation mode, all entities that can be scheduled are assigned non-zero bandwidth share, see Figure 7-76 for an example of bandwidth distribution.

Bandwidth allocation is not accumulative. If virtual port did not use its bandwidth share, for any reason (e.g. no work available, bandwidth limit, priority flow control, etc.), it does not accumulate any additional bandwidth and next time it becomes active, it will be allowed to consume its allocated bandwidth share, regardless a period of staying idle and not using bandwidth.

Granularity of the bandwidth allocation per virtual link is 1% of the bandwidth available for that virtual link.The virtual port propagates its bandwidth allocation to the uplink egress port of the switching element, or to the PCIe function owning this virtual port. This bandwidth allocation can be in turn shared by virtual ports/VSIs of the switching element or by transmit queues of respective PCIe function.

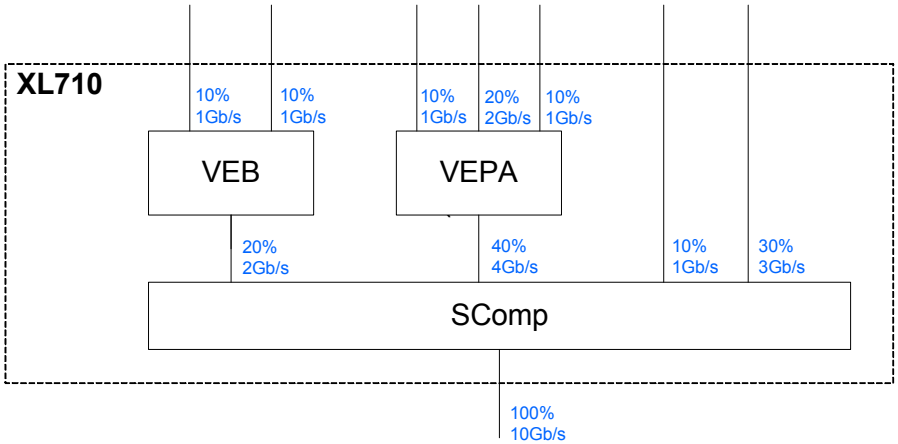


Figure 7-76. Example of Bandwidth Distribution

Diagram above shows an example of bandwidth distribution within switching hierarchy. S-comp uplink port is connected to the 10Gb physical port of the chip, and owns all the bandwidth available on the wire. S-comp has four virtual ports - two VSIs is two S-channels connected to the VEB/VEPA switching components, and bandwidth allocation for each port is as shown on the diagram. Note that bandwidth distribution is relative, and each virtual port gets a percentage of the bandwidth available for the S-comp. For example if the total amount of bandwidth available for S-comp will be decreased to half, then bandwidth available for each virtual port will be effectively halved, or if some virtual ports won't use a

bandwidth allocated for them, other virtual ports can use the remaining bandwidth. VEB switching component is configured to get 20% of the bandwidth available for the S-comp. Bandwidth available for the VEB in turn is equally distributed between VEB VSIs.

7.8.1.2.1 Arbitration Schemes

The XL710 supports three types of arbitration schemes: Weighted Strict Priority, Weighted Round Robin and combination of both. Arbitration scheme can be specified by software as a part of the bandwidth distribution definition.

A weighted strict priority scheme allows software to assign different bandwidth share priorities to the entities that can be scheduled. Multiple entities can be configured with weighted strict priority. Entity with higher priority is allowed to fully consume its bandwidth share before virtual port with lower priority can use a bandwidth share allocated to it. This scheme allows one or more lightly loaded ingress virtual ports with high bandwidth share priority to use its bandwidth as soon as data becomes available on those ports independent of the ports with lower priority. If virtual port is configured to weighted strict priority, software can specify its bandwidth share to be unlimited by allocating 127 bandwidth share credits to the virtual port. Strict Priority arbitration scheme is a Weighted Strict Priority with an unlimited bandwidth allocation credits.

A weighted round robin scheme allows virtual ports to use allocated portions of bandwidth in round robin fashion. In this configuration, all virtual ports are assumed to have same bandwidth allocation priority and ports are interleaving in round robin sequence while using a portion of the bandwidth allocated to the port.

A combined configuration allows software to specify per switching component two sets of ingress virtual ports: one sharing bandwidth using weighted strict priority scheme and another sharing bandwidth using weighted round robin scheme. Virtual ports belonging to the weighted round robin group can use their bandwidth share only if all virtual ports in weighted strict priority group either fully used their bandwidth share or cannot use their share due to other restrictions (e.g. no data available for the port, or the port exceeded its static rate limit).

Software is allowed to modify bandwidth allocation of the virtual port on the fly using the Admin Queue commands described in Section 7.8.4. New bandwidth allocation may take effect after a few scheduling cycles.

7.8.1.3 Static Rate Limiting

The XL710 allows the configuration of a maximum bandwidth limit for each virtual port of switching element (both ingress and egress ports) and for the TCs/UPs on ETS/SLA enabled ports. This limit specifies maximum amount of bandwidth that can be consumed by the virtual port. Limiting maximum bandwidth of the uplink (egress) port of the switching element effectively limits the total bandwidth that can be used by all virtual (ingress) ports of that switching element. This causes propagation of the maximal bandwidth limit thru the switching hierarchy. Maximum bandwidth allocated for a virtual port should not exceed the maximum bandwidth of the respective uplink port (if configured). The sum of the bandwidth limits of all virtual ports of the switching element may exceed a bandwidth limit of uplink port to allow efficient bandwidth utilization.

Maximum bandwidth limit can be configured for the ETS Traffic Class Groups and Traffic Classes. See Section 7.8.1.4.

Maximum bandwidth limits can be used to share bandwidth between VSIs. This is not the most efficient method, but it is still a supported bandwidth distribution scheme. In this case, relative bandwidth allocation should match a maximum bandwidth limit for each port, and the sum of bandwidth limits should add up to the total bandwidth available for the switching element.

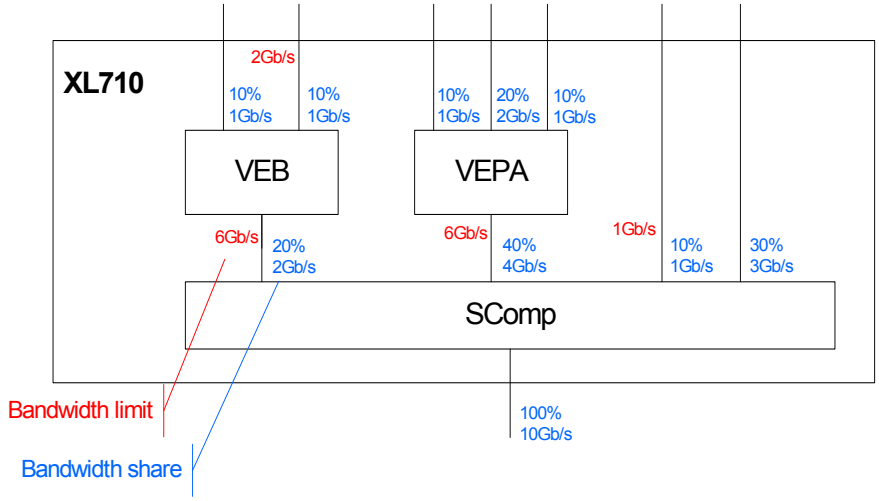


Figure 7-77. Example of Limiting Maximum Bandwidth

Figure 7-77 shows an example of bandwidth distribution between switching elements and their virtual ports in conjunction with limiting the maximum bandwidth available for some of virtual ports. Bandwidth limit on the uplink port of VEB limits total bandwidth available for that switching element (e.g. 6Gb/s for the VEB uplink port on Figure 7-77). Some virtual ports of the switching element may be bandwidth limited as well (e.g. right side VSI of VEB is limited to 2Gb/s). Bandwidth limit of an individual virtual port should be less or equal to the bandwidth limit of the switching element or its uplink port. However, the sum of bandwidth limits of the virtual ports should (for the efficient bandwidth utilization) exceed a bandwidth limit of the switching element. For example, the sum of bandwidth limits of S-comp virtual ports is 13Gb/s, while the total available bandwidth is 10Gb/s.

The XL710 allows configurable accumulation of the bandwidth limit credits. If a virtual port, with static rate limit enabled, does not use allowed bandwidth, the port can accumulate rate limiting credits and use more bandwidth than allowed by static rate limiter up to the configurable limit for the burst of Quanta bytes. The XL710 maintains single Quanta for the entire chip; this is configured via the TSCDQUANTA register.

The XL710 static rate limiters can be configured to the minimal rate limit of 50Mb/s with a granularity of 50Mb/s. The maximum rate limit supported by the XL710 is 40Gb/s.

Static bandwidth limits can be configured to VSIs and Switching components in all supported configuration schemes. See Section 7.8.2.

7.8.1.4 ETS

In a DCB enabled environment, a single physical port or VSI can be shared by multiple traffic types. The Enhanced Transmission Selection (ETS) standard defines sharing of the virtual or physical link bandwidth by multiple types of traffic. Different types of traffic are segregated in Traffic Classes based on User Priorities. Each Traffic Class is mapped to one or more User Priorities. Mapping of User Priorities to the Traffic Classes and sharing of bandwidth of VSI by TCs is defined by the ETS specification. Though ETS bandwidth allocation is defined for the egress ports of the networking devices, it also may apply to the ingress port configuration of adjacent devices sharing the same physical or virtual link.

The XL710 allows ETS to be enabled and independently configured for the uplink ports and VSIs.When enabled, ETS defines number of TCs allocated per port, UP to TC mapping and distribution of port bandwidth between TCs. Due to limited amount of scheduling resources, the XL710 limits the number of TCs/UPs that can be installed to average of two per VSI.

Per Traffic Type bandwidth allocation of VSIs and physical port ETS are independent, each VSI or switching component can be configured with different number of Traffic Classes and different bandwidth distribution. However, ETS configuration of the entire internal switching fabric must be consistent For example, Traffic Classes enabled for VSI, must be enabled for respective switching components and physical port, and a total bandwidth allocated to all VSIs for particular TC must be equal to the bandwidth allocated for that TC on switching component.

The XL710 supports a mixed bandwidth allocation for the TCs, when some TCs are configured as a Strict Priority TCs and others are configured as ETS TCs. Strict Priority TCs have a scheduling priority over ETS TCs and can consume unlimited amount of bandwidth. Strict Priority TCs are intended to be lightly loaded TCs carrying high priority traffic that does not consume much bandwidth. ETS TCs are sharing bandwidth unused by Strict Priority TCs relatively to their bandwidth share allocation. See Section 7.8.1.2.1 for the description of arbitration schemes.

7.8.1.5 Transmit Queues and Queue Sets Assignment

Queue Set is a set of transmit queues carrying unaccelerated traffic generated by the networking stack or FCoE.

A VSI may have multiple sets of transmit queues associated with it.

A VSI configured to the carry single type of traffic (one TC) has a one pair of transmit Queue Sets assigned to it: one State Queue Set and one Stateless Queue Set.

A VSI configured to carry multiple types of traffics has one pair of Queue Sets allocated for each Traffic Class configured for the port.

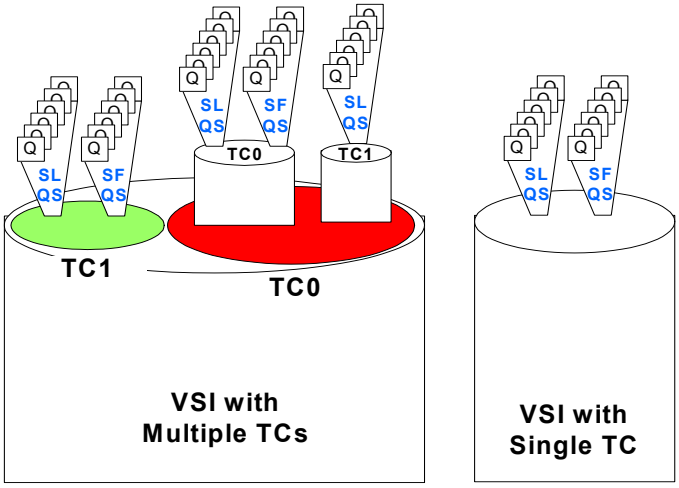
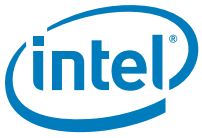


Figure 7-78. Queue Set Assignment Diagram



Allocation of Queue Sets to VSIs is managed by firmware. This is done either at VSI creation time, or when ETS/SLA configuration of the VSI is enabled or modified by software. See Section 7.8.4.8 and Section 7.8.4.17.

Bandwidth allocation between Queue Sets in the pair is configured separately in addition to bandwidth distribution between VSIs and ETS bandwidth distribution within VSI.

All queues assigned to the same Queue Set belong to the same Traffic Class, and have even bandwidth allocation.

Assignment of the Queues to Queue Sets is performed by software.

To comply with Function Level Reset requirement all Queues associated with the same Queue Set must belong to the same PCIe function.

7.8.1.5.1 Queue Set Reassignment

Certain modifications of Scheduler Configuration such as changing number of TCs enabled for VSI (Section 7.8.4.8) may result in reassignment of active Transmit Queues from one Queue Set to another. Transmit Queue reassignment involves changing of the Transmit Queue context to carry a new Queue Set Handle.

If both Queue Sets remain active after configuration change, then software can perform Transmit Queue reassignment lazily in the background in parallel with the regular Scheduling operation. Transmit Queues might get scheduled once via old Queue Set before switching to the scheduling using a new Queue Set.

If an old Queue Set should be de-allocated, then software should:

- Upon completion of the configuration change, start reassigning Transmit Queues to the new Queue Set
- Meanwhile hardware might keep scheduling Transmit Queues using old Queue Set. The re-assigned Transmit Queue might be scheduled using an old Queue Set at most once.
- When the Transmit Queue reassignment is completed, software should notify firmware using the Release Queue Set AdminQ command (Section 7.8.4.9) that it has completed its part of transition and that firmware can proceed with Queue Set de-allocation

Software should not request release of the Queue Set, if it still has Transmit Queues assigned to it.

7.8.2 Transmit Scheduler Configuration Schemes

The XL710 Scheduler supports wide variety of configuration schemes. Programming Interfaces described in Section 7.8.4 allow software to configure ETS-Based configuration scheme described in Section 7.8.2.1.

7.8.2.1 ETS-Based Scheduler Configuration Scheme

Figure 7-79 shows a logical diagram of ETS-based configuration scheme.

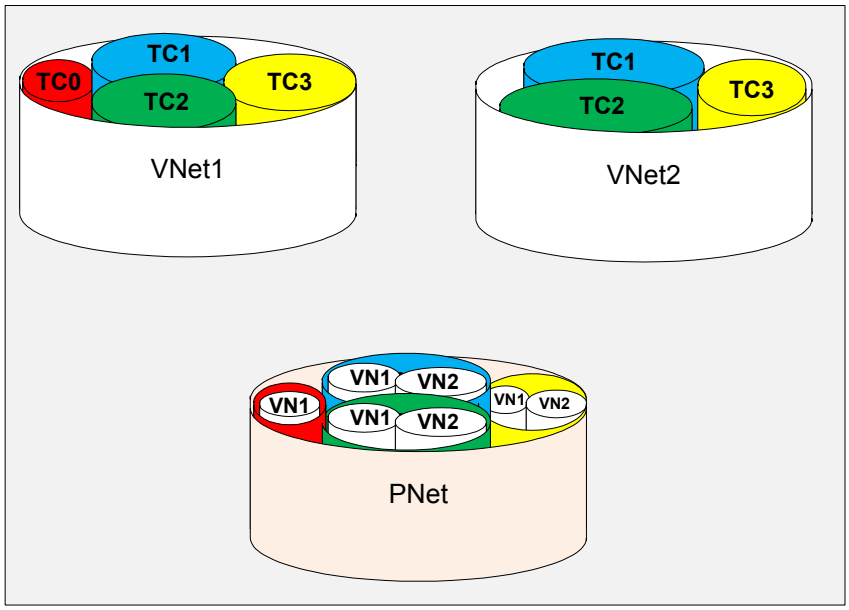


Figure 7-79. ETS-Based Scheduler Configuration Scheme

The diagram above shows two Virtual Networks. Each Virtual Network carries multiple types of traffic. Each traffic type can be identified by its Traffic Class or User Priority. Virtual Networks are merging to the Physical Network. Physical Network shows a bandwidth distribution between virtual networks and their traffic types in ETS-based scheduler configuration scheme.

In the ETS-based configuration, bandwidth is first distributed between types of traffic on the wire and then between virtual networks within each traffic type. Note that traffic type bandwidth distribution might be more complex than shown on the Figure 7-79 and may include multiple bandwidth distribution layers. These may include bandwidth distribution between traffic classes and then between user priorities within traffic class. Similarly, bandwidth distribution between Virtual Networks might involve multiple bandwidth distribution levels (e.g. virtual switches, and their virtual ports).

If virtual network does not have traffic currently available of a particular traffic type (e.g. VNet2.TC3), then the physical network attempts to preserve ratio of bandwidth allocated for that traffic type and the remaining bandwidth is distributed between other Virtual Networks having same traffic type available (e.g. VNet1.TC3).

Such per-traffic type bandwidth distribution does not guarantee any bandwidth allocation for the virtual network, but does allow full control over traffic distribution on the wire per traffic type (a.k.a. ETS). This scheme does guarantee virtual network bandwidth allocation per type of traffic (TC or UP).

This configuration is very suitable for the DCB-enabled fabric. As long as traffic is available for each traffic type enabled on the virtual network the bandwidth distribution on the physical wire will match ETS configuration of the physical port.

ETS-based configuration scheme allows configure an ETS for the Switching Component or VSI connected to the Physical Port, and hierarchically distribute per-traffic type bandwidth between all other Switching Components (VEB/PA) and VSIs.

Figure 7-80 below shows an example of bandwidth distribution in ETS-based scheduler configuration scheme.

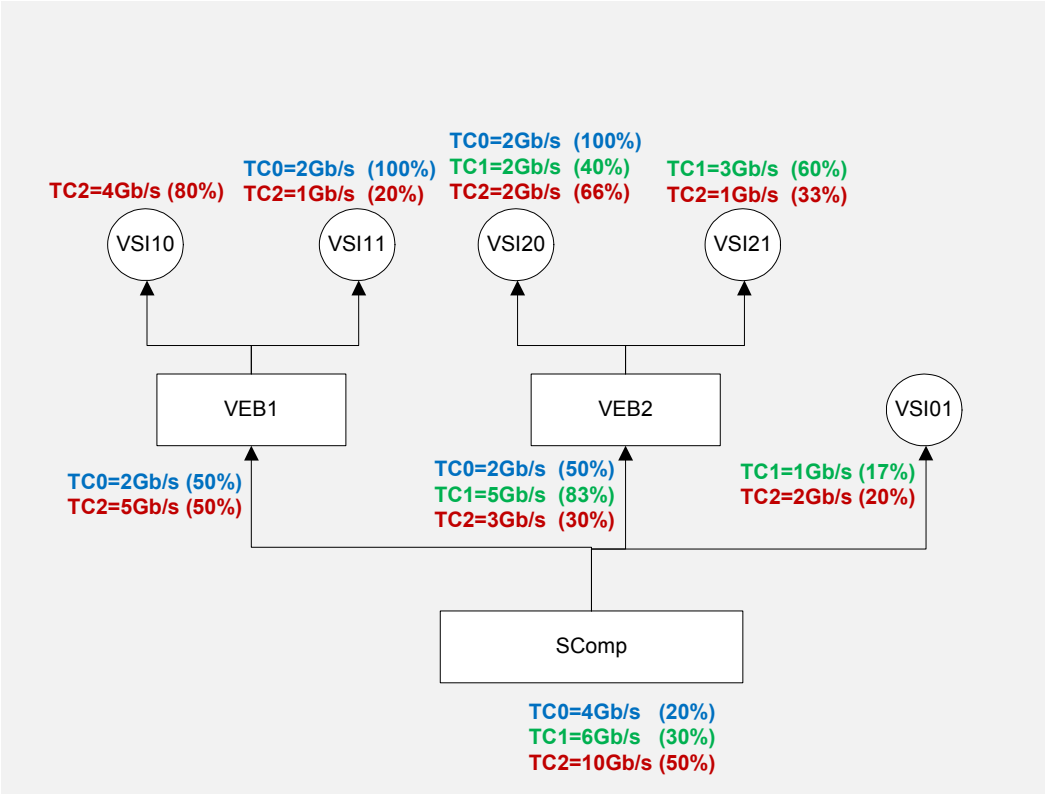


Figure 7-80. ETS-Based Bandwidth Configuration Example

Figure 7-80 shows a single port configuration with SComp. Two VEB switching components are connected to SComp (VEB1 and VEB2), along with VSI (VSI01). Each VEB has two VSIs (VSI10 and VSI11, and VSI20 and VSI21 respectively). Each one of the Switching Components is configured with bandwidth allocation per traffic type. The Physical Port is configured with ETS, showing one level of bandwidth distribution (per TC). The Physical Port can be configured with two level ETS including bandwidth allocation per TC, and bandwidth allocation per UP within respective TC.

Bandwidth allocation between Switching Components and VSIs on each level of switching topology is relative within same Traffic Class and hierarchical. Figure 7-80 above shows an example bandwidth allocation in Gb/s and relative percentage. Color coding helps to visualize bandwidth distribution between switching components and VSIs within TC.

- SComp level.
- VEB Level (VEB1, VEB2 VSI01).
- Per Traffic Class bandwidth is distributed between VEBs and VSI. Bandwidth distribution is relative. Total of 100% of bandwidth per Traffic Class.
  - TC0: VEB1=50%, VEB2=50%
  - TC1: VEB2=83%, VSI01=17%
  - TC2: VEB1=50%, VEB2=30%, VSI01=20%

Total effective bandwidth (in Gb/s) allocated for TCs on VEB level equals to the bandwidth allocated per TC for SComp. (i.e. VEB1.TC0 + VEB2.TC0 = SComp.TC0, VEB2.TC1 + VSI01.TC1 = SComp.TC1, and VEB1.TC2 + VEB2.TC2 + VSI01.TC2 = SComp.TC2).

- VSI Level (VSI10, VSI11) and (VSI20, VSI21).
- Per Traffic Class bandwidth is distributed between VSIs within respective VEB. Bandwidth allocation for each Traffic Class is relative within VEB.
-
- VEB1 VSIs:
  - TC0: VSI11=100%
  - TC1: 0%
  - TC2: VSI10=80%, VSI11=20%

Total effective bandwidth (in Gb/s) allocated for TCs on VEB1 VSI level (VSI10 and VSI11) equals to the bandwidth allocated per TC for VEB1. (i.e. VSI10.TC0 + VSI11.TC0 = VEB1.TC0, and VSI11.TC2 = VEB1.TC2)

VEB2 VSIs:

  - TC0: VSI20=100%
  - TC1: VSI20=40%, VSI21=60%
  - TC2: VSI20=66%, VSI21=33%

Total effective bandwidth (in Gb/s) allocated for TCs on VEB2 VSI level (VSI20 and VSI21) equals to the bandwidth allocated per TC for VEB2. (i.e. VSI20.TC0 = VEB2.TC0, VSI20.TC1 + VSI21.TC1 = VEB2.TC1, and VSI20.TC2 + VSI21.TC2 = VEB2.TC2).

Bandwidth allocation must be consistent. Bandwidth allocated to the Traffic Class on the SComp level, must be equal to the total bandwidth allocated for the same Traffic Class on the level of VEBs and VSIs connected directly to the SComp (VEB1, VEB2 and VSI01). Bandwidth allocated to the Traffic Class on VEB (VEB1) level must be equal to the total bandwidth allocated to that Traffic Class on VSI level (VSI10 and VSI11).

The ETS-Based scheduler configuration scheme does not support bandwidth allocation per Switching Component or VSI. It supports only bandwidth allocation for Switching Component or VSI within particular Traffic Class.

ETS-Based Scheduler configuration scheme allows software to instantiate bandwidth limits for each of the Switching Components and VSIs. Bandwidth limit can be programmed to restrict total bandwidth available for Switching Component and VSI, or limit bandwidth available for the Switching Component or VSI for the particular type of traffic, identified by User Priority or Traffic Class.

7.8.3 Scheduling

The The XL710 Scheduler operates with Queue Sets. Every scheduling cycle Scheduler selects a next Queue Set to be served. The amount of traffic that can be generated on the wire by transmit queues associated with the selected Queue Set is limited to the configurable value - Quanta. Selection of the Queue Set is based on the bandwidth share, bandwidth limit and ETS configuration of the VSI and uplink ports of the internal switching components. The Scheduler fairly distributes available bandwidth

among Queue Sets that have transmit queues with work available staying within limits of the bandwidth allocation dictated by bandwidth limit, bandwidth share and ETS constraints. Bandwidth distribution between Queues within same Queue Set is intended to be even.

The Scheduler limits amount of data that can be transmitted by selected Queue Set to the configurable value - Quanta. Single Quanta may allow transmission of multiple ethernet frames. Quanta constrains the burstiness of the traffic generated by the XL710. Independent of the rate limit implied on the Queue Set, the XL710 transmits Quanta worth of data with the wire speed of the associated physical port. Different Queue Sets may have different Quantas associated with them. Supported Quanta values vary in the range of 1K-128K bytes, and are configured via the GLSCD_QUANTA register. The default Quanta value is 4KB. Quanta are configured via an NVRAM setting and software can retrieve current Quanta value by reading the GLSCD_QUANTA register.

Scheduler configuration is derived from the configuration of internal switching elements, and in most cases follows hierarchy of the switching elements.

The XL710 does not schedule individual packets. All packets belonging to the same transmit queue will follow same route within internal switching hierarchy and are subject to the same set of bandwidth constraints.

When the XL710 transmits packet from one of the transmit queues of the selected Queue Set, it effectively uses bandwidth of all switching components in internal switching hierarchy leading from the virtual port that Queue Set is associated with to the physical port of the chip. Bandwidth constraints applied to the switching component on the route from the Queue Set to physical port of the chip automatically apply to the Queue Set. This means that a particular Queue Set can be scheduled only if there is enough bandwidth available in all switching components on the transmission route.

The Scheduler is aware of the PFC, and stops scheduling Queue Sets associated with paused traffic class to avoid head of line blocking in pipeline.

7.8.3.1 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 keep configuration consistent, direct access to the internal scheduler configuration is restricted to firmware.

The Scheduler does not expose its internal configuration tables to the standard deployment software. A set of registers allowing a direct access to the internal scheduler structures is exposed to firmware to allow programming of Scheduler configuration tables. Those registers are exposed in debug mode to bring up and other privileged software. A default configuration of the scheduling tables is done by firmware and based on the internal switch and scheduling configuration profiles kept in NVRAM. Software is allowed to modify the default configuration by proxying its requests through firmware using Admin Queue commands. Software should use an admin queue interface to communicate its scheduler configuration requests with firmware. Firmware is responsible to constrain access to the scheduling configuration tables based on the software privilege level.

The XL710 maintains single Quanta for entire chip, configured via the TSCDQUANTA register. Quanta accounts for payload and headers generated by software for the LAN traffic and payload. Quanta does not account for the L2 Tags, Ethernet, FCoE CRCs and FcoE padding inserted by hardware on the fly.

The Scheduler is intended to be configured by Physical Function driver using the Admin Queue commands described in Section 7.8.4. A Physical Function driver is considered to be a trusted software, and firmware should accept requested configuration changes. Most of the switching components are owned either by single physical function or by one of the virtual functions associated with same physical functions. If switching component is shared among physical functions (e.g. s-comp in MFP

environment), this component is either configured by EVB agent running in firmware, or by service running in one of physical function. In the later case, firmware relies on software to coordinate scheduler configuration among different physical functions.

Most of the scheduler configuration is fully controlled by physical function driver in the standard deployment environment. If virtual function driver or configuration software running in virtual functions is granted control over one of the scheduling configuration parameters (e.g. the ETS/SLA configuration of the virtual port, scheduler configuration commands should be proxied via PF driver).

The Scheduler supports up to 768 pairs of Queue Sets, regardless of the number of QueueSets populated in each pair.

7.8.4 Admin Queue Commands

The Internal structure of the Scheduler configuration tables is not exposed to software. Firmware is responsible for the scheduler configuration and provides software with admin queue interface allowing alter scheduler configuration. In some configurations, a privileged software is allowed to perform direct programming of the scheduler configuration tables bypassing firmware. Firmware or privileged software should maintain a mapping table of the scheduling resources assigned to each PCIe function.

This section defines admin queue commands that should be used by software to program scheduler configuration attributes.

A standard XL710 scheduler configuration is based on the configuration of the internal switch and its components including instantiated switching components, their VSIs, S-channels, and connectivity between internal switching component. Scheduler configuration adds bandwidth management attributes to the configured switching components, including allocation of bandwidth for the switching components, and their VSIs, enablement and configuring ETS or bandwidth allocation per Traffic Class, and instantiation of rate limiters.

The AdminQ commands described in this section are intended to be used by PF driver only. VF driver is not allowed directly participate in Scheduler configuration. The PF driver is considered to be a trusted software component within this PF. Firmware will validate that AdminQ commands impacting configuration of the components owned by the PF driver issuing those commands. This will be done using association of the AdminQ with particular PF and information kept in the switch configuration tables. When PF performs configuration on behalf of VF, corresponding VF must be provided within AdminQ command.

Table 7-155 provides a list of admin queue commands allowing configuration of internal scheduler structures. Creation of Switching Components, including VSIs is described in Chapter 7.4.9.4.2.

Firmware must avoid partial command execution. All command validation and resource availability checks must be done prior to updating Scheduler Configuration tables. Aborting command with partially updated Scheduler Configuration table may lead to unpredictable hardware behavior.

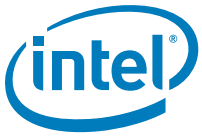


Table 7-155. Scheduler Configuration Admin Queue Commands

Command	Opcode	Brief description	Detailed Description
Configure VSI Bandwidth Limit	0x0400	This command allows software to configure bandwidth limit to the specified VSI of the specified switch component. This is a global bandwidth limit that applies to all Traffic Classes enabled for VSI.	Section 7.8.4.6
Configure VSI Bandwidth Limit per Traffic Type	0x0406	This command allows to configure bandwidth limits assigned to TCs of the specified VSI exposed to PCIe interface	Section 7.8.4.7
Configure VSI Bandwidth Allocation per Traffic Type	0x0407	This command allows to specify Traffic Types enable for VSI and configure relative bandwidth allocation of VSI within each traffic type (TC). This command is valid for ETS-Based configuration only.	Section 7.8.4.8
Query VSI Bandwidth Configuration	0x0408	This command allows to retrieve current configuration of the specified VSI of the specified switching component. This configuration should include bandwidth limit, bandwidth allocation.	Section 7.8.4.15
Query VSI Bandwidth Configuration per Traffic Type	0x040A	This command allows to retrieve current bandwidth configuration of VSI within each Traffic Type (TC). This command is valid for ETS-Based configuration only.	Section 7.8.4.16
Configure Switching Component Bandwidth Limit	0x0410	This command allows software to enable, disable or modify bandwidth limit of the egress port of specified switching component. This is a global bandwidth limit that applies to all Traffic Classes enabled for Switching Component.	Section 7.8.4.9
Enable Physical Port ETS	0x0413	This command allows enable ETS configuration of the egress port of the specified switching component or VSI directly connected to the Physical Port. It carries full specification of ETS configuration for the port, including number of TCs. user priority to TC mapping, TC bandwidth allocation, TC arbitration scheme, and bandwidth allocation. This command is valid for switching components directly connected to the Physical Port only. This command is valid in ETS-Based configuration only,	Section 7.8.4.10
Modify Physical Port ETS	0x0414	This command allows modify ETS configuration of the egress port of the specified switching component or VSI directly connected to the Physical Port. It carries full specification of ETS configuration for the port, including number of TCs. UP to TC mapping, TC bandwidth allocation, TC arbitration scheme, and bandwidth allocation. This command is valid for switching components directly connected to the Physical Port only. This command is valid in ETS-Based configuration only,	Section 7.8.4.10
Disable Physical Port ETS	0x0415	This command allows disabled ETS configuration of the egress port of the specified switching component or VSI directly connected to the physical port. It carries full specification of ETS configuration for the port, including number of TCs. user priority to TC mapping, TC bandwidth allocation, TC arbitration scheme, and bandwidth allocation. This command is valid for switching components directly connected to the Physical Port only. This command is valid in ETS-Based configuration only,	Section 7.8.4.10
Configure Switching Component Bandwidth Limit per Traffic Type	0x0416	This command allows to enable, disable or modify bandwidth limits assigned to TCs of the egress port of the specified switching component. This command is valid in ETS-Based configuration only,	Section 7.8.4.13

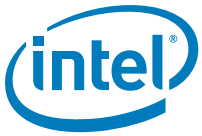
Table 7-155. Scheduler Configuration Admin Queue Commands

Command	Opcode	Brief description	Detailed Description
Configure Switching Component Bandwidth Allocation per Traffic Type	0x0417	This command allows to specify Traffic Types enabled for Switching Components and configure relative bandwidth allocation of Switching Component within each traffic type (TC) This command is valid for ETS-Based configuration only.	Section 7.8.4.14
Suspend Port's TX Traffic	0x041B	This command allows PF to Suspend port's Transmit traffic. The command is completed after all Qsets belong to the port are suspended and the Tx pipe of the port is drained. This allow SW to modify port's configuration like DCB setting. This command is valid only under SFP mode.	Section 7.8.4.11
Resume PF Traffic	0x041C	This command is used to resume suspended Qsets. It will resume all Qsets belong to the PF.	Section 7.8.4.12
Query Switching Component Bandwidth Configuration	0x0418	This command allows to retrieve current configuration of the switching component.	Section 7.8.4.17
Query Physical Port ETS Configuration	0x0419	This command allows to retrieve current ETS configuration of the switching component. This should include number of valid TCs bandwidth allocation and bandwidth limit for TCs. This command is valid for switching components directly connected to the Physical Port only. This command is valid in ETS-Based configuration only,	Section 7.8.4.18
Query Switching Component Bandwidth Configuration per Traffic Type	0x041A	This command allows to retrieve current bandwidth configuration of Switching Component within each Traffic Type (TC). This command is valid for ETS-Based configuration only. This command is valid only under SFP mode.	Section 7.8.4.19
Add VSI	0x0210	This is a switch configuration command. Execution of this command affects scheduler configuration. This command allocates VSI and returns a VSI SEID that should be used to modify VSI bandwidth configuration.	Section 7.4.9.5.4.1, Section 7.8.4.2
Add VEB, Add_PE	0x0210	This is a switch configuration command. Execution of this command affects scheduler configuration. This command allocates switching component, and returns SEID that should be used to modify switching component bandwidth configuration.	Section 7.4.9.5.5.1, Section 7.4.9.5.6.1, Section 7.8.4.2
Delete Element	0x0210	This is a switch configuration command. Execution of this command affects scheduler configuration.This command deallocates switching elements including VSIs. Deallocation of the switching element results in deallocation of he scheduling resources associated with this switching element, and cleanup of scheduler configuration tables.	Section 7.4.9.5.7.1, Section 7.8.4.2

7.8.4.1 Scheduler Initialization Flow

Main scheduler configuration and initialization flows are initiated by software and performed by firmware using AdminQ interface. The Sections below define various AdminQ commands that can be used by software to configure Transmit Scheduler.

Prior to initialization of AdminQ Command interface, firmware can use information available in NVRAM to perform default configuration of the scheduler. Similar to the standard configuration, default scheduler configuration follows a default configuration of the Switching Components.



7.8.4.2 Allocation of Switching Elements

A default configuration of the scheduler tables is performed as a part of the internal switch configuration. Each time software adds a new switching component (Section 7.4.9.5.5.1, Section 7.4.9.5.6.1) or VSI (Section 7.4.9.5.4.1) to the internal switch, the configuration tables of the scheduler are updated respectively by firmware. A new added switching component or VSI is configured with a default bandwidth allocation and bandwidth limit disabled. The Allocated Switching Component or VSI is created with single TC enabled (TC0). Software can enable TCs either by using the bitmask in the Add VSI or Add VEB/PA commands, or by using the AQ command described in Section 7.8.4.7 and Section 7.8.4.14. Along with default bandwidth, distribution firmware allocates a pair of Queue Sets for each configured TC of the new allocated VSI.

With completion of a switching element allocation request, software is provided with a handle per allocated Queue Set. The Queue Set Handle should be used to associate Queue with a Queue Set.

A default bandwidth allocation provides a new instantiated switching component and VSI with a minimal bandwidth share with respect to other switching components or ports sharing same virtual link. Software can modify the default bandwidth allocations using the Admin Queue command described in Chapter 7.8.4.8 and Chapter 7.8.4.14.

Initial configuration of the scheduler tables is done as a part of the initial configuration of the internal switch and is based on the chip profiles. No software involvement is required to perform initial scheduler configuration. As soon as firmware completes initial configuration, scheduler is enabled and ready to schedule transmit work.

Software is allowed to change internal switch configuration or modify scheduler configuration by adding/removing internal switching components and/or VSIs, or by changing bandwidth distribution, bandwidth limits and ETS configuration of Physical Port, switching components or VSIs. Some configuration changes, causing modification of the internal switching structure, may require firmware to partially or fully suspend scheduler operation until it completes a local or global modification of the internal scheduler tables. Certain scheduler configuration changes, such as bandwidth redistribution, or modification of the bandwidth limit, can be done without suspending normal scheduler operation.

Firmware is responsible to track allocated resources and their configuration, and manage configuration of internal scheduler tables. If software performs incremental modification of the internal switching structure, it should rely on firmware to maintain consistency of internal scheduler tables, and fit requested modifications to the existing configuration. Each allocated switching element, including instance of internal switch or VSI is assigned a unique within the chip identification number - SEID. For the description see Chapter 7.4.2.1. SEID should be used by software when referencing to the switching component during scheduler configuration.

7.8.4.2.1 Queue Set Assignment

As a part of the default scheduler configuration, each instantiated VSI is supplied with a pair of Queue Sets per configured TC - one for stateless and one for the stateful Queues. Firmware manages shared pool of the Queue Sets. Allocation and deallocation of Queue Sets is hidden from software, and done based on the software resource allocation/deallocation requests. Handles of allocated Queue Sets are returned to software as a part of completion of Admin Queue command and should be used by software to associate transmit Queues and Queue Sets. Firmware allocates a single Queue Set Handle to the LAN Queue Sets pair.

Software can change a default configuration of the VSI. (e.g. change number of configured TCs). A change in VSI configuration results in changing the number of TCs allocated for the port, causing firmware to reallocate QueueSets assigned to the VSI to match number of TCs.

Allocated Queue Set Handles are provided in completion of respective AdminQ command. Firmware always returns 8 Queue Set handles. Order of handles returned matching an order of TCs enabled for VSI (from 0 to 7). Invalid Queue Set handles carry value of 0xffff. Queue Set handles are returned for all AdminQ commands that can be result in allocation or change of allocation of Queue Sets (Create VSI, Configure VSI Bandwidth Allocation per Traffic Type, Configure VSI Bandwidth Limit per Traffic Type). For more details see Section 7.8.4.7 and Section 7.8.4.8. Software can also use AdminQ command to query Queue Sets allocated for particular VSI, see Section 7.8.4.16.

Queue Set assignment is completely hidden from software. Software should use a QueueSet Handle when referring to the particular Queue Set.

Assignment of Queues to Queue Sets is not described here, and outside of the scope of the scheduler configuration definition. If software changes configuration of the VSI and this change results in reallocation of Queue Sets, as long as TC remains assigned to the VSI, all Queue associated with the Queue Set identified by the VSI and TC will remain associated with the Queue Set. If software modifies the VSI configuration, and eliminates the TC which has active Queue associated with, software must resolve contention and reassign Queues.

7.8.4.3 Release of the Switching Elements

Software can use the AQ Commands Delete Element and the Configure Bandwidth Allocation/Limit of the VSI/Switching Component per Traffic Type to remove Switching Components and VSIs or change their configuration per Traffic Type. Removal of the switching Component or VSI is allowed only if the respective uplink switching element has been previously removed or a Traffic Type configuration has been previously adjusted to the uplink switching element. For example, if software decides to change VEB configuration per Traffic Type and reduce number of Traffic Types enabled for VEB, it must first respectively adjust Traffic Types for all VSIs allocated for that VEB. Software may request to remove VEB, only after it removed all VSIs allocated for that VEB.

Software may request to remove VSI only if Queue Set associated with that VSI has all Transmit Queues removed, see Section 7.8.4.3.1.

7.8.4.3.1 Queue Set Release

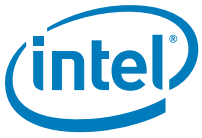
Certain modifications to the Scheduler configuration, such as change in number of TCs enabled for the VSI, or change in TC mapping or VSI deallocation, may lead to release of the Queue Set previously allocated for VSI.

Prior to performing any operation that may require release of the Queue Set, Software must remove from the Queue Set all Queues that were previously associated with the Queue Set. For Both the LAN and FCoE Qs, this operation requires Q disable. In any case, Software must either suspend or disable all Qs that were previously associated with the Queue Set. Completion of Q disable operation must be confirmed by hardware.

Scheduler configuration firmware validates that a Queue Set subject to release does not have Qs associated with, and fails the requested operation, reporting EBUSY error, if it does.

7.8.4.4 Common Processing and Error Handling

The generic structure of Admin Queue Command is defined in Section 7.10.5. All Transmit Scheduler Admin Queue Commands described in the sections below are derived from the generic Admin Queue commands, and using a generic Admin Queue Command structure as a baseline.



Sections describing Transmit Scheduler Admin Queue Commands are relating to the Admin Queue command fields that are specific to the particular command. For the description of the common Admin Queue Command fields, see Section 7.10.5.

The Transmit Scheduler uses various types of Admin Queue Commands. Some of them Direct Admin Queue Commands (all command information is provided within the body of command) and some Indirect (additional data is provided within the buffer referred by Admin Queue COmmand). Some commands report completion within Admin Queue command body and others carry completion information within buffer provided by original command. Each Transmit Scheduler Admin Queue command will indicate its type in the command description. For the detailed description of all Admin Queue Command types and their differences, see Section 7.10.5.

Transmit Scheduler Admin Queue commands use generic error reporting structure described in Section 7.10.8. Table 7-156 lists errors specific to Transmit Scheduler Admin Queue Commands and reported in Admin Queue Completions, providing their cause and a reported error code. All error codes are defined in Table 7-201.

Table 7-156. 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, QS Handle. Those handles must be used by software for the future reference to the allocated resources. This error indicates that handle provided by software is not valid.
Invalid SEID	ENOENT	SEID provided within Admin Queue command is not valid
Parameter out of range	ERANGE	Provided argument does not match definition (e.g. 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 AdminQ commands should not be used by software (e.g. in ETS-based configuration software should not attempt configure bandwidth allocation to VSI or Switching Component). This error indicates that software attempted AdminQ command that is not valid in the current
Resource is busy	EBUSY	Software attempted to release Queue Set that had Qs associated with

7.8.4.5 Function Level Reset

Transmit Scheduler configuration tables are updated by firmware as a part of the Function Level Reset processing by the XL710. All scheduling resources allocated for a particular function are released during this process, including VSIs and Queue Sets. Respective resources can be later on allocated to the different PCI Functions.

Scheduler operation is not suspended during Function Level Reset but its performance and bandwidth distribution might be intermittently affected. Upon completion of Function level reset processing, Transmit Scheduler will complete its normal operation.

7.8.4.6 Configure VSI Bandwidth Limit

This command allows software to configure a bandwidth limit for the specified VSI of the switch component that is exposed to PCIe interface. To configure bandwidth limit of the egress port, Software should used command described in Section 7.8.4.9. The XL710 provides hierarchical bandwidth limit, that apply to VSI and all associated TCs and UPs. See Section 7.8.1.3.

Software cannot have both Bandwidth Limit and Bandwidth Limit per Traffic Type enabled for the same VSI. A request to enable Bandwidth Limit for VSI that has Bandwidth Limit per Traffic Type enabled should fail and report an EPERM error.

Under MFP, if the VSI is connected directly to EMP FW will ignore BW parameters of the AQC and will retrieve the parameters of TC enable and BW configuration per PF from MFP Switch configuration in Alternate RAM data and UP to TC mapping.

This is a Direct Admin Queue Command with completion reported within the completion descriptor structure. Table 7-157 describes command format and defines command specific fields.

.

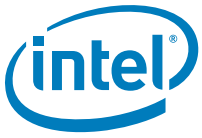
Table 7-157. Configure VSI Bandwidth Limit Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x0400	Command opcode
Datalen	4-5	0	Length of response buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
VSI SEID	16-17		Unique identifier of the VSI.
Reserved1	18-19	0	Reserved. Must be set to 0.
Bandwidth Limit Credits	20-21		Bandwidth limit in Mbit/s. Supported range is 50Mbit/s - 40Gbit/s, in increments of 50Mbit/s 0 indicates that bandwidth limit is disabled One credit corresponds to bandwidth limit of 50Mb/s
Bandwidth Limit Max	24.0-24.2		Max bandwidth limit indicates how much bandwidth limit credits can be accumulated due to inactivity. Single scheduling Quanta consumes multiple bandwidth limit credits. The XL710 allows accumulating of following discrete values of Quanta-worth credits:0,1, 2, and 4 Quanta worth credits. This field is valid, only if bandwidth limit is enabled
Reserved2	24.3-31	0	Reserved. Must be set to 0.

7.8.4.7 Configure VSI Bandwidth Limit per Traffic Type

This command allows software to specify Traffic Classes enabled for VSI and configure a bandwidth limits of TCs enabled for the specified VSI of the switching component.

In an ETS-Based configuration scheme, Software can configure either VSI Bandwidth Limit or VSI Bandwidth Limit per Traffic Type.



Software should keep track of already configured bandwidth limits and provide complete bandwidth limit configuration, and not just do incremental modifications.

Software allowed to change bandwidth limit per traffic type and a traffic classes enabled for VSI that has a VSI bandwidth allocation configured. A new TCs enabled for VSI will have a default bandwidth allocated. Software should use a Configure VSI Bandwidth Allocation per VSI AQ Command to modify a default bandwidth allocation.

Software should not request change of TC configuration of VSI leading to release or remapping of TCs, prior to disassociating Qs with affected Queue Sets. Operation will fail with EBUSY error, if software requested to release Queue Sets having Qs associated with.

The only exception for the above rule is when remapping of only one TC is required. In this case, SW is allowed to use this command when this Qset is suspended and the XL710 Transmit pipeline is drained from packets belongs to this Qset. After the TC remapping takes place, software may resume its transmit activity using the command "Resume PF Traffic" Section 7.8.4.12. EMP FW is required to keep remapped TCs Qset handle for its new location. Currently the XL710 supports transmit pipe draining in port's granularity. This is used by DCBX flows Section 7.8.5.6.1.

Under MFP, if the VSI is connected directly to EMP FW will ignore TC Enable and BW parameters of the AQC and will retrieve the parameters of TC enable and BW configuration per PF from MFP Switch configuration in Alternate RAM data and UP to TC mapping

If TC re-configuration is needed while the VSI is suspended then new generated TC nodes of this VSI will be suspended too.

Software cannot have both Bandwidth Limit and Bandwidth Limit per Traffic Type enabled for the same VSI. Request to enable Bandwidth Limit per Traffic Type for VSI that has Bandwidth Limit enabled should fail and EPERM error reported in completion.

This is an Indirect Admin Queue Command. Software should provide a valid buffer carrying additional command attributes as defined in Table 7-159.

Table 7-158 describes command format, and defines fields specific to the command.

Table 7-158. Configure VSI Bandwidth Limits per Traffic Type Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x0406	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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
VSI SEID	16-17		Unique identifier of the VSI.
Reserved1	18-23	0	Reserved. Must be set to 0.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

Table 7-159 defines format of the command buffer and additional command attributes.

Table 7-159. Configure VSI Bandwidth Limit per Traffic Type Command Buffer

Category	Byte/Bit	Field	Description
TC Valid	0.0-0.7	TC0-TC7 Valid	Valid bit per TC. Should list all TCs enabled for VSI. Must be consistent with Physical Port and uplink Switching Component.
Reserved1	1-15	0	Reserved. Must be set to 0.
TC Bandwidth Limit Credits	16-17	TC0 Bw Limit Credits	This field specifies a bandwidth limit per TC in Mb/s. Supported bandwidth limit range is 50Mb/s-40Gb/s, with increments of 50Mb/s. Bandwidth limit of 0Mb/s indicates that bandwidth limit is disabled Bandwidth limit should be provided for the valid TC handles only. One credit corresponds to bandwidth limit of 50Mb/s
	18-19	TC1 Bw Limit Credits	
	20-21	TC2 Bw Limit Credits	
	22-23	TC3 Bw Limit Credits	
	24-25	TC4 Bw Limit Credits	
	26-27	TC5 Bw Limit Credits	
	28-29	TC6 Bw Limit Credits	
	30-31	TC7 Bw Limit Credits	
TC Max Bandwidth Limit	32.0-32.2	TC0 Max Bw Limit	Max bandwidth limit indicates how much bandwidth limit credits can be accumulated due to inactivity. Single scheduling Quanta consumes multiple bandwidth limit credits. The XL710 allows accumulating of following discrete values of Quanta-worth credits: 0,1, 2, 4, 8,16 32 and 64 Quanta worth credits.
	32.3	Reserved	
	32.4-32.6	TC1 Max Bw Limit	
	32.7	Reserved	
	33.0-33.2	TC2 Max Bw Limit	
	33.3	Reserved	
	33.4-33.6	TC3 Max Bw Limit	
	33.7	Reserved	
	34.0-34.2	TC4 Max Bw Limit	
	34.3	Reserved	
	34.4-34.6	TC5 Max Bw Limit	
	34.7	Reserved	
	35.0-35.2	TC6 Max Bw Limit	
	35.3	Reserved	
	35.4-35.6	TC7 Max Bw Limit	
	35.7	Reserved	
Reserved2	36-63	0	Reserved. Must be set to 0.

Table 7-160 defines format of completion returned in response buffer

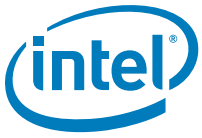


Table 7-160. Configure VSI Bandwidth Limit per Traffic Type Completion Buffer

Category	Byte/ Bit	Field	Description
QueueSet Handles	16-17	QS0 Handle	Handle per Queue Set. Completion will always carry 8 Queue Set handles. Only TCs that were marked as a valid TCs in the Command will have a valid Queue Set handles returned in completion. Invalid handle is marked by 0xffff. Order of Queue Set handles matches order of TCs. Queue Set handles must be used by software to associated transmit Q with Queue Set.
	18-19	QS1 Handle	
	20-21	QS2 Handle	
	22-23	QS3 Handle	
	24-25	QS4 Handle	
	26-27	QS5 Handle	
	28-29	QS6 Handle	
	30-31	QS7 Handle	

7.8.4.8 Configure VSI Bandwidth Allocation per Traffic Type

This command allows software to specify Traffic Classes enabled for VSI, and configure relative bandwidth allocation of VSIs within each Traffic Class.

This command can be used for VSI configured in ETS-Based mode only. Otherwise command would be ignored, and error reported.

To configure VSI bandwidth allocation per Traffic Type, software should provide complete relative bandwidth allocation for all Traffic Classes enabled for VSI.

This is an Indirect Admin Queue Command. Software should provide a buffer that would be used both to provide additional command attributes and for the command completion.

Software allowed to change bandwidth allocation per traffic type and a traffic classes enabled for VSI that has a VSI bandwidth limit enabled. It is responsibility of firmware to adjust hardware configuration to reflect this change.

Software should not request change of TC configuration of VSI leading to release or remapping of TCs, prior to disassociating Qs with affected Queue Sets. Operation will fail with EBUSY error, if software requested to release Queue Sets having Qs associated with.

The only exception for the above rule is when remapping of only one TC is required. In this case, SW is allowed to use this command when this Qset is suspended and the XL710 Transmit pipeline is drained from packets belongs to this Qset. After the TC remapping takes place, software may resume its transmit activity using the command “Resume PF Traffic” Section 7.8.4.12. EMP FW is required to keep remapped TCs Qset handle for its new location. Currently the XL710 supports transmit pipe draining in port’s granularity. This is used by DCBX flows Section 7.8.5.6.1.

Under MFP, if the VSI is connected directly to EMP FW will ignore TC Enable and BW parameters of the AQC and will retrieve the parameters of TC enable and BW configuration per PF from MFP Switch configuration in Alternate RAM data and UP to TC mapping

If TC re-configuration is needed while the VSI is suspended then new generated TC nodes of this VSI will be suspended too.

Table 7-161 describes the command format and defines a command specific fields.

Table 7-161. Configure VSI Bandwidth Allocation per Traffic Type Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x0407	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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
VSI SEID	16-17		Unique identifier of the VSI
Reserved1	18-23	0	Reserved. Must be set to 0.
Data Address high	24-27		Address of buffer. This buffer is used both to convey configuration to firmware, and provide software with a list of TC Handles.
Data Address low	28-31		

Table 7-162 describes format of the command buffer and defines command attributes.

Table 7-162. Configure VSI Bandwidth Allocation per Traffic Type Command Buffer

Category	Byte/Bit	Field	Description
TC Valid	0.0-0.7	TC0-TC7 Valid	Valid bit per TC. Should list all TCs enabled for VSI. Must be consistent with Physical Port and uplink Switching Component. At least one TC musts be valid.
Reserved2	1-3	0	Reserved. Must be set to 0.
TC Bandwidth Share Credits	4	TC0 Bandwidth Credits	Relative VSI credits within same TC with respect to other VSIs or the Switching Components Valid range of credits is 1-127 credits, in increments of the single credit. 127 - indicates infinite credits
	5	TC1 Bandwidth Credits	
	6	TC2 Bandwidth Credits	
	7	TC3 Bandwidth Credits	
	8	TC4 Bandwidth Credits	
	9	TC5 Bandwidth Credits	
	10	TC6 Bandwidth Credits	
	11	TC7 Bandwidth Credits	
Reserved4	12-32	0	Reserved. Must be set to 0.

Table 7-163 defines format of completion returned in response buffer

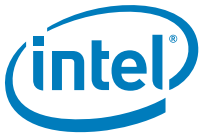


Table 7-163. Configure VSI Bandwidth Allocation per Traffic Type Completion Buffer

Category	Byte/ Bit	Field	Description
QueueSet Handles	16-17	QS0 Handle	Handle per Queue Set. Completion will always carry 8 Queue Set handles. Only TCs that were marked as a valid TCs in the Command will have a valid Queue Set handles returned in completion. Invalid handle is marked by 0xffff. Order of Queue Set handles matches order of TCs. Queue Set handles must be used by software to associated transmit Q with Queue Set.
	18-19	QS1 Handle	
	20-21	QS2 Handle	
	22-23	QS3 Handle	
	24-25	QS4 Handle	
	26-27	QS5 Handle	
	28-29	QS6 Handle	
	30-31	QS7 Handle	

7.8.4.9 Configure Switching Component Bandwidth Limit

This command allows software to enable, disable or modify a bandwidth limit for the specified switch component. By enabling bandwidth limit of switching component, software effectively enabling bandwidth limit on the egress port of the switching component.

This command allows to enable a global bandwidth limit regardless the Traffic Type. to enable bandwidth limit per Traffic Type, software should use command described in Section 7.8.4.13.

Software cannot have both Bandwidth Limit and Bandwidth Limit per Traffic Type enabled for the same Switching Component. Request to enable Bandwidth Limit for Switching Component that has Bandwidth Limit per Traffic Type enabled should fail and EPERM error reported in completion.

Under MFP, EMP FW will ignore TC Enable BW parameters of the AQC. Instead, EMP FW will retrieve the parameters of TC enable and BW configuration per PF from MFP Switch configuration in Alternate RAM data and UP to TC mapping.

The XL710 provides hierarchical bandwidth limit. configuring bandwidth limit to the switching component, software automatically limit a total bandwidth available for all VSI allocated for that switching component. See Section 7.8.1.3.

This is a Direct Admin Queue Command with completion reported within the completion descriptor structure. Table 7-164 describes command format and defines command specific fields.

Table 7-164. Configure Switching Component Bandwidth Limit Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x0410	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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command

Table 7-164. Configure Switching Component Bandwidth Limit Command Fields

Name	Bytes.Bits	Value	Remarks
Switching Component SEID	16-17		Unique identifier of the switching component
Reserved1	18-19	0	Reserved. Must be set to 0.
Bandwidth Limit Credits	20-21		Bandwidth limit in Mbit/s. Supported range is 50Mbit/s - 40Gbit/s, in increments of 50Mbit/s 0 indicates that bandwidth limit is disabled One credit corresponds to bandwidth limit of 50Mb/s
Bandwidth Limit Max	24.0-24.2		Max bandwidth limit indicates how much bandwidth limit credits can be accumulated due to inactivity. Single scheduling Quanta consumes multiple bandwidth limit credits. The XL710 allows accumulating of following discrete values of Quanta-worth credits: 0, 1, 2, and 4 Quanta worth credits.
Reserved2	24.3-31	0	Reserved. Must be set to 0.

7.8.4.10 Enable, Disable or Modify Physical Port ETS

This command allows software to enable, disable or modify ETS configuration of the specified switching component connected directly to the Physical Port. Configuring ETS for the switching component effectively results in configuring ETS for the switching component egress port.

This command can be used only for Switching Components directly connected to the physical port configured to ETS-Based Scheduler configuration. Otherwise, command is ignored, and error is reported.

Software is expected to store a current ETS configuration of the switching component, and fully specify ETS configuration every time it requests its modification. Software cannot provide incremental changes to the ETS configuration.

Firmware will configure TC arbitration to be a weighted round robin arbitration scheme, Software can chose to specify different arbitration scheme (Strict Priority or combination of WRR and WSP), it also has an option to provide an infinite number of credits to Strict Priority TCs.

This is an Indirect Admin Queue command with additional command attributes and completion attributes are provided within the data buffer. Table 7-165 describes command format and defines command specific fields.

Table 7-165. Configure Physical Port ETS Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	 0x0413 0x0414 0x0415	Command opcode 0x0413 - Enable ETS 0x0414 - Modify ETS 0x0415 - Disable ETS
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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command

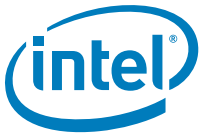


Table 7-165. Configure Physical Port ETS Command Fields

Name	Bytes.Bits	Value	Remarks
Physical Port SEID	16-17		Unique identifier of the physical port
Reserved	18-23	0	Reserved. Must be set to 0.
Data Address high	24-17		Address of buffer.
Data Address low	28-31		

Table 7-166 describes format of the data buffer carrying additional command attributes. Majority of the fields in this table are valid for ETS enable and modify operations only. ETS disable operation should refer Switching Component SEID only.

Table 7-166. Configure Physical Port ETS Command Buffer

Category	Byte/Bit	Field	Description
Reserved1	0-3		Reserved to match VSI configuration format
TC Valid	4.0-4.7	TC0-TC7 Valid	Valid bit per TC. Should list all TCs enabled for Physical Port. At list one TC must be enabled.
Reserved1	5.0-5.7		Reserved. Must be set to 0.
TC Strict Priority	6.0-6.7	TC0-TC7 Strict Priority Flag	Strict priority flag per TC. If set then TC should be arbitrated based on its priority. If software decides to configure one or more TCs to be a strict priority TC, then TC with higher TC number has a higher priority. .
Reserved2	7.0-7.7		Reserved. Must be set to 0.
Reserved3	8-23	0	Reserved to match VSI configuration format
TC Bandwidth Share Credits	24	TC0 Bandwidth Credits	Relative credits per TC. Valid range of credits is 1-127 credits, in increments of the single credit. 127 - indicates infinite credits if TC is configured to strict priority
	25	TC1 Bandwidth Credits	
	26	TC2 Bandwidth Credits	
	27	TC3 Bandwidth Credits	
	28	TC4 Bandwidth Credits	
	29	TC5 Bandwidth Credits	
	30	TC6 Bandwidth Credits	
	31	TC7 Bandwidth Credits	
Reserved4	32-127		Reserved to match VSI configuration format

7.8.4.11 Suspend Port's TX Traffic

This command allows PF to suspend port's Transmit traffic. The command is completed immediately. After all Qsets belonging to the port are suspended and the TX pipe of the port is drained, EMP FW will use "Send Link Status Event" (see Section 3.2.4.1.6 for details) notification to update the PF. During the time period between processing this command and getting the event notification about port draining, the AQ commands "Set PHY Config", "Set MAC Config", and "Set Link and Restart AN" (in Section 3.2.4) for this port, are forbidden and will be rejected with error code "EAGAIN {8}".

Prior to calling this Admin command PF must use "Set Event Mask" (see for Section 3.2.4.1.7 details) to register itself to link events notifications.

This command allows SW to modify port's configuration like DCB setting when it owns LLDP agent.

Port draining is part of DCB configuration flow. This command is used to verify that the TX pipe is drained from any TX packet belong

to the port. This is a pre-condition to some configuration changes of the port TC or DCB. This command is valid only under SFP mode. See below Section 7.8.5.6.1 for more flows' details.

This is a Direct Admin Queue Command with completion reported within the "Direct Command Completion" structure. Table 7-167 describes command format and defines its specific fields

Table 7-167. Suspend Port's TX Traffic

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x041B	Command opcode
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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Physical Port SEID	16-17		Unique identifier of the physical port
Reserved	18-31	0	Reserved. Must be set to 0.

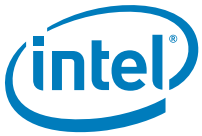
7.8.4.12 Resume PF Traffic

This command is used to resume suspended TX traffic for the PF. It will resume all Qsets belonging to the PF (All its VEBs and VSIs for all their TCs). This command is called by PF driver after the completion of a configuration flow which requires Port's TX pipe to be suspended. See below Section 7.8.5.6.1 for more flows' details.

This is a Direct Admin Queue Command with completion reported within the "Direct Command Completion" structure. Table 7-168 describes command format and defines its specific fields

Table 7-168. Resume PF Traffic

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x041C	Command opcode
Datalen	4-5	0	Length of response buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31	0	Reserved. Must be set to 0.



7.8.4.13 Configure Switching Component Bandwidth Limit per Traffic Type

This command allows software to specify Traffic Classes enabled for Switching Component and configure bandwidth limits of TCs enabled for the specified switch component.

Software should keep track of already configured bandwidth limits, and provide complete bandwidth limit configuration, and not just incremental modifications.

This command can be used only for Switching Components belonging to the physical port configured to ETS-Based Scheduler configuration. Otherwise, command is ignored, and error is reported.

Software is allowed to change bandwidth limit per traffic type and a traffic classes enabled for Switching Component that has a bandwidth allocation configured. A new TCs enabled for Switching Component will have a default bandwidth allocated. Software should use a Configure Switching Component Bandwidth Allocation per VSI AQ Command to modify a default bandwidth allocation.

This command can be used to enable bandwidth limit per traffic class for the physical port.

Software cannot have both Bandwidth Limit and Bandwidth Limit per Traffic Type enabled for the same Switching Component. Request to enable Bandwidth Limit per Traffic Type for the Switching Component that has Bandwidth Limit enabled should fail and EPERM error reported in completion.

Software can configure either Switching Component Bandwidth Limit, or Switching Component TC Bandwidth Limit.

Under MFP, EMP FW will ignore TC Enable BW parameters of the AQC. Instead, EMP FW will retrieve the parameters of TC enable and BW configuration per PF from MFP Switch configuration in Alternate RAM data and UP to TC mapping.

If TC re-configuration is needed while the VSI is suspended then new generated TC nodes of this VSI will be suspended too.

This is an Indirect Admin Queue Command. Software should provide a valid buffer carrying additional command attributes as defined in Table 7-169. Command completion does not provide any additional information beyond status, and does not use a data buffer provided by software for command attributes.

Table 7-169 describes command format and defines command specific fields.

Table 7-169. Configure Switching Component Bandwidth Limits per Traffic Type Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x0416	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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Switching Component SEID	16-17		Unique identifier of the switching component or physical port.

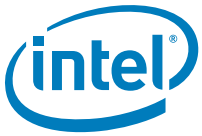
Table 7-169. Configure Switching Component Bandwidth Limits per Traffic Type Command Fields

Name	Bytes.Bits	Value	Remarks
Reserved	18-23	0	Reserved. Must be set to 0.
Data Address high	24-17		Address of buffer.
Data Address low	28-31		

Table 7-170 describes format of the data buffer carrying additional command attributes.

Table 7-170. Configure Switching Component Bandwidth Limit per Traffic Type Command Buffer

Category	Byte/Bit	Field	Description
TC Valid	0.0-0.7	TC0-TC7 Valid	Valid bit per TC. Should list all TCs enabled for Switching Component. Must be consistent with the Physical Port configuration. At least one TC must be enabled.
Reserved0	1-15	0	Reserved. Must be set to 0.
TC Bandwidth Limit Credits	16-17	TC0 Bw Limit Credits	This field specifies a bandwidth limit per TC in Mb/s. Supported bandwidth limit range is 50Mb/s-40Gb/s, with increments of 50Mb/s. Bandwidth limit of 0Mb/s indicates that bandwidth limit is disabled Bandwidth limit should be provided for the valid TC handles only. One credit corresponds to bandwidth limit of 50Mb/s
	18-19	TC1 Bw Limit Credits	
	20-21	TC2 Bw Limit Credits	
	22-23	TC3 Bw Limit Credits	
	24-25	TC4 Bw Limit Credits	
	26-27	TC5 Bw Limit Credits	
	28-29	TC6 Bw Limit Credits	
	30-31	TC7 Bw Limit Credits	
TC Max Bandwidth Limit	32.0-32.2	TC0 Max Bw Limit	Max bandwidth limit indicates how much bandwidth limit credits can be accumulated due to inactivity. Single scheduling Quanta consumes multiple bandwidth limit credits. The XL710 allows accumulating of following discrete values of Quanta-worth credits: 0,1, 2, 4, 8,16 32 and 64 Quanta worth credits.
	32.3	Reserved	
	32.4-32.6	TC1 Max Bw Limit	
	32.7	Reserved	
	33.0-33.2	TC2 Max Bw Limit	
	33.3	Reserved	
	33.4-33.6	TC3 Max Bw Limit	
	33.7	Reserved	
	34.0-34.2	TC4 Max Bw Limit	
	34.3	Reserved	
	34.4-34.6	TC5 Max Bw Limit	
	34.7	Reserved	
	35.0-35.2	TC6 Max Bw Limit	
	35.3	Reserved	
	35.4-35.6	TC7 Max Bw Limit	
	35.7	Reserved	
Reserved2	36-63	0	Reserved. Must be set to 0.



7.8.4.14 Configure Switching Component Bandwidth Allocation per Traffic Type

This command allows software to specify Traffic Classes enabled for Switching Component, and configure relative bandwidth allocation of Switching Components within each Traffic Class.

This command can be used for Switching Components configured in ETS-Based mode only. Otherwise command would be ignored, and error reported.

To configure Switching Component bandwidth allocation per Traffic Type, software should provide complete relative bandwidth allocation for all Traffic Classes enabled for the Switching Component. In MFP environment, when different Switching Components sharing same Physical Port are owned by different Physical Functions, software is allowed to provide an Absolute Bandwidth Allocation Credits, and firmware is responsible to translate those to the relative credits.

Software is allowed to change bandwidth allocation per traffic type and a traffic classes enabled for switching component that has a switching component bandwidth limit enabled. It is responsibility of firmware to adjust hardware configuration to reflect this change.

Under MFP, EMP FW will ignore TC Enable BW parameters of the AQC. Instead, EMP FW will retrieve the parameters of TC enable and BW configuration per PF from MFP Switch configuration in Alternate RAM data and UP to TC mapping.

If TC re-configuration is needed while the VSI is suspended then new generated TC nodes of this VSI will be suspended too.

This is an Indirect Admin Queue Command. Software should provide a buffer that would be used both to provide additional command attributes and for the command completion.

Table 7-171 describes the command format and defines a command specific fields.

Table 7-171. Configure Switching Component Bandwidth Allocation per Traffic Type Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x0417	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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Switching Component SEID	16-17		Unique identifier of the VSI
Reserved1	18-23	0	Reserved. Must be set to 0.
Data Address high	24-27		Address of buffer. This buffer is used both to convey configuration to firmware, and provide software with a list of TC Handles.
Data Address low	28-31		

Table 7-158 describes format of the command buffer and defines command attributes.

Table 7-172. Configure Switching Component Bandwidth Allocation per Traffic Type Command Buffer

Category	Byte/Bit	Field	Description
TC Valid	0.0-0.7	TC0-TC7 Valid	Valid bit per TC. Should list all TCs enabled for Switching Component. Must be consistent with the Physical Port configuration. At least one TC must be enabled.
Reserved2	1-3.6	0	Reserved. Must be set to 0.
Absolute Credits	3.7	Absolute Credits Enable	If set indicates that credits provided are absolute credits, and not relative to the bandwidth allocated to other switching components on the same TC. Software should avoid use of absolute credits.
TC Bandwidth Share Credits	4	TC0 Bandwidth Credits	Relative Switching Component credits within same TC with respect to other Switching Components or VSIs enabled and connected to s-channels on the same Physical Port Valid range of credits is 1-127 credits, in increments of the single credit. 127 - indicates infinite credits
	5	TC1 Bandwidth Credits	
	6	TC2 Bandwidth Credits	
	7	TC3 Bandwidth Credits	
	8	TC4 Bandwidth Credits	
	9	TC5 Bandwidth Credits	
	10	TC6 Bandwidth Credits	
	11	TC7 Bandwidth Credits	
Reserved4	12-32	0	Reserved. Must be set to 0.

7.8.4.15 Query VSI Bandwidth Configuration

This command allows software retrieve current bandwidth configuration of the specified VSI of the specified switching component.

This is an Indirect Admin Queue command. Software should provide a buffer for command completion. Command completion carries VSI bandwidth configuration attributes, defined in Table 7-174.

DCB flow might require VEB configuration changing followed by VSI configuration changing. calling to “Query VSI Bandwidth Configuration” between the above two steps will be responded with Error code “EBUSY (12)”

Table 7-173 describes a command format, and defines a command specific fields.

Table 7-173. Query VSI Bandwidth Configuration Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x0408	Command opcode
Datalen	4-5		Length of response buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware. This should carry a status and an indication whether ETS is enabled or not.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command

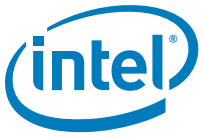


Table 7-173. Query VSI Bandwidth Configuration Command Fields

Name	Bytes.Bits	Value	Remarks
VSI SEID	16-17		Unique identifier of the VSI
Reserved	18-23	0	Reserved. Must be set to 0.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

Table 7-174 describes format of the completion buffer, and defines completion attributes. In addition to the bandwidth characteristics of VSI, this completion carries UP Handles and QS Handles of UPs and Queue Sets allocated for VSI.

Table 7-174. Query VSI Configuration Completion Buffer

Category	Byte/Bit	Field	Description
TC Valid	0.0-0.7	TC0-TC7 Valid	Valid bit per TC.
TC Suspended	1.0-1.7	TC0-TC7 Suspended	Marks per TC if this TC transmission is suspended.
Reserved0	2-15	0	Reserved. Must be set to 0.
QueueSet Handles	16-17	QS0 Handle	QueueSet handles of configured Queue Sets. Invalid handles will carry value of 0xffff. Order of Queue Set Handles matches order of TCs (0-7).
	18-19	QS1 Handle	
	20-21	QS2 Handle	
	22-23	QS3 Handle	
	24-25	QS4 Handle	
	26-27	QS5 Handle	
	28-29	QS6 Handle	
	30-31	QS7 Handle	
Reserved1	32-35	0	Reserved. Must be set to 0.
Bandwidth Limit	36-37		Bandwidth limit configured for the VSI in Mb/s. 0 indicates that bandwidth limit was not configured for the VSI One credit corresponds to bandwidth limit of 50Mb/s
Reserved2	38-39	0	Reserved. Must be set to 0.
Bandwidth Limit Max	40.0-40.2		Max bandwidth limit indicates how much bandwidth limit credits can be accumulated due to inactivity. Single scheduling Quanta consumes multiple bandwidth limit credits. The XL710 allows accumulating of following discrete values of Quanta-worth credits: 0,1, 2, and 4 Quanta worth credits.
Reserved3	40.3-63	0	Reserved. Must be set to 0.

7.8.4.16 Query VSI Bandwidth Configuration per Traffic Type

This command allows software to retrieve a bandwidth configuration of the specified VSI within each Traffic Class.

This command can be used for VSI configured in ETS-Based mode only. Otherwise command would be ignored, and error reported.

If Physical Port ETS is configured to single level ETS, and defines bandwidth distribution between Traffic Classes only. Software can assume one-to-one mapping of User Priorities to Traffic Classes from the bandwidth allocation perspective, and use this command to retrieve a bandwidth allocation for VSI within each Traffic Class.

This is an Indirect Admin Queue command. Software should provide a buffer for additional command attributes and command completion. Command completion carries VSI ETS/SLA configuration attributes, defined in Table 7-175.

DCB flow might require VEB configuration changing followed by VSI configuration changing. calling to "Query VSI Bandwidth Configuration" between the above two steps will be responded with Error code "EBUSY (12)"

Table 7-175 describes command format, and defines command specific fields.

Table 7-175. Query VSI Bandwidth Allocation per Traffic Type Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x040A	Command opcode
Datalen	4-5		Length of response buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware. This should carry a status and an indication whether ETS is enabled or not.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
VSI SEID	16-17		Unique identifier of the VSI
Reserved	18-23	0	Reserved. Must be set to 0.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

Table 7-176 describes format of completion buffer, and its attributes.

Table 7-176. Query VSI Bandwidth Allocation per Traffic Type Completion Buffer

Category	Byte/Bit	Field	Description
TC Valid	0.0-0.7	TC0-TC7 Valid	Valid bit per TC
TC Suspended	1.0-1.7	TC0-TC7 Suspended	Marks per TC if this TC transmission is suspended.
Reserved1	2-3	0	Reserved. Must be set to 0.

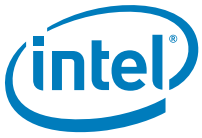


Table 7-176. Query VSI Bandwidth Allocation per Traffic Type Completion Buffer

Category	Byte/Bit	Field	Description
TC Bandwidth Share Credits	4	TC0 Bandwidth Credits	Relative VSI credits within same TC with respect to other VSIs enabled on the same Switching Component Valid range of credits is 1-127 credits, in increments of the single credit. 127 - indicates infinite credits
	5	TC1 Bandwidth Credits	
	6	TC2 Bandwidth Credits	
	7	TC3 Bandwidth Credits	
	8	TC4 Bandwidth Credits	
	9	TC5 Bandwidth Credits	
	10	TC6 Bandwidth Credits	
	11	TC7 Bandwidth Credits	
TC Bandwidth Limit Credits	12-13	TC0 Bw Limit Credits	This fiend specifies a bandwidth limit per TC in Mb/s. Supported bandwidth limit range is 50Mb/s-40Gb/s, with increments of 50Mb/s. Bandwidth limit of 0Mb/s indicates that bandwidth limit is disabled. Bandwidth limit should be provided for the valid handles only. One credit corresponds to bandwidth limit of 50Mb/s
	14-15	TC1 Bw Limit Credits	
	16-17	TC2 Bw Limit Credits	
	18-19	TC3 Bw Limit Credits	
	20-21	TC4 Bw Limit Credits	
	22-23	TC5 Bw Limit Credits	
	24-25	TC6 Bw Limit Credits	
	26-27	TC7 Bw Limit Credits	
TC Max Bandwidth Limit	28.0-28.2	TC0 Max Bw Limit	Max bandwidth limit indicates how much bandwidth limit credits can be accumulated due to inactivity. Single scheduling Quanta consumes multiple bandwidth limit credits. The XL710 allows accumulating of following discrete values of Quanta-worth credits: 0,1, 2, 4, 8,16 32 and 64 Quanta worth credits.
	28.3	Reserved	
	28.4-28.6	TC1 Max Bw Limit	
	28.7	Reserved	
	29.0-29.2	TC2 Max Bw Limit	
	29.3	Reserved	
	29.4-29.6	TC3 Max Bw Limit	
	29.7	Reserved	
	30.0-30.2	TC4 Max Bw Limit	
	30.3	Reserved	
	30.4-30.6	TC5 Max Bw Limit	
	30.7	Reserved	
	31.0-31.2	TC6 Max Bw Limit	
	31.3	Reserved	
	31.4-31.6	TC7 Max Bw Limit	
	31.7	Reserved	

7.8.4.17 Query Switching Component Configuration

This command allows software retrieve current bandwidth management configuration of the specified switching component.

This is an Indirect Admin Queue command. Software should provide a buffer for command completion. Command completion carries switching component bandwidth configuration attributes, defined in Table 7-178.

Table 7-177 describes a command format, and defines a command specific fields.

Result of this command is returned as a completion to the Admin Queue command.

Table 7-177. Query Switching Component Configuration Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x0418	Command opcode
Datalen	4-5		Length of response buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware. This should carry a status and an indication whether ETS is enabled or not.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Switching Component SEID	16-17		Unique identifier of the switching component
Reserved	18-23	0	Reserved. Must be set to 0.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

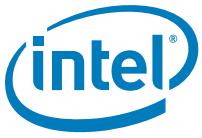
Table 7-178 describes format of completion buffer and its attributes.

Table 7-178. Query Switching Component Configuration Completion Buffer

Category	Byte/Bit	Field	Description
TC Valid	0.0-0.7	TC0-TC7 Valid	Valid bit per TC
Reserved1	1-31	0	Reserved. Must be set to 0.
Reserved1	32-35	0	Reserved. Must be set to 0.
Bandwidth Limit	36-37		Bandwidth limit configured for the port in Mb/s. 0 indicates that bandwidth limit was not configured for the Switching Component One credit corresponds to bandwidth limit of 50Mb/s
Reserved2	38-39	0	Reserved. Must be set to 0.
Bandwidth Limit Max	40.0-40.2		Max bandwidth limit indicates how much bandwidth limit credits can be accumulated due to inactivity. Single scheduling Quanta consumes multiple bandwidth limit credits. The XL710 allows accumulating of following discrete values of Quanta-worth credits: 0,1, 2, and 4 Quanta worth credits.
Reserved3	40.3-63	0	Reserved. Must be set to 0.

7.8.4.18 Query Physical Port ETS Configuration

This command allows software to retrieve ETS configuration of the specified switching component or VSI directly connected to the Physical Port.



This is an Indirect Admin Queue command. Software should provide a buffer for additional command attributes and command completion. Command completion carries Switching Component ETS/SLA configuration attributes, defined in Table 7-180.

This command can be used only for Switching Components directly connected to the physical port configured to ETS-Based Scheduler configuration. Otherwise, command is ignored, and error is reported.

Table 7-179 describes command format, and defines command specific fields.

Table 7-179. Query Physical Port ETS Configuration Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x0419	Command opcode
Datalen	4-5		Length of response buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware. This should carry a status and an indication whether ETS is enabled or not.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Physical Port SEID	16-17		Unique identifier of the physical port
Reserved	18-23	0	Reserved. Must be set to 0.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

Table 7-180 describes format of completion buffer, and its attributes.

Table 7-180. Query Physical Port ETS/SLA Completion Buffer

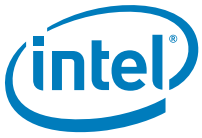
Category	Byte/Bit	Field	Description
Reserved1	0-3		Reserved to match VSI configuration format
TC Valid	4.0-4.7	TC0-TC7 Valid	Valid bit per TC This field is valid for ETS enable operation only. For ETS Modify operation TC Handles must be used instead Upto 8 TCs can be valid at any time.
TC Strict Priority	6.0-6.7	TC0-TC7 Strict Priority Flag	Strict priority flag per TC. If set then TC should be arbitrated based on its priority. If software decides to configure one or more TCs to be a strict priority TC, then TC with higher TC number has a higher priority. Configuring TCs to the strict priority may be useful for SLA configuration.

Table 7-180. Query Physical Port ETS/SLA Completion Buffer

Category	Byte/Bit	Field	Description
TC Bandwidth Share Credits	8	TC0 Bandwidth Credits	Relative credits per TC. Valid range of credits is 1-127 credits, in increments of the single credit. 127 - indicates infinite credits if TC is configured to strict priority
	9	TC1 Bandwidth Credits	
	10	TC2 Bandwidth Credits	
	11	TC3 Bandwidth Credits	
	12	TC4 Bandwidth Credits	
	13	TC5 Bandwidth Credits	
	14	TC6 Bandwidth Credits	
	15	TC7 Bandwidth Credits	
TC Bandwidth Limit Credits	16-17	TC0 Bw Limit Credits	This field specifies a bandwidth limit per TC in Mb/s. Supported bandwidth limit range is 50Mb/s-40Gb/s, with increments of 50Mb/s. Bandwidth limit of 0Mb/s indicates that bandwidth limit is disabled Bandwidth limit should be provided for the valid TC handles only. One credit corresponds to bandwidth limit of 50Mb/s
	18-19	TC1 Bw Limit Credits	
	20-21	TC2 Bw Limit Credits	
	22-23	TC3 Bw Limit Credits	
	24-25	TC4 Bw Limit Credits	
	26-27	TC5 Bw Limit Credits	
	28-29	TC6 Bw Limit Credits	
	30-31	TC7 Bw Limit Credits	
TC Max Bandwidth Limit	32.0-32.2	TC0 Max Bw Limit	Max bandwidth limit indicates how much bandwidth limit credits can be accumulated due to inactivity. Single scheduling Quanta consumes multiple bandwidth limit credits. The XL710 allows accumulating of following discrete values of Quanta-worth credits: 0,1, 2, 4, 8,16 32 and 64 Quanta worth credits.
	32.3	Reserved	
	32.4-32.6	TC1 Max Bw Limit	
	32.7	Reserved	
	33.0-33.2	TC2 Max Bw Limit	
	33.3	Reserved	
	33.4-33.6	TC3 Max Bw Limit	
	33.7	Reserved	
	34.0-34.2	TC4 Max Bw Limit	
	34.3	Reserved	
	34.4-34.6	TC5 Max Bw Limit	
	34.7	Reserved	
	35.0-35.2	TC6 Max Bw Limit	
	35.3	Reserved	
	35.4-35.6	TC7 Max Bw Limit	
	35.7	Reserved	
Reserved3	36-67	0	Reserved to match VSI configuration format. Must be set to 0.

7.8.4.19 Query Switching Component Bandwidth Configuration per Traffic Type

This command allows software to retrieve a bandwidth configuration of the specified Switching Component within each Traffic Class.



This command can be used for Switching Component configured in ETS-Based mode only. Otherwise command would be ignored, and error reported.

This is an Indirect Admin Queue command. Software should provide a buffer for additional command attributes and command completion. Command completion carries VSI ETS/SLA configuration attributes, defined in Table 7-181.

Table 7-181 describes command format, and defines command specific fields.

Table 7-181. Query Switching Component Bandwidth Allocation per Traffic Type Command Fields

Name	Bytes.Bits	Value	Remarks
Flags	1:0	0	See Section 7.10.5 for details.
Opcode	2-3	0x041A	Command opcode
Datalen	4-5		Length of response buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware. This should carry a status and an indication whether ETS is enabled or not.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
SEID	16-17		Unique identifier of the VSI or switching component
Reserved	18-23	0	Reserved. Must be set to 0.
Data Address high	24-27		Address of buffer.
Data Address low	28-31		

Table 7-182 describes format of completion buffer, and its attributes.

Table 7-182. Query Switching Component Bandwidth Allocation per Traffic Type Completion Buffer

Category	Byte/Bit	Field	Description
TC Valid	0.0-0.7	TC0-TC7 Valid	Valid bit per TC
Reserved1	1-3.6	0	Reserved. Must be set to 0.
Absolute Credits	3.7	Absolute Credits Enable	If set indicates that credits provided are absolute credits, and not relative to the bandwidth allocated to other switching components on the same TC. Software should avoid use of absolute credits.
TC Bandwidth Share Credits	4	TC0 Bandwidth Credits	Relative Switching Component credits within same TC with respect to other Switching Components enabled on the same Physical Port Valid range of credits is 1-127 credits, in increments of the single credit. 127 - indicates infinite credits
	5	TC1 Bandwidth Credits	
	6	TC2 Bandwidth Credits	
	7	TC3 Bandwidth Credits	
	8	TC4 Bandwidth Credits	
	9	TC5 Bandwidth Credits	
	10	TC6 Bandwidth Credits	
	11	TC7 Bandwidth Credits	

Table 7-182. Query Switching Component Bandwidth Allocation per Traffic Type Completion Buffer

Category	Byte/Bit	Field	Description
TC Bandwidth Limit Credits	12-13	TC0 Bw Limit Credits	This fiend specifies a bandwidth limit per TC in Mb/s. Supported bandwidth limit range is 50Mb/s-40Gb/s, with increments of 50Mb/s. Bandwidth limit of 0Mb/s indicates that bandwidth limit is disabled. Bandwidth limit should be provided for the valid handles only. One credit corresponds to bandwidth limit of 50Mb/s
	14-15	TC1 Bw Limit Credits	
	16-17	TC2 Bw Limit Credits	
	18-19	TC3 Bw Limit Credits	
	20-21	TC4 Bw Limit Credits	
	22-23	TC5 Bw Limit Credits	
	24-25	TC6 Bw Limit Credits	
	26-27	TC7 Bw Limit Credits	
TC Max Bandwidth Limit	28.0-28.2	TC0 Max Bw Limit	Max bandwidth limit indicates how much bandwidth limit credits can be accumulated due to inactivity. Single scheduling Quanta consumes multiple bandwidth limit credits. The XL710 allows accumulating of following discrete values of Quanta-worth credits: 0,1, 2, 4, 8,16 32 and 64 Quanta worth credits.
	28.3	Reserved	
	28.4-28.6	TC1 Max Bw Limit	
	28.7	Reserved	
	29.0-29.2	TC2 Max Bw Limit	
	29.3	Reserved	
	29.4-29.6	TC3 Max Bw Limit	
	29.7	Reserved	
	30.0-30.2	TC4 Max Bw Limit	
	30.3	Reserved	
	30.4-30.6	TC5 Max Bw Limit	
	30.7	Reserved	
	31.0-31.2	TC6 Max Bw Limit	
	31.3	Reserved	
	31.4-31.6	TC7 Max Bw Limit	
	31.7	Reserved	

7.8.5 Scheduler Configuration Flows

This section describes use of AdminQ commands defined in Section 7.8.4 to configure scheduler. It is not intended to provide a complete coverage of all possible configuration flow, but rather cover a several main stream cases.

Table 7-183 list all possible configuration changes of the Internal Switch or bandwidth configuration that have an impact on the transmit scheduler configuration, along with AQ Commands that should be used.

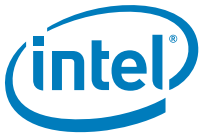


Table 7-183. Scheduler Configuration Summary

Configuration Change	Description	AQ Command
Default Port ETS Configuration	Default ETS configuration is ETS disabled, and the only TC enabled is TC0. See Section 7.8.5.3.	NA
Disable Physical Port	Disabling previously configured physical port does not directly impact scheduler configuration. Firmware will suspend disabled port, and software is responsible for resource deallocation using Delete Element AQ Command.	NA
Change Physical Port ETS Configuration	Driven by agent implementing DCBX protocol. Agent can run in firmware or in software. In case of software AQ command should be used to perform required ETS configuration change. See Section 7.8.5.3.	Enable, Disable, Modify Physical Port ETS. Section 7.8.4.10
Add Default VSI	Default VSI is allocated at switch initialization time. SFP configuration - one default VSI per Port. MFP configuration - one default VSI per Physical Function. Default VSI is configured with default bandwidth configuration (even bandwidth allocation, no bandwidth limits), and with single TC enabled - TC0. See Section 7.8.5.1.	NA
Add VSI	Regular VSI can be added directly connected to the Physical Port, or VEB. Scheduler configuration is done as a part of the Internal Switch configuration. New VSI is configured with a default bandwidth configuration (single bandwidth allocation credit, and no bandwidth limit enabled). New VSI can be configured with an initial set of TCs enabled for VSI. TCs enabled for VSI must be a subset of TCs enabled to the parent switching element or Physical Port. Section 7.8.5.3.	Add VSI Section 7.4.9.5.4.1,
Change TCs enabled for Default VSI or regular VSI.	Default or regular VSI TC configuration can be changed using Configure VSI Bandwidth Allocation per Traffic Type or Configure VSI Bandwidth Limit per Traffic Type AQ commands. In either case TCs enabled for the default VSI must be a subset of TCs enabled for the corresponding Physical Port. Update VSI CANNOT be used to change TC configuration of transmit scheduler.	Configure VSI Bandwidth Allocation per Traffic Type. Section 7.8.4.8. Configure VSI Bandwidth Limit per Traffic Type. Section 7.8.4.7
Change VSI bandwidth configuration	To change VSI bandwidth configuration software can use one of three AQ Commands: Configure VSI Bandwidth Limit - to enable/modify/disable bandwidth limit that applies to entire VSI (all enabled TCs) Configure VSI Bandwidth Limit per Traffic Type - to enable/ modify/disable bandwidth limit for individual TC enabled for VSI. Configure VSI Bandwidth Allocation per Traffic Type - to modify bandwidth allocation for the individual TC enabled for VSI.	Configure VSI Bandwidth Limit, Section 7.8.4.5 Configure VSI Bandwidth Allocation per Traffic Type. Section 7.8.4.8. Configure VSI Bandwidth Limit per Traffic Type. Section 7.8.4.7
Delete VSI	Software can remove VSI using Delete Element AQ Command. This includes removing all TC Nodes enabled for VSI.	Delete Element, Section 7.4.9.5.7.1
Add VEB to Default VSI	When VEB is added to the Default VSI, it is effectively inserted between default VSI and Physical Port. VEB can be configured with a set of TCs enabled for VEB. TCs enabled for VEB must include all TCs enabled for the Default VSI. VEB inherits bandwidth configuration of Default VSI. TCs that are enabled for VEB and not enabled for the Default VSI must be configured with a default bandwidth configuration. See Section 7.8.5.2.	Add VEB/PA. Section 7.4.9.5.5.1, Section 7.4.9.5.6.1
Add floating VEB	Not supported by current Scheduler Definition. Transmit scheduler must provide a Physical Port and TC along with the scheduling request to allow proper chip resource allocation by the pipeline. There are several ways how floating VEB can be enabled for the scheduler. All options result in modification of the Scheduler configuration tables.	Add VEB/PA. Section 7.4.9.5.5.1, Section 7.4.9.5.6.1

Table 7-183. Scheduler Configuration Summary

Configuration Change	Description	AQ Command
Change TCs enabled for VEB	To change number of TCs enabled for VEB software should use either Configure Switching Component Bandwidth Limit per Traffic Type, or Configure Bandwidth Allocation per Traffic Type AQ commands. TCs enabled for VEB must be a subset of TCs enabled for the parent switching component or Physical Port. Update VEB CANNOT be used to change TC configuration of transmit scheduler.	Configure Switching Component Bandwidth Limit per Traffic Type. Section 7.8.4.14 Configure Switching Component Bandwidth Allocation per Traffic Type. Section 7.8.4.13
Change VEB bandwidth configuration.	To change VEB bandwidth configuration software can use one of three AQ Commands: Configure Switching Component Bandwidth Limit - to enable/ modify/disable bandwidth limit that applies to entire VEB (all enabled TCs) Configure Switching Component Bandwidth Limit per Traffic Type - to enable/modify/disable bandwidth limit for individual TC enabled for VEB. Configure Switching Component Bandwidth Allocation per Traffic Type - to modify bandwidth allocation for the individual TC enabled for VEB.	Configure Switching Component Bandwidth Limit. Section 7.8.4.9. Configure Switching Component Bandwidth Limit per Traffic Type. Section 7.8.4.14 Configure Switching Component Bandwidth Allocation per Traffic Type. Section 7.8.4.13
Delete VEB	Software can remove VEB using Delete Element AQ Command. This includes removing all TC Nodes enabled for VEB. Software can remove VEB only after removing all VSIs attached to it after VEB creation. When VEB is removed, a default VSI is attached back to the S-Channel or Physical Port.	Delete Element, Section 7.4.9.5.7.1

7.8.5.1 Default VSI

Default VSI is automatically created for each Physical Function. In SFP environment single default VSI is allocated per Physical Port. In MFP environment multiple default VSIs can be created per Physical Port - one per Physical Function. Default VSI is always associated with TC0. Default VSI owns all bandwidth allocated for the Physical Port or a Physical Function. Software can change TCs enabled for the Default VSI and bandwidth allocation per Traffic Type using AQ command described in Section 7.8.4.8.

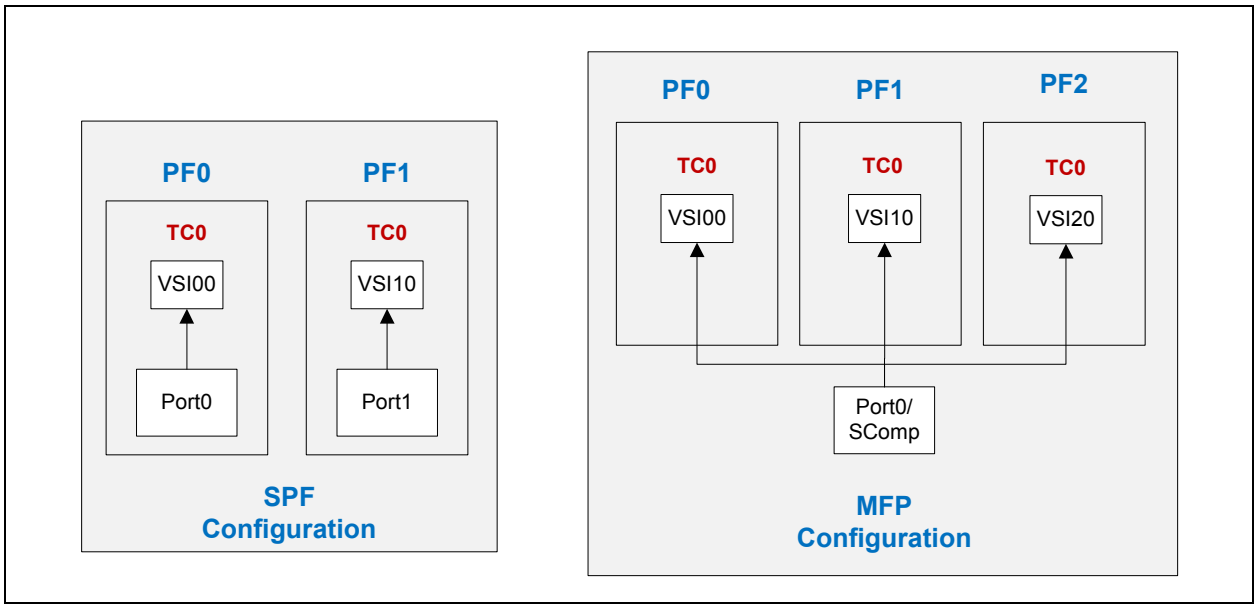


Figure 7-81. Default VSI Configuration

Figure shows an example of the default VSI configuration in SFP and MFP modes.

Firmware allocates a Queue Set for the each default VSI. Software can retrieve a Queue Set Handle using Query VSI command. There is no additional configuration of default VSI is required, and once respective Queue Set handle is associated with Transmit Queue, that queue can be scheduled for transmission.

ETS is disabled for the default VSI. Software can enable ETS configuration for the Physical Port and then configure ETS for VSI. See Section 7.8.5.3.

Software can instantiate a Switching Component using Add VEB, see Section 7.8.5.2.

7.8.5.2 Adding VEB/PA

SComp is effectively replacing a Physical Port in the Transmit Scheduler configuration hierarchy. All attributes configured for the Port automatically apply for the SComp (e.g. ETS configuration, etc.).

VEB/PA added to the switching hierarchy can be inserted between a default VSI and a Physical Port or an SComp, or can be instantiated using its own S-Channel.

Regular VEB, attached to physical port, always inserted between Port or VSI. If VSI is not a default VSI automatically allocated by firmware for each physical function or physical port, software is required to allocate VSI prior to adding VEB.

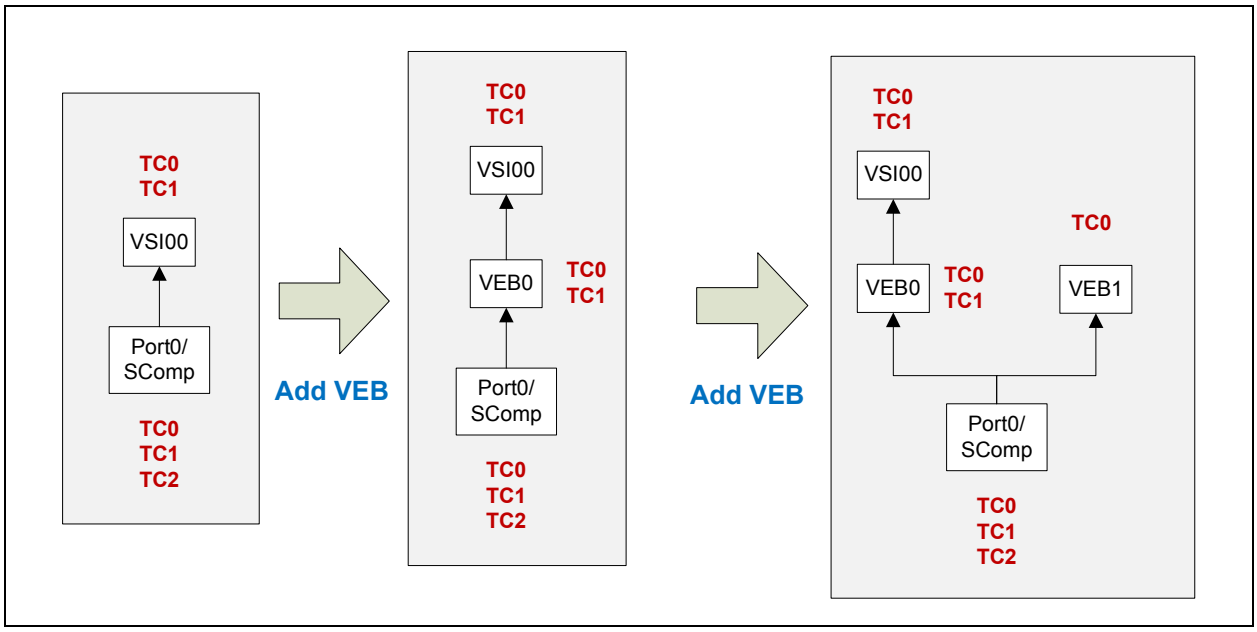


Figure 7-82. Adding VEB/PA

Figure 7-82 shows a transition of adding a VEB/PA Switching Component.

First step shows insertion of the VEB between Physical Port and a default VSI (VSI00). In this example Port has an ETS enabled, and a default VSI is configured with multiple Traffic Classes. Inserted VEB inherits VSIs configuration (e.g. Traffic Classes enabled for VSI, and bandwidth management attributes). VEB may have more traffic classes enabled than VSI. Software can change Traffic Classes enabled for VEB and relative bandwidth allocation within each Traffic Class using AQ command described in Section 7.8.4.14.

Insertion of VEB between default VSI and S-Channel does not impact Queue Set allocated for VSI. Transmit Queues, if any, previously associated with Queue Sets allocated for Default VSI will remain operational.

Second step shows a new VSI directly attached to the S-Channel. Software is required to add VSI prior to adding a new VEB. In the example shown on the diagram, new VSI has TC0 enabled. Software can specify TCs enabled for VSI using a TC enable bitmap in Add VSI AQ Command, or can change it later using Configure VSI Bandwidth Allocation per Traffic Type, or Configure VSI Bandwidth Limit per Traffic Type AQ Commands. Upon allocation of VSI, firmware allocates a Queue Set for each TC enabled for VSI, and provides it in the AQ Command completion.

Third step shows a second VEB/PA inserted between VSI10 (VEB1). Process of creation and bandwidth configuration is similar to one described above for the VEB0.

7.8.5.3 Adding VSI

In addition to the default VSI created for each Port in SFP mode and for each PF in MFP mode, software can request adding VSI to the previously allocated switching component. By default VSI is allocated with single TC enabled - TC0. Software can configure TCs enabled for VSI either using a bitmask in Add VSI AQ command, or using scheduler configuration AQ command described in Section 7.8.4.8.

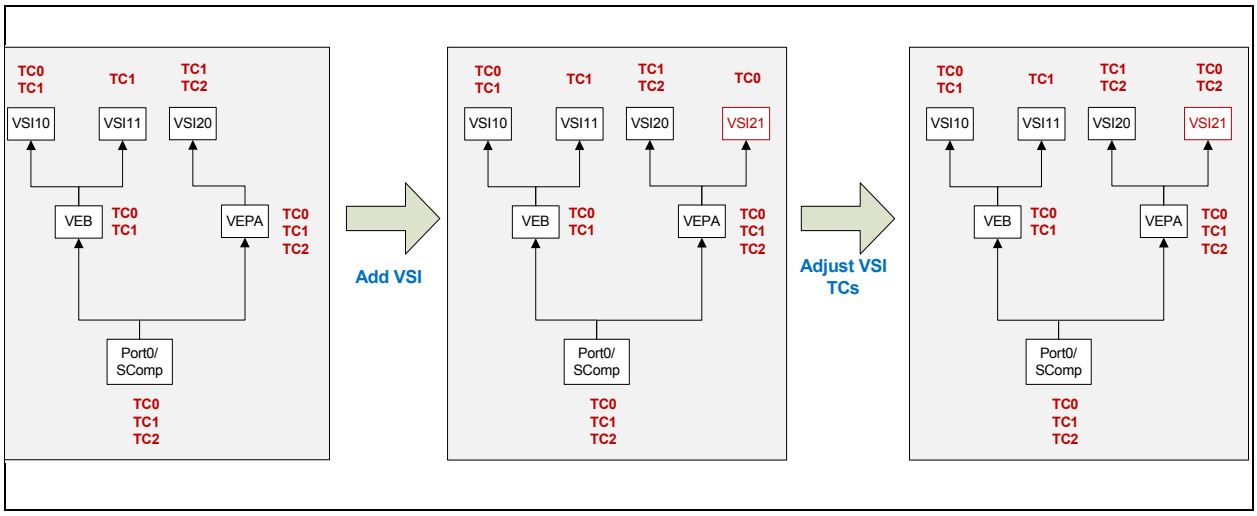


Figure 7-83. Add VSI

Each allocated VSI is allocated a Queue Set per enabled TC. Queue Set handles are returned in completion of Add VSI, or AQ command described in Section 7.8.4.8.

7.8.5.4 Bandwidth Limiting

Each allocated switching component and VSI may have a bandwidth limit configured. The XL710 supports two kinds of bandwidth limits - global, including traffic generated on all Traffic Classes enabled for VSI or Switching Component, and per Traffic Type bandwidth limit. Global bandwidth limits can be enabled using AQ commands described in Section 7.8.4.9 and Section 7.8.4.5. Per Traffic Class bandwidth limits can be enabled using AQ commands described in Section 7.8.4.7.

Bandwidth limits are be default disabled. Software can enable only one kind of bandwidth limit to each Switching Component and VSI.

Software does not have to modify relative bandwidth allocation for the switching components and VSIs to enable bandwidth limits. The default (even) bandwidth allocation can be used, and bandwidth would be equally distributed between switching components and VSIs as long as they have not reached configured bandwidth limit.

7.8.5.5 Bandwidth Distribution Ownership

7.8.5.5.1 SFP Mode

In SFP configuration one Physical Function owns all resources allocated to the Physical Port.

ETS configuration of the Physical Port by default would be performed by firmware based on data provided by DCBX agent running in firmware. Software can take over the role of DCBX agent, and then it will control ETS configuration of the Physical Port as well. Refer to Section 7.7.3 for details on DCBX ownership transfer. In either case, allocation or bandwidth management of the Switching Components and VSIs is completely owned by software, and performed using AQ commands described in Section 7.8.4.

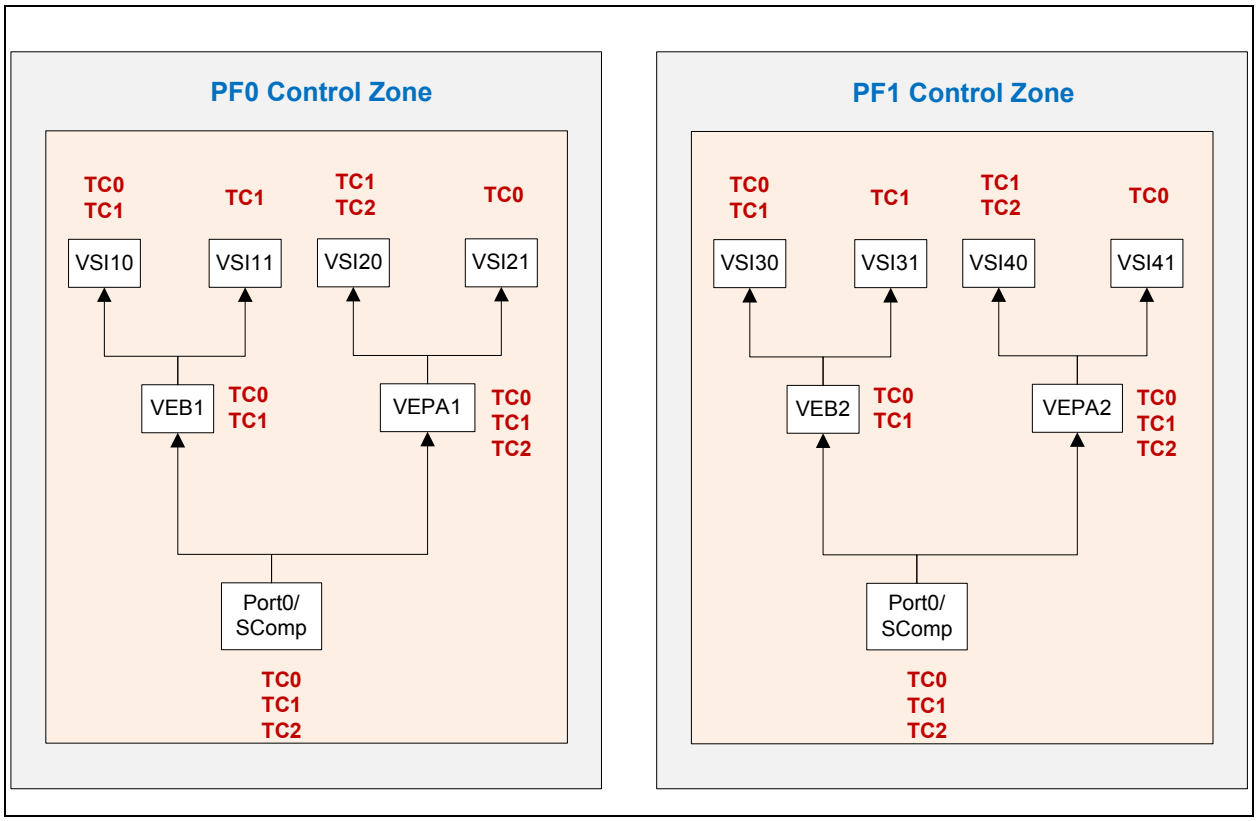


Figure 7-84. Software Control Zone in SFP Mode

Bandwidth allocation should be done using relative credits only. Physical Function driver should have all information available to manage relative credits.

7.8.5.5.2 MFP Mode

In MFP configuration multiple Physical Functions share same Physical Port. Allocation of resources between Physical Functions must be done by centralized management. In some configurations this management software will communicate directly with firmware via side-band interfaces or by wire. In other configurations, management can communicate with Physical Function software and provide it with information regarding allocated resources. In either case, Physical Function software does not control resources allocated for that Physical Function, but does own a full control over distribution of the allocated resource within Physical Function.

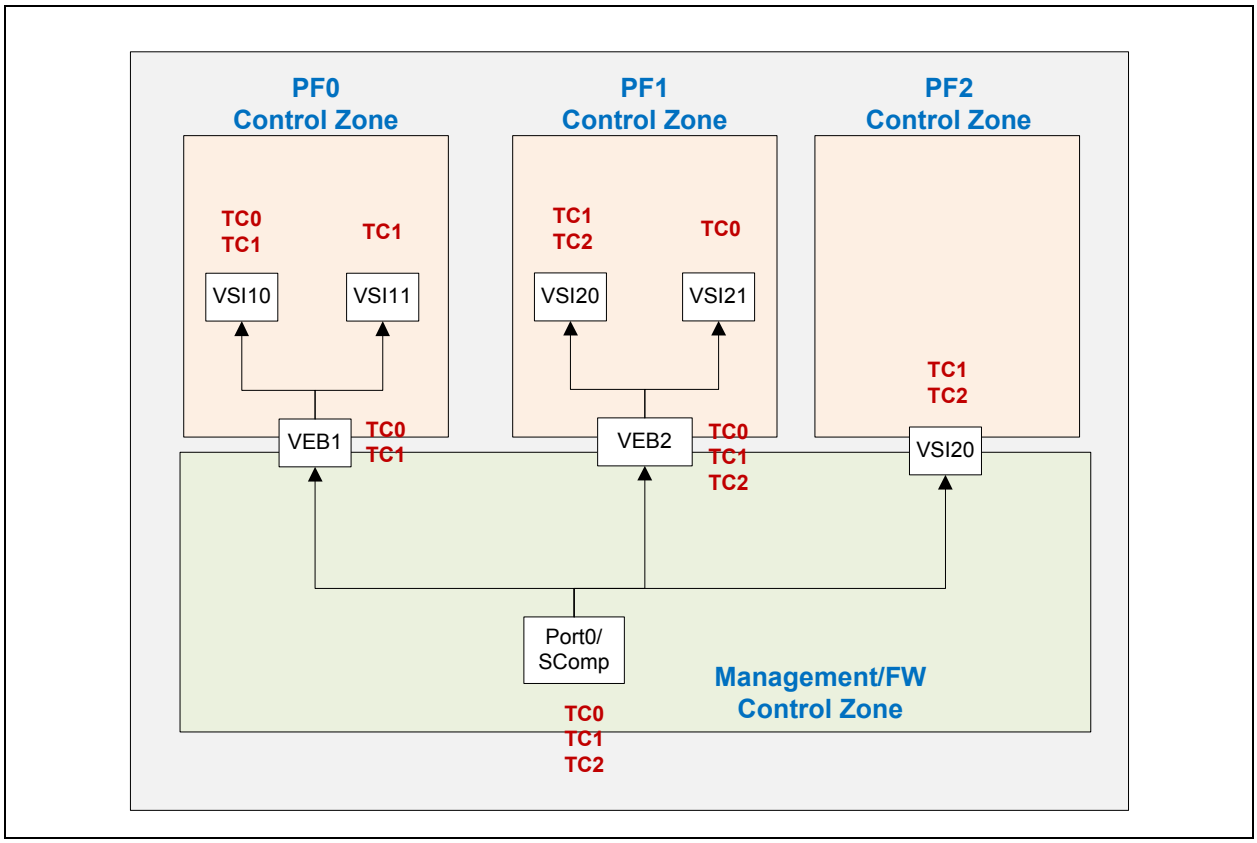


Figure 7-85. Software Control Zone in MFP Mode

Figure 7-85 above shows an example of MFP system. Each one of VEBs and VSIs can be configured with relative bandwidth allocation per enabled TC and bandwidth limits. Usually this configuration would be done directly by firmware based on instructions from the system management software.

Bandwidth distribution between VSIs allocated within each Physical Function is controlled by Physical Function driver.

If system management software communicates with Physical Function driver, and conveys resource allocation for the Physical Function, software can use AQ command described in Section 7.8.4.14 to configure bandwidth allocation for the Switching Component per enabled Traffic Class or use AQ commands described in Section 7.8.4.9 and Section 7.8.4.13 to enable bandwidth limit. Since Physical Function driver might not be able to coordinate its configuration with other Physical Functions, it is allowed to provide an absolute bandwidth allocation credits, and have firmware translate those to the relative bandwidth credits used to program hardware configuration tables.

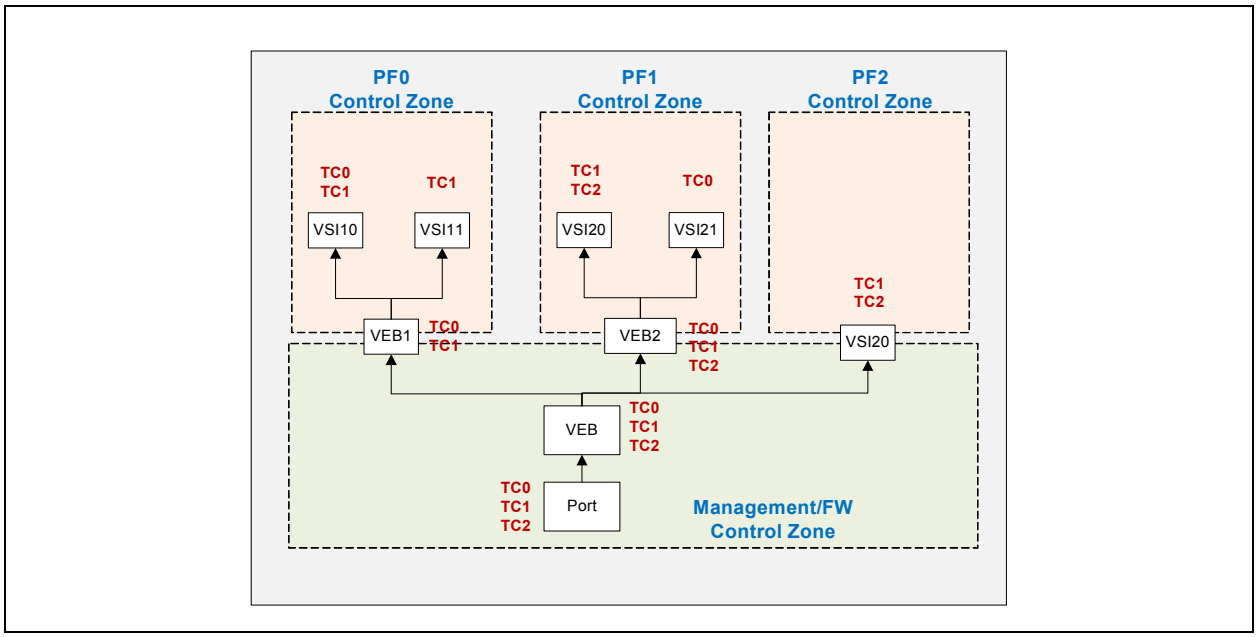


Figure 7-86. PF0-PF2 control zone

7.8.5.6 Enabling/Changing Port DCB Configuration

ETS configuration of Physical Port can be enabled or modified at any point. This is not expected to be a frequent event, and therefore transition period to adjust scheduler configuration to reflect the change is acceptable.

At least one TC must be enabled. If DCB is disabled for the Port, TC0 should be used.

In the default flow, firmware negotiates ETS configuration of the Physical Port, and configures TCs enabled by ETS for the Physical Port. Default VSI, previously created for the Port or Physical Functions associated with the Port, remains associated with TC0 only. Software should use AQ command described in Section 7.8.4.8 to modify TCs enabled for VSI and a relative bandwidth allocation of VSI within each enabled TCs.

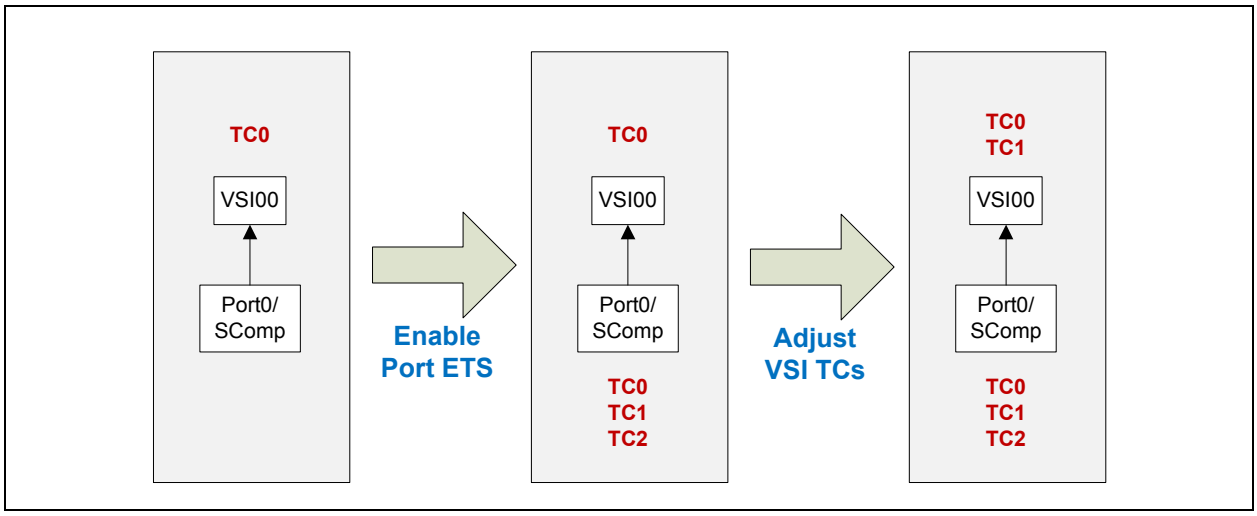


Figure 7-87. Enable Physical Port ETS

If software instantiated additional VSIs or Switching Components prior to ETS is enabled for the port, or if ETS configuration has been changed, firmware is responsible to update ETS configuration of Physical Port, and add or remove TCs. All previously allocated VSIs and Switching Components remain associated with TCs previously configured TCs. Software should use AQ commands described in Section 7.8.4.8 and Section 7.8.4.14 to adjust TCs enabled for VSIs and Switching Components, and modify bandwidth allocation per Traffic Type as required.

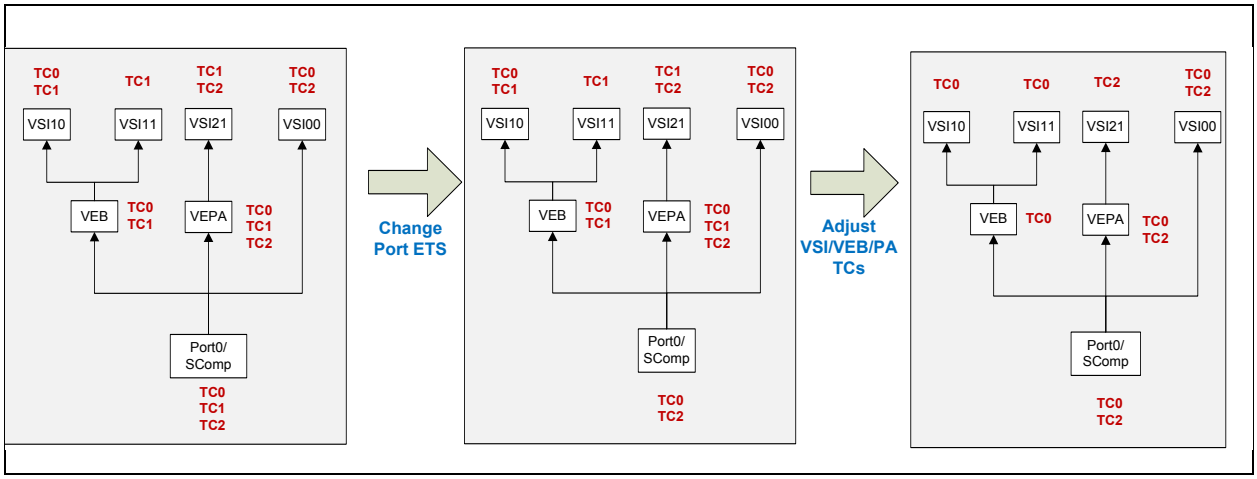


Figure 7-88. Change Physical Port ETS

Scheduler will start scheduling requests for the new Traffic Classes added to the Physical Port configuration, only after this Traffic Class is enabled for one or more VSIs. Adding more TCs to VSI will lead to allocation of new Queue Sets, which in turn must be associated with Transmit Queues.

Once Traffic Class is removed from the Physical Port configuration, transmit Scheduler suspends scheduling traffic for all Queue Sets associated with that Traffic Class and then notifies SW with LLDP MIB Change Event. To resume scheduling traffic on Transmit Queues associated with such Queue Sets, software will need to adjust TC configuration of VSIs and Switching Components, and reassign affected Transmit Queues to Queue Sets associated with other Traffic Classes.

Port DCB configuration can be directly changed by firmware as a result of the DCBX exchange, this assumes a DCBX agent running in firmware (which is a default configuration). Or it can be configured by software using AQ command described in Section 7.8.4.10 if software took over duty of running DCBX agent. Transmit scheduler configuration does not depend on the location of the DCBX agent.

ETS reconfiguration of VSI may move TX queues between TCs. Running this configuration flow while TX pipe is loaded with TX packets belong to this queue (either in the TCB or in the TPB) might cause out of order completion for this TX queue.

To prevent this error case, it is required to verify that TX pipe is drained from any TX packet belonging to re-configured queue or Qset. There are two trivial ways SW can verify this draining.

1. PF can stop feeding the TX ring and wait till all pending work is completed.
2. PF can disable the TX queue. This provides faster draining mechanism but flushes all pending work.

EMP Firmware provides a service which suspends port's Qsets and drains the TX pipe "on the fly". This "Port draining" service is used as part of DCBX change (see Section 7.8.4.11 for more details).

The general flow DCBX exchange is: If ETS setting of VSI's is required to be changed then EMP will drain first the TX pipe of the port (initiated by LLDP owner, done by EMP and TX scheduler). After the port is drained, FW will notify PF which will make those required ETS changes (Via TX scheduler AQC). After VSI's ETS setting is adjusted, PF will resume its Qsets using AQC described in Section 7.8.4.7 and in Section 7.8.4.8 .

Some other DCB parameters require TX pipe draining before re-configuration. DCBX owner (EMP Firmware or SW) will use TX scheduler draining service. More details below in Section 7.8.5.6.1.3.

Some of DCBX flows require change of Queue context configuration. In those cases, SW must disable the Queue first.

7.8.5.6.1 Tx Scheduler DCB control flows

7.8.5.6.1.1 General

DCBX and TX scheduler control flows were declared under some assumptions and decisions as followed:

- For DCBX exchange event, SW runs the similar flow under SFP or MFP
- Usually DCBX agent runs in the EMP. under SFP mode, SW might take LLDP agent ownership.
- While SW didn't take LLDP ownership, the difference between DCBX SW flows when running under SFP or MFP needs to be minimized.
- DCBX event might occur during first connectivity to the network as part of power up flow, as part of link down link up event, or in rare case, the switch had decided to change DCB setting.
- Three typical boot sequences are identified (DCBX runs at a different stage in each case):

  Case I - Typical MFP power on sequence

  a. EMP sets the initial scheduler configuration - One initial VSI per PF. Detailed flow in Section 7.8.5.6.1.2
  b. DCBX runs immediately when EMP initializes (when Link goes up). Detailed flow in Section 7.8.5.6.1.3
  c. MFP configuration runs
  d. After Alternate Ram Done a Global Reset is triggered and EMP reloads configuration

e. EMP sets the initial scheduler configuration including MFP initial configuration. Detailed flow in Section 7.8.5.6.1.2
f. SW boot. Detailed flow in Section 7.8.5.6.1.4

Case II - Typical SFP boot sequence or MFP system reboot after MFP settings already done

a. EMP sets the initial scheduler configuration - One initial VSI per PF. Detailed flow in Section 7.8.5.6.1.2
b. DCBX runs. Detailed flow in Section 7.8.5.6.1.3
c. SW boot. Detailed flow in Section 7.8.5.6.1.4

Case III - Late DCBX, DCBX parameters changed by the peer, or Link event.

a. EMP sets the initial scheduler configuration - One initial VSI per PF. Detailed flow in Section 7.8.5.6.1.2
b. SW boot. Detailed flow in Section 7.8.5.6.1.4
c. DCBX runs or makes changes in DCBX setting. Detailed flow in Section 7.8.5.6.1.3

- When link goes up either while power up flow or while system already runs, all DCBX MIBs are reset to their default values (All TC are in DROP policy, all UPs are mapped to TC#0 and TC#0 is the only active TC for this port). So, after link up, the DCBX flow runs twice, a) Reset MIBs to default. b) New DCBX exchange with the new link partner.

7.8.5.6.1.2 Reset/Init Flow

- EMP initializes Switch and Tx Scheduler tables.
- For SFP, EMP builds one initial VSI per active port.
- If MFP mode and Alternate RAM already loaded then
  - — EMP builds a Scomp per port and one initial VSI per active PF
  - — EMP reads Min and Max BW configuration of each PF from the Alternate Ram and set this in TX Scheduler.

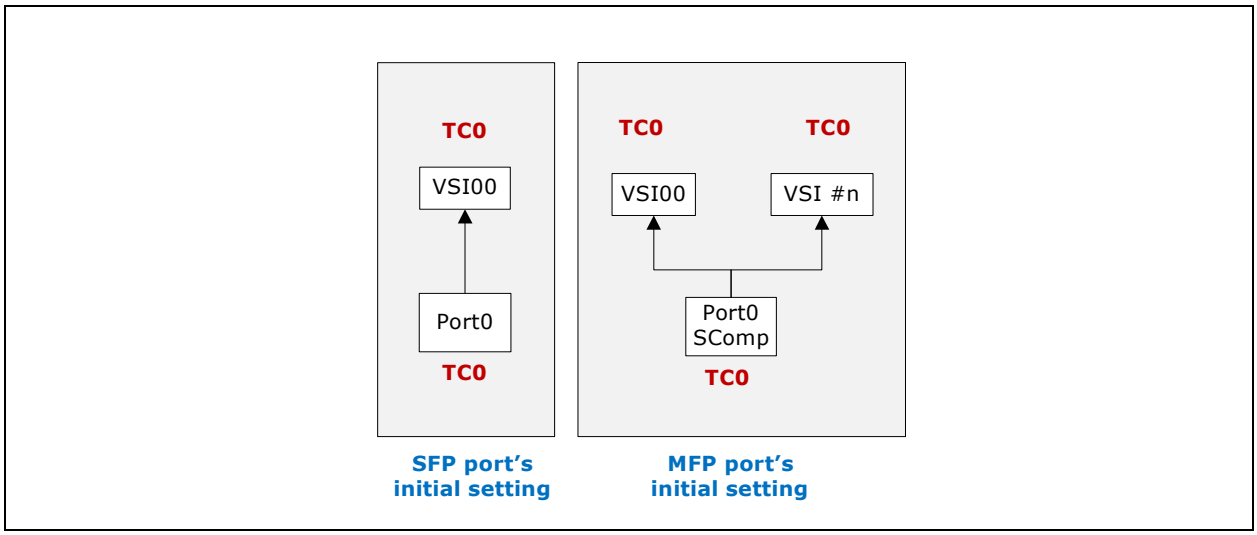


Figure 7-89. initial setting in SFP (one initial VSI/Port) and MFP (one initial VSI/PF)

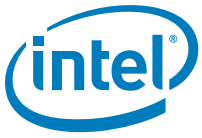
7.8.5.6.1.3 DCBX exchange

- When EMP owns LLDP
  - — IF port draining is required (change in Drop policy, UP to TC mapping or Port TC mapping)

    - EMP suspends all port's TX (all TCs - no credits)
    - EMP drains port's TX pipeline (wait till all pipe monitor counters are zeroed).
  - EMP re-configures needed DCB settings. (See Section 7.7.3.1.1 step 2 for details)
  - IF port is suspended then EMP resumes manageability TX traffic.
  - EMP re-configures TX scheduler
    - Port ETS setting (suspend the TC nodes of disabled TCs, Resume TC nodes of enabled TCs)
    - TC BW configuration
  - EMP notifies DCBX event to all PFs which registered for LLDP MIB notification.
    - This is done via posting LLDP MIB Change Event, after waking up the host if it was not in D0 state. (See "LLDP MIB Change" Event notification details in Section 7.12.5.2.3.3)
      - "LLDP MIB Change" Event notification includes "miscellaneous" field which mark to the PF if the MIB change involved "Port Draining" and if per TC pipe flushing was used.
    - PF which is not registered for LLDP event must periodically check for LLDP MIBs update status.

// The below is relevant only when SW is up and running (case III)

- When SW owns LLDP (applies only in SFP mode)
  - IF port draining is required, PF calls "Suspend Port's TX Traffic" AQC (See Section 7.8.4.11)
    - EMP suspends all port's TX (all TCs - no credits)
    - EMP stops manageability TX traffic.
    - EMP drains port's TX pipeline (wait till all pipe monitor counters are zeroed).
      -
    - EMP drains port's TX pipeline (wait till all pipe monitor counters are zeroed).
      -
  - PF re-configures needed DCB settings as needed. (See Section 7.7.3.1.1 step 2 for details)
  - PF instructs EMP to configure TX Scheduler with port's TC setting (via "Configure Physical Port ETS" AQC)
- PF computes the needed tree topology changes according the new UP to TC mapping and PFs' UP settings
- PF adjusts VEBs and VSIs per TC setting (in MFP, each PF does this for its resources)
  - "Done via "Configure VSI Bandwidth Limit per Traffic Type", "Configure VSI Bandwidth Allocation per Traffic Type", "Configure Switching Element Bandwidth Limit per Traffic Type" or "Configure Switching Element Bandwidth Allocation per Traffic Type"
    - In case the commands "Configure VSI Bandwidth Limit per Traffic Type", "Configure VSI Bandwidth Allocation per Traffic Type" are called and remap only one TC node, EMP will give this TC node its original Qset handle. This allows PF to skip the step of disabling and enabling all queues.
    - The commands "Configure VSI Bandwidth Limit per Traffic Type", "Configure VSI Bandwidth Allocation per Traffic Type" maybe used to expand VSI's TC setting. EMP will return Qset handle for each one of the enabled TCs including the new used TCs.
    - PF copies the new Qset handles to the contexts of the TX queues belonging to the new established Qsets. When a queue context needs change its Qset (moving between Qsets), SW must disable the queue, reconfigure and enable back.
    - Please see Figure 7-85 above for explanation on Control Zones in MFP. Under MFP, PFs TC setting, Min BW and Max BW configuration are derived from Alternate Ram MFP configuration data. When PF tries to configure TC or BW configuration of an entity which is in FW control zone, EMP Firmware will ignore the parameters provided in the command. Instead, EMP will compute the TC setting and BW configuration from Alternate Ram function's configuration. EMP will return back the actual TC and BW setting. Same function



call is used for both SFP and MFP although in MFP the parameters are not used. This is done for compatibility purposes (identical flow in SW under both SFP and MFP).

- After configuration is done, PF resumes its suspended TX traffic using AQC "Resume PF traffic" (see Section 7.8.4.12).

7.8.5.6.1.4 SW boot

- PF reads the DCBX MIBs (incl. UP-TC mapping)
  - In order to get consistent MIB data, PF is required to verify that LLDP owner is not processing LLDP message right now by polling “PRTDCB_GENS.DCBX_STATUS” till it will be set to DONE.
- Before making any change to VSI settings, PF reads initial configuration and configure all Qsets.
  - Tree topology reading done via "Get Switch Configuration" command (see Section 7.4.9.5.3.1).
  - For each VSI, PF needs to call "Query VSI Bandwidth Configuration" (see Section 7.8.4.15).
    - The response buffer includes:
    - A bitmap; Which TCs are enabled in this VSI
    - A Qset handle for each enabled TC.
    - A bitmap; Which of the Enabled TCs is suspended due to DCBX event.
    - PF must verify that a Qset is established for each enabled TC.
    - For each enabled TC, PF is required to copy the Qset handle value to all Queue belong to the Qset (before enabling the queue)
- PF determines whether to move/add any VSI to a different TC. Done via "Configure VSI Bandwidth Limit per Traffic Type", "Configure VSI Bandwidth Allocation per Traffic Type"
  - In case the commands "Configure VSI Bandwidth Limit per Traffic Type", "Configure VSI Bandwidth Allocation per Traffic Type" are called and move only one TC node, EMP will give this TC node, its original Qset handle. This allows PF to skip the step of disabling and enabling all queues.
    - PF resumes its suspended VSI's using AQC "Resume PF traffic" (see Section 7.8.4.12).

- Whenever PF needs to change TC setting of a VSI (not as a response to DCBX event, This VSI is NOT suspended), PF MUST drain all TX queues belonging to moved TCs (in the VSI) before changing its TC settings. (Changing any setting of a Qset while it has packets in the TX pipe might cause out of order completion).
- Any topology change is done via "Add VSI" or "Add VEB" command (Enabled TCs field) that defines the TCs for the VSI or VEB
  - EMP adds VEB, VSI, Qs accordingly
  - As a response to ADD VSI command, EMP provides the Qset handles of the enabled TCs.
  - For each enabled TC, PF is required to copy the Qset handle value to all Queue belong to the Qset

7.8.5.7 Transmit Scheduler Resource Allocation Control

Each Physical Function is configured with limited number of resources. The XL710 Scheduler allows flexible resource allocation, but due to limited number of Queue Set supported, it allows to allocated in average two Queue Sets per VSI.

Number of VSIs dedicated per Physical Function is a Switch Configuration parameter configured via NVRAM. The rest of available VSIs are shared on the First-Come-First-Serve bases.

Scheduler configuration firmware only controls resources allocated for PF, and allows Physical Function driver distribute those resources within PF.

Allocation of Queue Sets per PF is proportional to the allocation of VSIs, assuming average of two Queue Sets per VSI. Shared Queue Sets can be allocated to PFs on First-Come-First-Served bases, similar to VSIs.

7.8.5.8 Scheduler Configuration Schemes

ETS-Based scheme allows ETS configuration for the Switching Component or VSI directly connected to the Physical Port, and bandwidth allocation of other VSIs and Switching Components within each traffic type that can be either User Priority or Traffic Class enabled for that VSI or Switching Component, Section 7.8.2.1.

Software should not attempt to configure bandwidth management attributes that are not supported by configured scheme. Firmware will reject to perform invalid operation and will return an EPERM error described in Section 7.8.4.4.

7.8.5.8.1 ETS-Based Scheme: Relative Bandwidth Credits Calculation

ETS-based configuration allows XL710 Software configure relative bandwidth allocation for each Switching Component and VSI within each traffic type (User Priority or Traffic Class). Software should use AdminQ commands described in Section 7.8.4.8 and Section 7.8.4.14 respectively.

XL710 software should be able to configure ETS-based transmit scheduler based on

- An absolute bandwidth allocation (in GB/s or Mb/s) for VSIs and Switching Components within each Traffic Class or User Priority enabled for the respective VSI or Switching Component
- An absolute bandwidth allocation for VSIs and Switching Components, and ETS configuration for each VSI and Switching Component.

In either case, configuration provided by system management software will need to be translated to relative bandwidth allocation of VSIs and Switching Components within Traffic Class or User Priority expressed in relative bandwidth allocation credits.

Figure 7-90 shows an example of ETS-based scheme bandwidth configuration and process of relative bandwidth credits calculation.

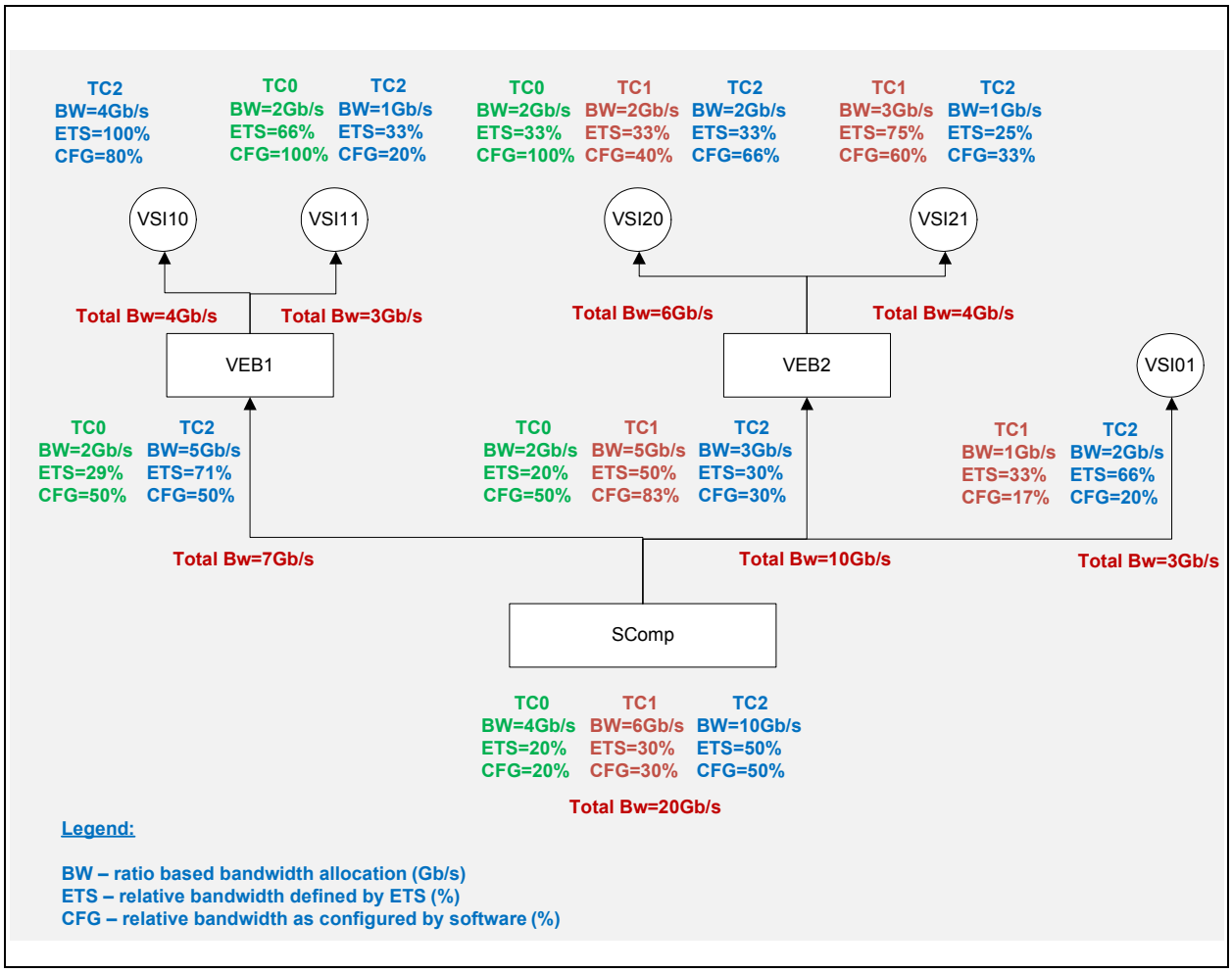


Figure 7-90. ETS-Based Scheme: Relative Bandwidth Calculation Example

Figure 7-90 shows a single port configuration with SComp. Two VEB switching components are connected to SComp (VEB1 and VEB2), along with VSI (VSI01) directly attached to SComp. Each VEB has two VSIs (VSI10 and VSI11, and VSI20 and VSI21 respectively).

This diagram for each Switching Component and VSI shows a per traffic class bandwidth allocation using 3 terms:

- ETS - relative bandwidth allocation per TC within VSI/Switching Component in percentage with respect to other TCs. can be provided by system software along with total bandwidth allocated for the Switching Component and VSI.
- CFG - calculated hierarchical relative bandwidth allocation of VSIs and Switching Components within each Traffic Class, in percentage with respect to other VSIs and Switching Components on the same hierarchy level.
- Total Bandwidth - Can be provided by system management software along with ETS configuration for the Switching Components and VSIs.

Note, ETS-Based configuration does not guarantee bandwidth allocation for VSIs and Switching Components in ETS-Based configuration, but it can use bandwidth allocation provided by system software to configure VSI and Switching Components relative bandwidth allocation within each Traffic Class or User Priority.

If Physical Port is configured to two level ETS (i.e. bandwidth is allocated for TCs, and then distributed between UPs within same TC), then VSI and Switching Component bandwidth is allocated per User Priority.

If Physical Port is configured to single level ETS (bandwidth distribution between TCs only), XL710 software can assume a logical one-to-one mapping of User Priorities to Traffic classes for the bandwidth distribution purpose, and use AdminQ commands described in Section 7.8.4.8 and Section 7.8.4.14 to configure VSi and Switching Component bandwidth allocation within Traffic Class.

In either case, relative credits calculation should be done following algorithm described below:

- If system software provided absolute bandwidth allocation per Traffic Class or User Priority
  - The XL710 software should calculate a relative bandwidth allocation with respect to the parent switching entity within each traffic type (TC or UP)
  - Translate relative bandwidth allocation to credits, minimizing number of credits allocated
- If system software provided an absolute bandwidth allocation for VSI or Switching Component and ETS
  - XL710 software should calculate an absolute bandwidth allocation per traffic type within VSI or Switching Component (which would be the same as an absolute bandwidth allocation of VSI or Switching Component within traffic type (TC or UP)).Calculate a relative bandwidth allocation with respect to the parent switching entity within each traffic type (TC or UP)
  - Translate relative bandwidth allocation to credits, minimizing number of credits allocated

Table 7-184 shows example of relative bandwidth credits calculation for the ETS-Based system configuration shown on Figure 7-90. This example assumes that system management software provided a total bandwidth allocated for each VSI and Switching Component, and ETS.

Table 7-184. ETS-Based Scheme-Relative Bandwidth Calculation Example

Switching Element Name	Parent Switching Element Name	Total BW	ETS (TCs)	ETS (%)	ETS (Gb/s)	Relative BW within TC	Relative Credits within TC	Comments
VEB1	SComp	7Gb/s	TC0	29%	2Gb/s	50%	1	VEB1 shares TC0 bandwidth with VEB2, 50% each, therefore 1 relative credit will correctly reflect a relative bandwidth allocation between VEB1 and VEB2 (1:1)
			TC2	71%	5Gb/s	50%	5	VEB1 shared TC2 bandwidth with VEB2 and VSI01, with ratio 5:3:2, which is reflected by relative credits
VEB2	SComp	10Gb/s	TC0	20%	2Gb/s	50%	1	VEB2 shares TC1 bandwidth with VSI01, relative ratio is 5:1, which is reflected in relative credits
			TC1	50%	5Gb/s	83%	5	
			TC2	30%	3Gb/s	30%	3	
VSI01	SComp	3Gb/s	TC1	33%	1Gb/s	17%	1	
			TC2	66%	2Gb/s	20%	2	

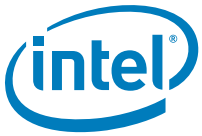


Table 7-184. ETS-Based Scheme-Relative Bandwidth Calculation Example

Switching Element Name	Parent Switching Element Name	Total BW	ETS (TCs)	ETS (%)	ETS (Gb/s)	Relative BW within TC	Relative Credits within TC	Comments
VSI10	VEB1	4Gb/s	TC2	100%	4Gb/s	80%	4	VSI10 shared TC2 bandwidth allocated for VEB1 with VSI11, with relative ratio 4:1, which is reflected in allocated relative credits
VSI11	VEB1	3Gb/s	TC0	66%	2Gb/s	100%	1	VSI11 consumes all TC0 bandwidth allocated for VEB1, and therefore can be configured with 1 relative credit
			TC2	33%	1Gb/s	20%	1	
VSI20	VEB2	6Gb/s	TC0	33%	2Gb/s	100%	1	VSI20 consumes all TC0 bandwidth allocated for VEB2, and therefore can be configured with 1 relative credit
			TC1	33%	2Gb/s	40%	2	VSI20 shared TC1 bandwidth allocated to VEB2 with VSI21 with relative ratio 2:3, which is reflected in relative credits
			TC2	33%	2Gb/s	66%	2	
VSI21	VEB2	4Gb/s	TC1	75%	3Gb/s	60%	3	VSI21 shared TC2 bandwidth allocated to VEB2 with VSI20 with relative ratio 1:2, which is reflected in relative credits
			TC2	25%	1Gb/s	33%	1	

Table cells with grey background show numbers calculated by XL710 software.

7.9 Host Memory Cache

The XL710 uses host memory as backing store for a number of context objects used to track queue state, FCoE DDP state. The Host Memory Cache (HMC) is the component responsible for managing the LAN, and FCoE 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 drivers operating environment and the number of PCI functions that are currently active. Resource profiles can be selected at driver initialization time.

7.9.1 Host Memory Usage

The HMC requires backing store for numerous data structures to be resident in host memory to perform its functions. Table 7-185 provides a list of the data structures and the amount of memory that needs to be allocated for each data structure. The HMC PCI Function Type 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, all LAN and FCoE objects for both PFs and VFs are located in the PF HMC object space. The resources can be sparsely populated. For example, if a function is allotted 512 QPs and only 8 are used, then only 4K of memory needs to be allocated not the entire memory for all 512 QPs. Some HMC object need to be fully populated at driver initialization such as FCoE Filters. See Section 8.1, Section 9.0 for more information on the HMC resource allocation policies for LAN, FCoE respectively.

Table 7-185. HMC Objects

HMC Object	HMC Object Location	Size (Bytes)	Max Quantity	Description
LAN Transmit Queue	PFs	128	1536 per device	The PF owns the objects for the associated VFs. LAN absolute queue numbers assignment to PFs are determined by the programming of the PFLAN_QALLOC registers. HMC object indexes match the LAN queue indexes. See section Section 8.2 for more details on LAN queue allocation.
LAN Receive Queue	PFs	32	1536 per device	The PF owns the objects for the associated VFs. See the previous description of LAN Transmit queues above for details on the relationship between absolute LAN QP numbers and HMC object indexes.
FCoE DDP	PFs	64	4096 per PCI Function	The PF owns the objects for the associated VFs. The PF and each VF assigned to the PF has 4096 objects allocated in HMC Private Memory Space.
FCoE Filters	PFs	64	36,864 per PCI Function	The PF owns the objects for the associated VFs. The PF and each VF assigned to the PF has 36,864 objects allocated in HMC Private Memory Space. The actual number of objects to be used is controlled by programming the PFQF_CTL_0 register.

In order to access (and cache in on-chip memory) the data structures defined in Table 7-185, the HMC uses the concept of private memory address space. The XL710 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). FPM objects for VF LAN and FCoE objects are located in the associated PFs FPM.

Figure 7-91 shows how the XL710 provides the address mapping between Private Memory and Host Physical Addresses. PM Address shown on the left side of the figure indicates XL710 Private Memory address from 0 to 8GB-1. The XL710 works with PM address space internally which is converted to Host Physical Addresses in order to access host memory.

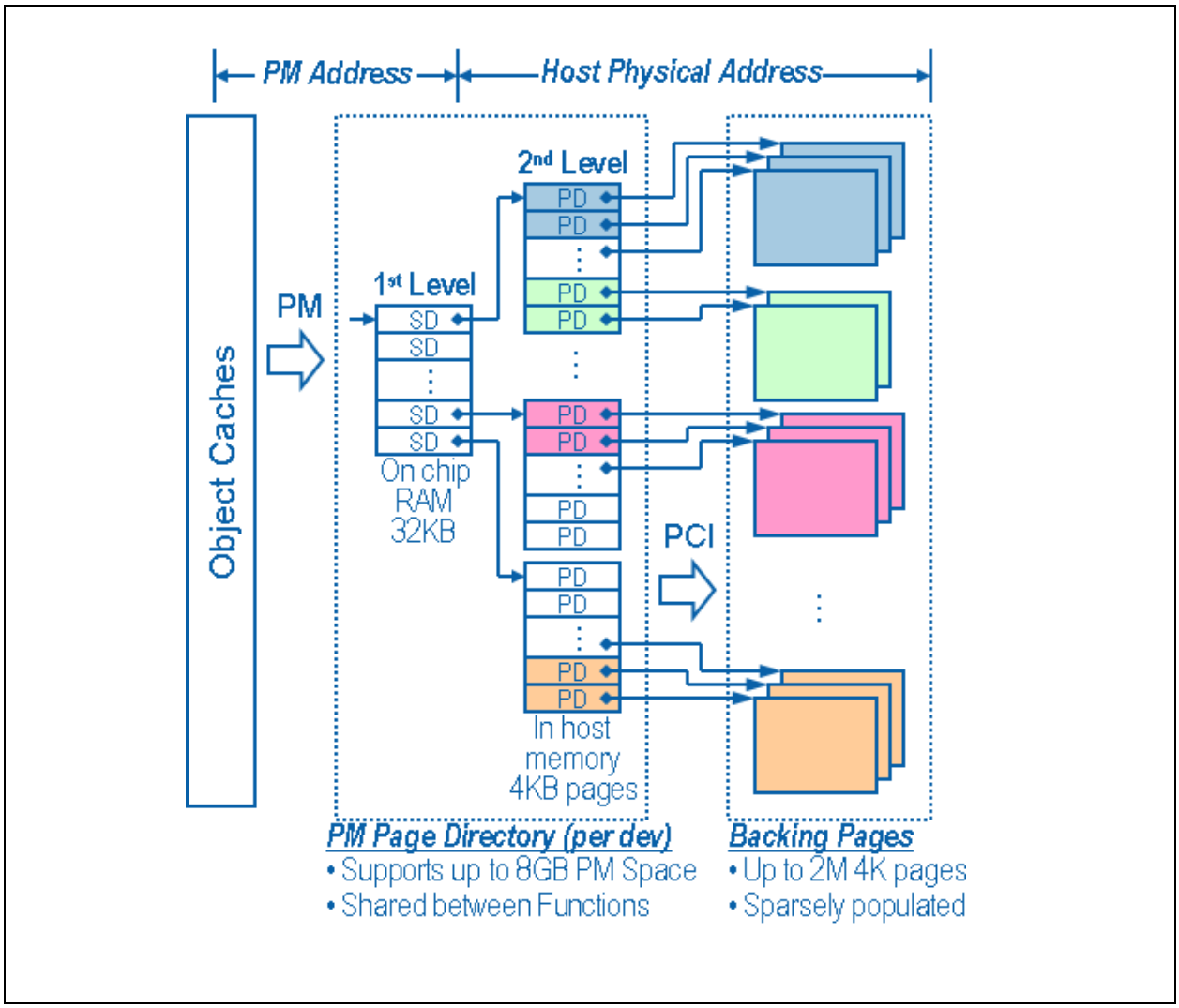


Figure 7-91. Host Memory Cache Private Memory Address Space

The left portion of Figure 7-91 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 32KB 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 XL710. SDs are programmed using the PFHMC_SDCMD (Section 11.2.2.8.52), PFHMC_SDDATALOW (Section 11.2.2.8.53), and PFHMC_SDDATAHIGH (Section 11.2.2.8.54) registers. Everything to the right of the Segment Descriptor Table in Figure 7-91 resides in host memory. Each PCI function has a set of registers (GLHMC_SDPART[n]) that define the base and number of SDs that belong to the PCI function. The GLHMC_SDPART[n] registers are programmed from NVM. The XL710 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 XL710 manages the SD base and number registers internally based on the resource profile that is loaded from NVM.

The second level of private memory address translation provided by the XL710 are 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 8GB 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 8GB 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 7-186.

Table 7-186. HMC Page Descriptor Format

Byte Offset	[Bit Range]Field Name	
0	[63:12]	HMC Backing Page Physical Address
	[11:1]	Reserved
	[0]	PD Valid

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 4KB 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 7-91. The physical address of these PD pages are used to populate SD entries by using the PFHMC_SDCMD, PFHMC_SDDATALOW, and PFHMC_SDDATAHIGH registers shown in Section 11.2.2.8.52, Section 11.2.2.8.53, and Section 11.2.2.8.54.

The Private Memory is further divided into separate PCI Function Private Memory (FPM) Addresses. A PCI function can be either physical function or virtual function. The first 16 FPM address spaces are reserved for PFs. NIC and FCoE VF HMC objects are located in the PF FPM.

Figure 7-92 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 2MB (1 SD). The maximum that could be allocated to a function would be the entire segment table in which case no other function may 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 2MB of HMC PM address space, the FPM also identifies the range of SDs that belong to a PCI function.

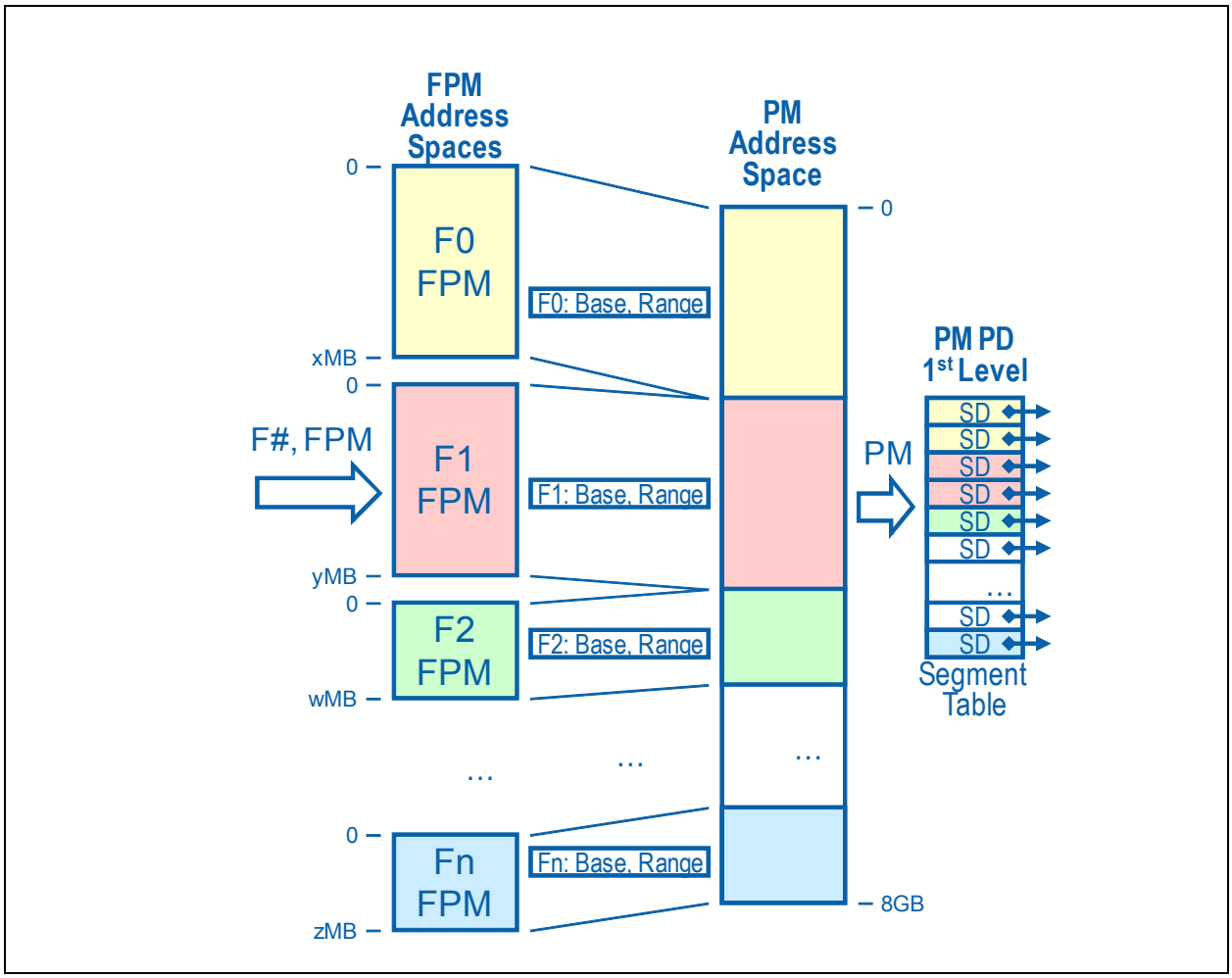


Figure 7-92. Host Memory Cache Function Private Memory Space

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 7-93 depicts some of the current objects that reside in the private memory space (See Table 7-185 for a complete list of the objects). The FPM address is calculated based off of the object type (which identifies the object base register) and object index (the FPM base has already been calculated). In the case of FCoE HMC objects, the VF index is also used to calculate the object index. FCoE objects are organized into blocks for the PF and also blocks for each VF assigned to the PF. Ultimately, the FPM base address, object base address, object size, and object index are all used to determine the private memory address.

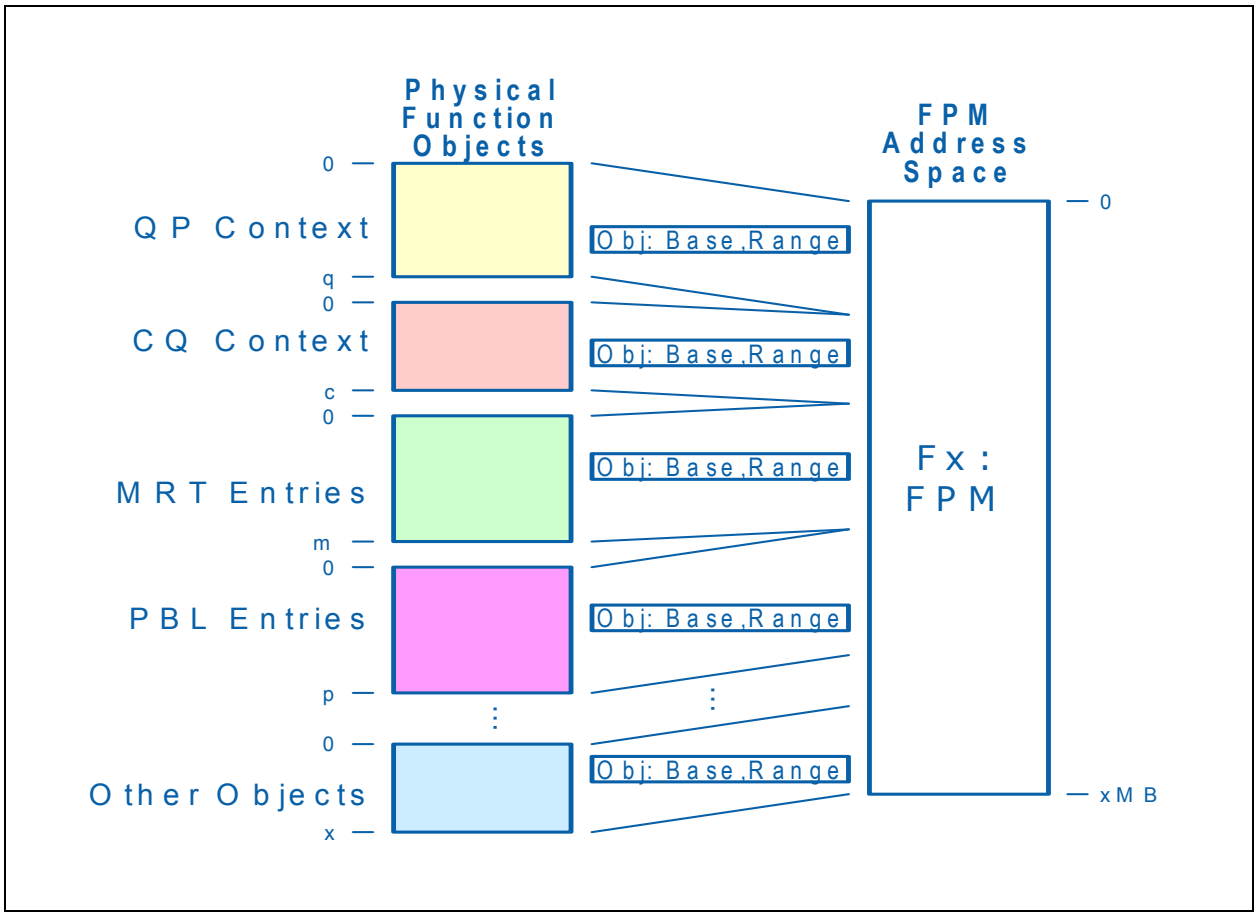


Figure 7-93. Host Memory Cache FPM Object Access

Figure 7-94 describes the decoding of the Private Memory Address into Host Address. Additionally, Figure 7-94 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 2MB pages.

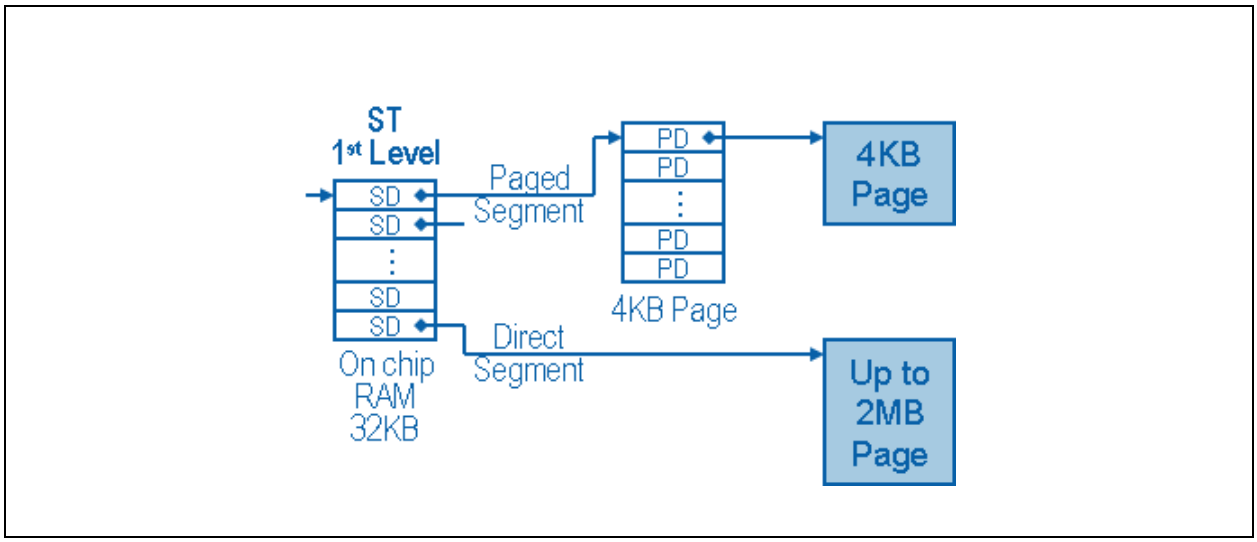


Figure 7-94. Host Memory Cache Direct Segment

See Section 7.9.3 for more details on the specific formats of the formats of the SD entries for Paged and Direct addressing modes.

7.9.2 Function Private Memory Space Configuration

Once NVM has set the default profile or the Set HMC Resource Profile admin queue command has been used to set an appropriate HMC resource profile, 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. In order 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. For LAN queue objects read PFLAN_QALLOC register associated with the PF to find the base queue index and number of queues associated with the PF.
3. For FCoE objects, the number of objects supported per PF and VF are indicated in the GLHMC_FCOEMAX and GLHMC_FCOEFMAX registers. The total number of objects allocated to a PMF for the PF and all associated VFs are defined in the GLHMC_FCOEDDPCNT and GLHMC_FCOEFCNT registers. The first set of objects in the PMF are allocated to the PF and the number of objects for the PF is defined in the PFQF_CTL_0.PFFCDSIZE and PFQF_CTL_0.PFFCHSIZE register fields. For example, if PFQF_CTL_0.PFFCDSIZE was programmed for 4K DDP objects and PFQF_CTL_0.PFFCHSIZE was programmed for 16K Hash buckets, the first 4K DDP objects and the first 20K filter objects would be used for the PF. The subsequent objects are used for the VFs associated with the PF. The number of objects for each VF are defined in the PFQF_CTL_0.VFFCDSIZE and PFQF_CTL_0.VFFCHSIZE register fields. Each VF consumes the same number of objects in the PMF. For example, if PFQF_CTL_0.VFFCDSIZE was set for 2K DDP objects and PFQF_CTL_0.VFFCHSIZE was programmed for 8K hash buckets, each VF assigned to the PF would use 2K DDP objects and 10K filter objects. The starting index of the DDP objects for the first VF would be 4K (leaving room for the PF objects) and the starting index of the filter objects for the first VF would be 20K (also leaving room for the PF objects). Subsequent VFs would start 2K and

10K increments past the first VF starting index. The actual number of FCoE Filter HMC object that must have populated backing pages for the VFs maybe decreased by setting FCHSIZE and FCDSIZE in the VPQF_CTL registers.

4. Each FPM object size register is written with the minimum of the values determined in steps 2-4 or the actual driver needs.
5. Write the base and count registers for each LAN object based on the maximum object index that would be used for the PCI function (HMC PM LAN objects are indexed with the absolute queue number).
6. Write the base registers for each FCoE Object based on the number of objects specified in step 5.
   a. Program LAN and FCoE GLHMC_{object}BASE registers.

Table 7-185 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 7-187. FPM Object Registers

HMC Object	Base Register Array	Object Count Register Array	Maximum Object Count Register	Object Element Size Register
LAN Transmit Queue	GLHMC_LANTXBASE	GLHMC_LANTXCNT	PFLAN_QALLOC	GLHMC_LANTXOBJSZ
LAN Receive Queue	GLHMC_LANRXBASE	GLHMC_LANRXCNT	PFLAN_QALLOC	GLHMC_LANRXOBJSZ
FCoE DDP	GLHMC_FCOEDDPBASE	GLHMC_FCOEDDPCNT	GLHMC_FCOEMAX[1]	GLHMC_FCOEDDPOBJSZ
FCoE Filters	GLHMC_FCOEFBASE	GLHMC_FCOEFCNT	GLHMC_FCOEFMAX[2]	GLHMC_FCOEFOBJSZ

1. PFQF_CTL_0.PFFCDSIZE defines the total FCoE DDP objects for a PF. Likewise, PFQF_CTL_0.VFFCDSIZE defines the number of FCoE DDP objects for each of the VFs assigned to the PF. If VFs will not be enabled for FCoE, then only PF FCoE filter objects need to be allocated. See the description earlier in this section for more details. The total objects for the PF and each VF assigned to the PF must be less than the total FCoE Object configured via the GLHMC_FCOEDDPCNT register.
2. The combination of PFQF_CTL_0.PFFCHSIZE and PFQF_CTL_0.PFFCDSIZE the define total FCoE filter objects for a PF. Likewise, PFQF_CTL_0.VFFCCHSIZE and PFQF_CTL_0.VFFCDHSIZE define the total FCoE filter objects for each VF assigned to the PF. The total objects for the PF and each VF assigned to the PF must be less than the total FCoE Object configured via the GLHMC_FCOEFCNT register. If VFs will not be enabled for FCoE, then only the PF FCoE filter objects need to be allocated. See the description earlier in this section for more details.

7.9.2.1 Programming the HMC FPM Base Registers

Host software is responsible for setting up the GLHMC_{object}CNT and GLHMC_{object}BASE registers for LAN and FCoE objects. All settings of FPM registers impact only the function associated with the registers. In other words, the driver on a given PCI function must only program the GLHMC_{object}CNT and GLHMC_{object}BASE registers for HMC PMs that are owned by that same PCI function. The FPM base of the first HMC object (GLHMC_LANTXBASE) 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 7-188 shows the FPM object order that must be maintained for proper HMC operation and the alignment requirements for each object.

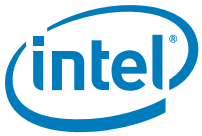


Table 7-188. FPM Object Order and Alignment

HMC Object Order	HMC Object	Alignment Requirement
0	LAN Transmit Queue	512B
1	LAN Receive Queue	512B
2	FCoE DDP	512B
3	FCoE Filters	512B

As a partial example of how the XL710 must program the registers listed in Table 7-188 the following steps would be taken to program the first three HMC objects:

1. Program GLHMC_LANTXBASE = 0
2. Program GLHMC_LANRXBASE = ROUNDUP512((GLHMC_LANTXBASE*512)+(GLHMC_LANTXCNT*$2^{GLHMC\_LANTXOBJSZ}$)) / 512
3. Program GLHMC_FCOETXBASE = ROUNDUP512((GLHMC_LANRXBASE*512)+(GLHMC_LANRXCNT*$2^{GLHMC\_LANRXOBJSZ}$)) / 512

Note: The driver can choose to set the GLHMC_{obj}CNT register to 0 if it does not need to utilize an object. For example, only the GLHMC_LANTXOBJSZ and GLHMC_LANRXOBJSZ register need be non-zero if only LAN resources are needed.

7.9.3 Populating HMC Backing Pages

Once the Function Private memory space has been programmed (Section 7.9.2), 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, PFHMC_SDDATALOW, and PFHMC_SDDATAHIGH registers shown in Section 11.2.2.8.52, Section 11.2.2.8.53, and Section 11.2.2.8.54 are used to populate the SD table for the PCI Function. A driver on a given PCI function must only manipulate SD table entries that are allocated for that PCI function via the SD partitioning process. SDs are addressed on a per PCI function basis starting and 0 and is limited by PMSDMAX. 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 using the PFHMC_SDCMD registers will be ignored (operation will not be performed) by the XL710 and an error will be 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 XL710 before the driver attempts to access the object. The breakdown of the FPM address into components is shown in Table 7-188.

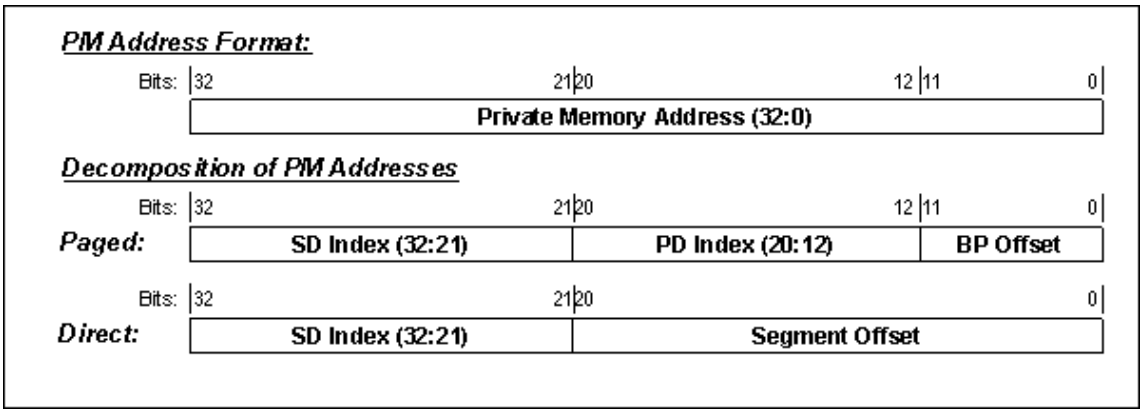


Figure 7-95. FPM Address Decomposition

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)

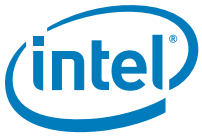
SD_index = INT(FPM_object_address / 2MB)

PD_index = INT(FPM_object_address / 4KB) & 0x1FF

HMC_PM_index = PF index or the HMC VF FPM index

An example of populating the backing pages the HMC is shown assuming that a driver wants to allocate FPM backing pages for 512 LAN Receive Queues starting at index 0:

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_LANTXBASE[HMC_PM_index]*512) + ($2^{GLHMC\_LANTXOBJSZ}$*GLHMC_LANTXCNT[HMC_PM_index])
   b. FPM_object_limit = FPM_object_address + ($2^{GLHMC\_LANTXOBJSZ}$*512)
   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 + 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 7-186 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.



a. Write the most significant 32 bits of the physical address of the PD page to the PFHMC_SDDATAHIGH
b. Write the last significant 32 bits of the physical address of the PD page to the PFHMC_SDDATALOW 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. The PMSDBPCOUNT field is used by the XL710 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 will be populated with the address of a single PD page, and entries 1-129 will be populated with address of the 128 FPM object backing pages that have been allocated.

7.9.4 De-populating HMC Backing Pages

The process of de-populating and freeing HMC object backing pages is the following:

- Ensuring that software and hardware are not going to access objects in the pages
- Calculating the SD range and/or PD range that provide the address mapping to the XL710
- Updating the associated PD entries
- Invalidating the on-die PD cache entries using the PFHMC_PDINV register
- Updating the SDs using the PFHMC_SDCMD, PFHMC_SDDATALOW, and PFHMC_SDDATAHIGH registers to notify the XL710 that the pages are no longer valid for use.

7.9.4.1 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 7-186) 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 (see Section 11.2.2.8.55) 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 the XL710 cache. Once this write is complete, the backing page may 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.

7.9.4.2 Removing a Page Descriptor Page

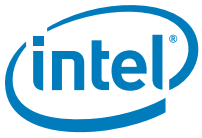
Once software has determined that an entire PD page is no longer needed, the PFHMC_SDDATALOW register must be written with PFHMC_SDDATALOW.PMSDVALID set to 0 and then the PFHMC_SDCMD must be written with PMSDIDX set to the proper SD index value and PMSDWR=1. Once this sequence is complete, software is free to deallocate or re-use the PD page.

7.9.5 HMC Error Reporting

HMC related errors are reported through the PFHMC_ERRORINFO (see Section 11.2.2.8.56) and PFHMC_ERRORDATA (see Section 11.2.2.8.57) registers. The HMC_ERR interrupt status bit in the PFINT_ICR0 register may also deliver an interrupt for HMC errors if the interrupt is enabled in the PFINT_ICR0_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 will be issued until the current error is acknowledged by writing a 0 to the PFHMC_ERRORINFO.ERROR_DETECTED bit. Table 7-189 describes the errors detected for each HMC object and the behavior associated with each error.

Table 7-189. HMC Errors

HMC Object	Error Type(s)	Error Behavior
LAN Transmit Queue	PMF Invalid, Invalid PMF Index	Not Applicable to this object type.
	Invalid LAN Queue Index or FCoE VF index, HMC Object Index Too Large	Index of LAN Queue is reported in the PFHMC_ERRORDATA register. The packets associated with the queue are not transmitted.
	HMC Private Memory Address Too Large, Segment Descriptor Invalid, Segment Descriptor Too Small, Page Descriptor Invalid	The Private Memory Address of the LAN queue object is reported in the PFHMC_ERRORDATA register. Packets associated with the queue are not transmitted.
LAN Receive Queue	PMF Invalid, Invalid PMF Index	Not Applicable to this object type.
	Invalid LAN Queue Index or FCoE VF index, HMC Object Index Too Large	Index of the LAN queue is reported in the PFHMC_ERRORDATA register. Packets associated with the queue are dropped.
	HMC Private Memory Address Too Large, Segment Descriptor Invalid, Segment Descriptor Too Small, Page Descriptor Invalid	The Private Memory Address of the LAN queue object is reported in the PFHMC_ERRORDATA register. Packets associated with the queue are dropped.
FCoE DDP	PMF Invalid, Invalid PMF Index	Not Applicable to this object type.
	Invalid LAN Queue Index or FCoE VF index, HMC Object Index Too Large	Index of the DDP queue or VF Index is reported in the PFHMC_ERRORDATA register. Packets associated with the FCoE DDP context are dropped.
	HMC Private Memory Address Too Large, Segment Descriptor Invalid, Segment Descriptor Too Small, Page Descriptor Invalid	The Private Memory Address of the DDP queue object is reported in the PFHMC_ERRORDATA register. Packets associated with the FCoE DDP context are dropped.
FCoE Filters	PMF Invalid, Invalid PMF Index	Not Applicable to this object type.
	Invalid LAN Queue Index or FCoE VF index, HMC Object Index Too Large	Index of the FCoE filter is reported in the PFHMC_ERRORDATA register. Packets associated with the FCoE filter entry are dropped.
	HMC Private Memory Address Too Large, Segment Descriptor Invalid, Segment Descriptor Too Small, Page Descriptor Invalid	The Private Memory Address of the FCoE filter object is reported in the PFHMC_ERRORDATA register. Packets associated with the FCoE filter entry are dropped.



7.10 Admin Queue

7.10.1 Preface

The admin queue is designed with the following goals:

- Abstract FW interface so that FW can be changed, whether for added functionality or bug fixes, without changing the driver. Further extension of the FW can allow changing parts of the HW 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 admin queue.
- It will be possible to write one driver that would work 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 admin queue for limited transmit and receive.

The XL710 provides the following sets of admin queues:

- One admin queue per PF, exposed in the PF memory BAR
- One admin queue per VF, exposed in the VF memory BAR

Assumptions:

- Currently, FW does not maintain context for any driver operation, except for the state of the queue itself.
- FW deals with one command at a time and does not start working on a new command before finishing it’s current task. This may change in a future version of the FW, so the queue mechanism must allow for it today in order to avoid the need to modify the driver if this happens. FW does however pipeline descriptor and data fetches to optimize execution latency.

7.10.2 Queue Structure

The admin queue is comprised of an admin transmit queue and an admin receive queue. Driver commands are posted on the Admin Transmit Queue (ATQ). FW completes driver commands by writing back onto the command descriptor. Events that are not an immediate result of a command are written to the Admin Receive Queue (ARQ). The driver posts empty buffers to the Admin Receive Queue (ARQ), and the FW fills them with events.

Both queues support direct commands, that fit entirely in the queue descriptor and extended, indirect commands that use 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 FW 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.

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.

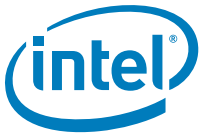
Table 7-190. Admin queue descriptor structure (in LE 32 order)

+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						VFE	ERR	CMP	DD
Return Value / VFID																Data Len															
Cookie High																															
Cookie Low																															
Param0																															
Param1																															
Data address high																															
Data address Low																															

Table 7-191. Admin descriptor Field descriptions

Name	Bytes.Bits	description
Flags.DD	0.0	Set by FW to mark entry done
Flags.CMP	0.1	Set by FW to mark entry as completion
Flags.ERR	0.2	Set by FW to mark entry as an error indication
Flags.VFE	0.3	Set by FW 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 FW needs to read indirect buffer.
Flags.VFC	1.3	Set by driver to indicate command on behalf of a VF.
Flags.BUF	1.4	This command uses additional data
Flags.SI	1.5	Do not interrupt when this command completes
Flags.EI	1.6	Interrupt on error - supersedes Flags.SI in case of an error
Flags.FE	1.7	If previous command completed in error, flush this one
Opcode	2-3	command opcode (see Table 7-202)
Datalen	4-5	indirect data len (can be used for other uses if Flags.RD is unset)
Return value/VFID	6-7	Return value / VF ID for command or event, only sent by FW or PF driver see Section 7.10.8 and Table 7-201
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 uses if Flags.RD is unset)
Data Address low	28-31	indirect data pointer (can be used for other uses if Flags.RD is unset)

See Section 7.10.5 for details on the different command types



7.10.2.1 Admin Queue CSRs

The admin queues have 32 byte descriptors. They are serviced by the following registers (Table 7-192).

{PFX} denotes the register scope prefix:

- for VFs it is "VF_"
- for PFs it is "PF_"

Except for the name prefix the PF, VF registers are exactly the same.

Table 7-192. Admin queue registers

Name	Width	comment
{PFX}ATQBAH	32 bits	High bytes of ATQ base address
{PFX}ATQBAL	32 bits	Low bytes of ATQ base address, address must be 64 byte aligned
{PFX}ATQLEN	10+4 bits	ATQ length in descriptors, MSB is set for queue enable, three error bits (critical error, overflow error and VF error) are set by FW to indicate error conditions (see further Section 7.10.9.1, "Critical Error Indication")
{PFX}ATQH	10 bits	ATQ head pointer (FW updates)
{PFX}ATQT	10 bits	ATQ tail pointer (Driver updates)
{PFX}ARQBAH	32 bits	High bytes of ARQ base address
{PFX}ARQBAL	32 bits	Low bytes of ARQ base address, address must be 64 byte aligned
{PFX}ARQLEN	10+4 bits	ATQ length in descriptors, MSB is set for queue enable, three error bits (critical error, overflow error and VF error) are set by FW to indicate error conditions (see further Section 7.10.9.1, "Critical Error Indication")
{PFX}ARQH	10 bits	ARQ head pointer (FW updates)
{PFX}ARQT	10 bits	ARQ tail pointer (Driver updates)

7.10.2.2 Admin Queue Interrupts

Firmware triggers an interrupt when it completes descriptors on the transmit queue or when it posts events to the receive queue. The interrupt triggered is the "other" interrupt and the ADMINQ cause bit is set to signals that this was the reason. (See Section 7.5.2.4, "Other Interrupt Causes").

7.10.3 Initialization

when initializing the queue the driver must do the following:

- The driver will allocate and setup 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 Chapter 7.10.3.1 for Receive queue element initialization).
- IThe driver clears the head and tail registers for each queue ({PFX}ATQH and {PFX}ARQH. Then, the driver program the base, and length registers for each queue ({PFX}ATQBAL, {PFX}ATQBAH, {PFX}ATQLEN, {PFX}ARQBAL, {PFX}ARQBAH and {PFX}ARQLEN) and sets legnth.enable to 1 to inform FW that the queue is now enabled. 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 will report the

mismatch and fail to load. For details and current queue version number see Section 7.10.12.1 - get version admin command.

- After the driver has verified the queue version, it sends a "Driver Version" command to the device. Device then sends an indication to the BMC that the PF driver is present
- 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.

7.10.3.1 Receive Queue Element Initialization by Driver

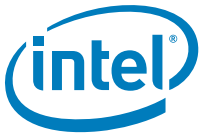
The driver must clear any unused fields (including unused Flags) and set data pointers and data length to a mapped DMA pointer.

The driver may 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 7-193. Receive queue element - initial values

Name	Bytes.Bits	Value	Remarks
Flags.DD	0.0	0	Driver Must Clear
Flags.CMP	0.1	0	Driver Must Clear
Flags.ERR	0.2	0	Driver Must Clear
Flags.VFE	0.3	0	Driver Must Clear
Flags.Reserved	0.4 1.0	0	Reserved, must be zeroed by sender and ignored by receiver.
Flags.LB	1.1	0 or 1	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 May set	Do not interrupt when this command completes
Flags.EI	1.6	Driver May set	Interrupt on error - supersedes Flags.SI in case of error
Flags.FE	1.7	0	Driver Must Clear
Opcode	2-3		Driver Must Clear
Datalen	4-5	Buffer Len	additional data len
Return value/VFID	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 ADDR	indirect data pointer (can be used for other uses if Flags.RD is unset)
Data Address low	28-31	Buffer ADDR	indirect data pointer (can be used for other uses if Flags.RD is unset)

The values written by the FW when it uses the EQ element are discussed in the following paragraphs.



7.10.4 Driver Unload and Queue Shutdown

When shutting down the admin queue the driver (both for PF and VF) will

- Post a "Queue Shutdown" admin command (0x0003) (Section 7.10.12.3). In this command, the driver will set the "Driver Unloading" flag if it intends to unload.
- SW must not send any additional commands on the queue till the flow is completed

FW will

- Wait for any pending DMA transactions from FW to be acknowledged by HW
- Close the RX queue by clearing its "enable" bit.
- Send a completion to the driver as usual, honoring all the interrupt control bits in the descriptor.

SW then closes the TX queue by clearing its "enable" bit.

If the driver is unloading, it issues a function reset (PFR or VFR) to the device

If the driver is unloading, FW will inform the BMC

- Only for a PF driver

Note that before the admin queues are re-enabled, software should clear the head and tail registers of the transmit and receive admin queue.

7.10.5 Commands Description

7.10.5.1 Direct Command

7.10.5.1.1 Direct Admin Command

The template for a command that is fully contained in the descriptor and does not need an additional data buffer.

Table 7-194. Direct Admin command structure

Name	Bytes.Bits	Value	Remarks
Flags.DD	0.0	0	Driver Must Clear
Flags.CMP	0.1	0	Driver Must Clear
Flags.ERR	0.2	0	Driver Must Clear
Flags.VFE	0.3	0	Driver Must Clear
Flags.Reserved	0.4-1.0	0	Reserved, must be zeroed by sender and ignored by receiver.
Flags.LB	1.1	0	A direct command has no additional buffer
Flags.RD	1.2	0	A direct command has no additional buffer
Flags.VFC	1.3	0	Driver Must Clear
Flags.BUF	1.4	0	A direct command does not have an additional write buffer
Flags.SI	1.5	Driver May set	Do not interrupt when this command completes
Flags.EI	1.6	Driver May set	Interrupt on error - supersedes Flags.SI in case of error

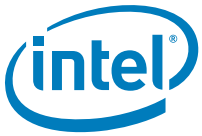
Table 7-194. Direct Admin command structure (Continued)

Name	Bytes.Bits	Value	Remarks
Flags.FE	1.7	Driver May set	If set, command will be flushed if the preceding command resulted in an error
Opcode	2-3	Opcode	Command opcode (see Table 7-202)
Datalen	4-5		Length of external buffer
Return value/VFID	6-7		
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW 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

7.10.5.1.2 Direct Command Completion

Table 7-195. Direct command completion event template

Name	Bytes.Bits	Value	Remarks
Flags.DD	0.0	1	FW Must set
Flags.CMP	0.1	1	FW Must set
Flags.ERR	0.2	0 or 1	FW Must set only if it reporting an error
Flags.VFE	0.3	0	FW Must clear
Flags.Reserved	0.4 -1.0	0	Reserved, must be zeroed by sender and ignored by receiver.
Flags.LB	1.1	echo	FW will copy value from command
Flags.RD	1.2	echo	FW will copy value from command
Flags.VFC	1.3	echo	FW will copy value from command
Flags.BUF	1.4	echo	FW will copy value from command
Flags.SI	1.5	echo	FW will copy value from command
Flags.EI	1.6	echo	FW will copy value from command
Flags.FE	1.7	echo	FW will copy value from command
Opcode	2-3	Opcode	Command opcode (see Table 7-202)
Datalen	4-5		Can be used for an additional command parameter
Return value/VFID	6-7		FW return value 0=no error (for error codes see Table 7-201)
Cookie High	8-11	echo	Opaque value, will be copied by the FW from the command
Cookie Low	12-15	echo	Opaque value, will be copied by the FW 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



7.10.5.2 Indirect Command

7.10.5.2.1 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 needs to be set.

Table 7-196. Indirect command template

Name	Bytes.Bits	Value	Remarks
Flags.DD	0.0	0	Driver Must Clear
Flags.CMP	0.1	0	Driver Must Clear
Flags.ERR	0.2	0	Driver Must Clear
Flags.VFE	0.3	0	Driver Must Clear
Flags.Reserved	0.4-1.0	0	Reserved, must be zeroed by sender and ignored by receiver.
Flags.LB	1.1	0 or 1	Set by driver if buffer is longer than AQ_LARGE_BUF (512)
Flags.RD	1.2	0 or 1	Set if additional buffer has command parameters
Flags.VFC	1.3	0	Driver Must Clear
Flags.BUF	1.4	0 or 1	Driver must set this flag on an indirect command.
Flags.SI	1.5	Driver May set	Do not interrupt when this command completes
Flags.EI	1.6	Driver May set	Interrupt on error - supersedes Flags.SI in case of error
Flags.FE	1.7	Driver May set	If set, command will be flushed if the preceding command resulted in an error
Opcode	2-3	Opcode	Command opcode (see Table 7-202)
Datalen	4-5	Buffer len	Usable length of additional buffer
Return value/VFID	6-7		
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Param0	16-19		First command parameter
Param1	20-23		Second command parameter
Data Address high	24-27	Buff Addr	High bits of buffer address
Data Address low	28-31	Buff Addr	Low bits of buffer address

7.10.5.2.2 Indirect Command Completion

When completing an indirect command the FW will overwrite the datalen with the actual length of data returned by the command.

Table 7-197. Indirect command completion template

Name	Bytes.Bits	Value	Remarks
Flags.DD	0.0	1	FW Must set
Flags.CMP	0.1	1	FW Must set
Flags.ERR	0.2	0 or 1	FW Must set only if it reporting an error
Flags.VFE	0.3	0	FW Must clear
Flags.Reserved	0.4 -1.0	0	Reserved, must be zeroed by sender and ignored by receiver.
Flags.LB	1.1	echo	FW will copy value from command
Flags.RD	1.2	echo	FW will copy value from command
Flags.VFC	1.3	echo	FW will copy value from command
Flags.BUF	1.4	echo	FW will copy value from command
Flags.SI	1.5	echo	FW will copy value from command
Flags.EI	1.6	echo	FW will copy value from command
Flags.FE	1.7	echo	FW will copy value from command
Opcode	2-3	Opcode	Command opcode (see Table 7-202)
Datalen	4-5		Can be used for an additional command parameter
Return value/VFID	6-7		FW return value 0=no error (for error codes see Table 7-201)
Cookie High	8-11	echo	Opaque value, will be copied by the FW from the command
Cookie Low	12-15	echo	Opaque value, will be copied by the FW 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

7.10.6 Command Fetch and Verification

When a command is posted FW looks it up in an internal permission table to decide if the request should be honored. Possible actions are:

- Allow - FW will act upon the command.
- Forward - FW will halt the queue and forward the command to the PF driver, a completion for this command will be initiated by the PF driver when it finishes it, only then further processing of commands from this queue is allowed. (the PF driver will need to re-enable the queue after it deals with the command.)
- Error - FW will complete the action by returning the error specified in the table.
- Drop - FW will behave as if the command succeed but do nothing.

A VF driver is only allowed to post the commands listed in the table below. If any other command is posted by a VF, FW will return the EPERM (operation not permitted) error code.

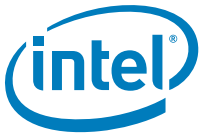


Table 7-198. Commands allowed for VF admin queues

Opcode	Command	Ref
0x0001	Get Version	Section 7.10.12.1
0x0801	Send to PF	Section 7.10.13.1

7.10.7 Non-Completion Events

Events that are not an immediate result of a command completion, are posted by the FW onto the receive queue.

Table 7-199 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 7-199. Non-completion events

Name	Opcode	Ref	Type
VF forwarded command	any[1]	Section	indirect
Send to PF	0x0801	Section 7.10.13.1	Indirect /Direct
Send to VF	0x0802	Section 7.10.13.2	Indirect /Direct
Send to peer	0x0803	Section 7.10.13.3	Indirect /Direct

1. A forwarded command is flagged by the setting of the VFE flag

7.10.8 Error Handling

When FW 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 7-201 the higher byte may be used by FW to report internal state or debug information, the driver must log this information, FW may change this value from release to release. It is not to be reported to the user and no other action is to be taken upon this data. when the return value is 0 (“no error”) FW must not set the high byte. In the event that the queue itself is inaccessible, FW will overwrite the queue base addresses with an error code and clear the enable bit of both queues.

Table 7-200. Admin queue return value fields

Name	bytes	comment
Code	0:7	Return value from Table 7-201
FW internal Code	8:15	FW internal code, must be 0 if Code field is 0

7.10.9 Error Codes

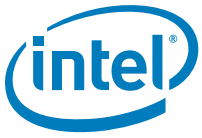
When FW completes a command it shall use the following error codes.

Table 7-201. Admin queue return values and error codes

Error Code	Value	Meaning
(No Error)	0	No Error (success)
EPERM	1	Operation not permitted
ENOENT	2	No such element
ESRCH	3	Bad opcode
EINTR	4	operation interrupted
EIO	5	I/O error
ENXIO	6	No such resource
E2BIG	7	Arg too long
EAGAIN	8	Try again
ENOMEM	9	Out of memory
EACCES	10	Permission denied
EFAULT	11	Bad address
EBUSY	12	Device or resource busy
EEXIST	13	Attempt to crate something that exists
EINVAL	14	Invalid argument
ENOTTY	15	Not a typewriter
ENOSPC	16	No space left or allocation failure
ENOSYS	17	Function not implemented
ERANGE	18	Parameter out of range
EFLUSHED	19	Command flushed because a previous command completed in error
BAD_ADDR	20	Internal error, descriptor contains a bad pointer
EMODE	21	Operation not allowed in current device mode
EFBIG	22	File Too Big

7.10.9.1 Critical Error Indication

- On any error that prevents data placement to a queue ATQLEN.ATQCRIT (or ARQLEN.ATQCRIT) bit will be set by FW and the queue will be stopped (by clearing its enable bit), then an interrupt is sent by FW to the driver.
- SW will read and report the error code and then reset the queue.
- If an overflow occurs and a message to the queue is dropped because the queue is full, FW will set ARQLEN.ATQOVFL and interrupt the driver. (FW will not stop the queue since, depending on the driver mode, this may be a recoverable error.)
- Note: This error can currently only happen on the receive queue, but to simplify the HW design the bit is present on both queues.



- When a VF has an event that causes FW to set an error bit in its queue, FW will set ATQLEN.ATQVFE in the corresponding PFs queue and interrupt it. (FW 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 SW.

7.10.10 List of Admin Commands

7.10.11 Command Opcodes

Opcodes are 16 bits the upper 8 designate the group of the opcode the lower 8 are the command in the group. Each group is described in it's relevant chapter.

Table 7-202. opcode groups

Name	Opcodes	Ref/Remark
Generic	0x00XX	Section 7.10.12 below and Table 7-203
Mac Address	0x0100	Section 4.2.1.5
PXE Mode	0x0110	Section 8.2.2.1
Switch	0x02XX	Chapter 7.4.9.5 and Table 7-68
DCB	0x03XX	Section 7.7.5
Scheduler	0x04XX	Section 7.8.4 and Table 7-155
HMC	0x05XX	Section 7.9.2
Link	0x06XX	Section 3.2.4
NVM	0x07XX	Section 3.3.10
Virtualization	0x08XX	Section 7.10.13
Alternate structure	0x09XX	Section 4.2.2.2.2
LLDP	0x0AXX	Section 7.12.5.2.3

7.10.12 Generic Admin commands

Table 7-203. Generic commands

Name	Opcode	Ref	Type
Get Ver	0x0001	Section 7.10.12.1	direct
Driver Version	0x0002	Section 7.10.12.2	direct
Queue Shutdown	0x0003	Section 7.10.12.3	direct
Set PF context	0x004		direct
Request resource ownership	0x0008	Section 7.10.12.5	direct

Table 7-203. Generic commands

Name	Opcode	Ref	Type
Release resource ownership	0x0009	Section 7.10.12.6	direct
Discover function capabilities	0x000A	Section 7.10.12.7	indirect
Discover device	0x000B	Section 7.10.12.7	indirect

7.10.12.1 Get Version Admin Command

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 FW version is compatible. If FW is still initializing it may delay response until it is done. Both the FW and the API have two unassigned 16 bit values as minor and major version. The driver must not continue loading if the major version mismatch. Minor versions are for tracking changes that do not need driver modifications.

Table 7-204. Get Version command (Opcode: 0x0001)

Name	Bytes.Bits	Value	Remarks
Flags	1-0		See Section 7.10.5.2.1 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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
ROM version	16-19		device ROM version
FW build id	20-23		FW build ID (informational
FW major ver	24-25		Major FW ver (unsigned 16 bit integer)
FW minor ver	26-27		Minor FW ver (unsigned 16 bit integer)
API major ver	28-29	1	Major queue ver (unsigned 16 bit integer) 1 for this version of the queues
API minor ver	31-32	1	Minor queue ver (unsigned 16 bit integer) 1 for this version of the queues

7.10.12.2 Driver Version Indirect Admin Command

Table 7-205. Driver Version command (Opcode: 0x0002)

Name	Bytes.Bits	Value	Remarks
Flags	1-0		See Section 7.10.5.1.1 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, will be copied by the FW into the completion of this command

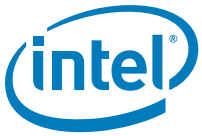


Table 7-205. Driver Version command (Opcode: 0x0002) (Continued)

Name	Bytes.Bits	Value	Remarks
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW 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 ***Note:***
Reserved	20-23	0x0	Reserved
Data Address high	24-27		Address of response buffer
Data Address low	28-31		Address of response buffer

Table 7-206. Driver Version Buffer

Name	Length	Description
Driver version	Datalen	Driver string (not null terminated) as reported by driver:

7.10.12.3 Queue Shutdown command

This is the final command posted to the queue, closing the queue as described in Section 7.10.4. When this command completes the driver is allowed to free any host resources associated with the admin queue.

If the driver is going to unload it must set the "Driver Unloading" flag to inform FW.

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 FW. This means that if an interrupt was not inhibited by setting the "SI" flag it will happen. 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 7-207. Queue Shutdown command (Opcode: 0x0003)

Name	Bytes.Bits	Value	Remarks
Flags	1-0		See Section 7.10.5.1.1 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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Driver unloading	16.0		1 if the driver intends to unload, 0 otherwise
Reserved	17-31	0x0	Reserved

7.10.12.4 Set PF context

This admin command is used to set explicit PF ID number initiated from the Tools admin queue. It is needed for some admin commands that requires control on specific PF context regardless of the PF from which the admin command is initiated.

The default PF context for Tools AQ is PF0. The PF context survive all resets except for EMPR/POR. The Firmware does not provide policy check of AQ commands sent by AQ for Tools on wrong context. The Firmware returns error code according to specific AQ, just like it was sent from the regular admin queue of the PFs.

Table 7-208. Queue shutdown command (opcode: 0x003)

Name	Bytes.Bits	Value	Remarks
Flags	1-0		Section 7.10.5.1.1.
Opcode	2-3	0x0003	Command code.
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 is copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value is copied by firmware into the completion of this command.
PF ID	16		Physical function iD.
Reserved	17-31	0x0	Reserved.

7.10.12.5 Request Resource Ownership Admin Command

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 7-209). On success the command returns a return value of 0. The completion specifies the maximum time in ms that the driver may hold the resource in the Timeout field. The driver must free the resource before that time. The driver must free a resource before asking for it again. A request for a resource that is already held by this driver will fail with the EACCESS error code. A timeout value of zero means no timeout.

If the resource is held by someone else, the command completes with a busy return value and the timeout field indicates the maximum time the current owner of the resource has to free it.

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.

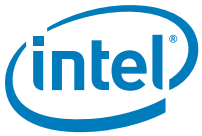


Table 7-209. Shared Resources

Resource	ID	Supported modes	default timeout
NVM	0x0001	1=read, 2=write	3000ms for read, 180000ms for write
SDP	0x0002	1=read, 2=write	0 (no timeout)

Table 7-210. Request Resource Ownership admin command (Opcode: 0x0008)

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.1.1 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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Resource ID	16-17	ID	see Table 7-209 for list of IDs
Access Type	18-19	Type	see Table 7-209 for list
Timeout	20-23	timeout	Timeout in ms (written by FW)
Resource number	24-27	Number	For an SDP, this is the pin ID of the SDP.
Reserved	28-31		

7.10.12.6 Release Shared resource admin command

This command is used to return ownership of a shared resource to the FW. The driver will specify the ID of the resource it is releasing in the ID field.

Table 7-211. Release Resource Ownership admin command (Opcode: 0x0009)

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.1.1 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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Resource ID	16-17	ID	see Table 7-209 for list of IDs

Table 7-211. Release Resource Ownership admin command (Opcode: 0x0009) (Continued)

Name	Bytes.Bits	Value	Remarks
Reserved	18-23	Reserved	Reserved
Resource number	24-27	Number	if a resource number was specified in the request, it needs to be specified here too.
Reserved	28-31	Reserved	Reserved

7.10.12.7 Discover Device/Function Capabilities

This command is used to request the list of capabilities of the device or the function.The FW will fill in the capabilities structure and return the length to the driver. If the buffer size is not big enough for the whole structure the FW will return ENOMEM and set length to the size of the structure (FW will not fill in a partial structure). This can be used to discover the structure size. Capabilities are described using the structure in Table 7-213 the list of resources recognized by this version of the command are in Table 7-214. Additional capabilities can be retrieved using the Get PHY Abilities (0x0600) command and the Get Switch Resource Allocation (0x0202) command (Section 3.2.4.1.4 and Section 7.4.9.5.3.5 respectively).

Table 7-212. Discover Device/Function Capabilities (Opcode: 0x000A, 0x000B)

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.1.1 for details.
Opcode	2-3	0x000A or 0x000B	Command opcode 0x000A for function capabilities 0x000B for device
Length	4-5	buffer length	Length of buffer
Return value/VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-19	Reserved	
cap count	20-23		Number of capability records returned (zeroed by driver written by FW)
Data Address high	24-27	0x0	Address of response buffer
Data Address low	28-31	0x0	Address of response buffer

Table 7-213. Capability Structure

Name	Length	Remarks
Capability	16 bits	See Table 7-214 for list of capabilities
Major Version	8 bits	
Minor Version	8 bits	
Number	32 bits	Number of resources described by this capability
Logical Id	32 bits	Only meaningful for some types of resources

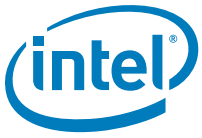


Table 7-213. Capability Structure

Name	Length	Remarks
Physical ID	32 bits	Only meaningful for some types of resources
Reserved1	64 bits	Reserved
Reserved2	64 bits	Reserved

Table 7-214. 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) • 1:Reserved • 2:Cloud • 3:UDP Cloud • Other: Reserved		
Manageability Mode	0x0002	1.0	Returns the value of the "Manageability Mode" field in the NVM		
OS2BMC Capable	0x0004	1.0	Returns the value of the "OS2BMC Capable" field in the NVM		
Functions Valid	0x0005	1.0	Bits 15:0 each corresponds to a PFPCI_STATUS1.FUNC_VALID bit for PFs 15:0		
Alternate RAM Structure	0x0006	1.0	1		
SR-IOV	0x0012	1.0	Set to one if enabled in config space	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		
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 for this product)

Table 7-214. Resources recognized by this version of the command (Continued)

Name	Capability	Ver	Number	Logical ID	Physical ID
FCoE	0x0021	1.0	~~Device = 1:~~ ~~One if enabled in device GLFCOE_ENA.FCOE_ENA~~ ~~Device = 0~~ ~~One if enabled in function.~~ ~~PFGEN_STATE[PF#].PFFCEN~~0x1 if FCoE is enabled. 0x0 if not enabled. For device capability, it is a reflection of the GLFCOE_ENA.FCoE_ENA flag and for a function capability it is a reflection of the PFGEN_STATE.PFFCEN flag.		
RSS	0x0040	1.0	Table size 512 for PFs. 64 for VFs.	Entry width in bits 6 for PFs. 4 for VFs.	
RX Queues	0x0041	1.0	Function: Number of queues allocated to the PF. Device: Total number of queues available to the device		ID of first queue
TX Queues	0x0042	1.0	Function: Number of queues Device: Total number of queues available.		ID of first queue
MSI-X	0x0043	1.0	Number of vectors		
VF-MSIX	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)	
1588	0x0046	1.0	Function: One if enabled on the port on which this function runs Device: One if enabled on any of the ports	Function: N/A Device: A bitmap of the enabled ports	
LED[1]	0x0061	1.0	Always 1	Action	pin number (GPIO Index)

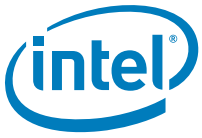


Table 7-214. Resources recognized by this version of the command (Continued)

Name	Capability	Ver	Number	Logical ID	Physical ID
SDP[2]	0x0062	1.0	Always 1	Action	pin number (GPIO Index)
MDIO[3]	0x0063	1.0	One if enabled	Mode: 0x0: MDIO Shared 0x1: MDIO dedicated 0x2: i2c	The interface used to control this port.
WR_CSR_PROT	0x0064	1.0	Bytes 0 … 3 of the WR_CSR_PROT parameter. LS byte is byte 0 of the field.	Bytes 4 … 7 of the WR_CSR_PROT parameter. MS byte is byte 7 of the field.	

1. Repeat this entry for each assigned LED. . If Device = 1, returns an entry for each LED in the device.
2. Repeat this entry for each assigned pin. If Device = 1, returns an entry for each SDP in the device.
3. This entry is not relevant for device = 1 and is not returned.

Note: Fields that are not applicable (empty in the table) are set to zero by the FW.

Note: When device capabilities are requested total numbers for the whole device should be returned.

7.10.13 Virtualization Admin Commands

Table 7-215. Virtualization admin commands

Name	Opcode	Ref	Type
Send to PF	0x0801	Section 7.10.13.1	Indirect /Direct
Send to VF	0x0802	Section 7.10.13.2	Indirect /Direct
Send to peer	0x0803	Section 7.10.13.3	Indirect /Direct

7.10.13.1 Send Message to PF Admin Command

This command, together with the next one, implement 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. If the PF queue is disabled or full the command returns EBUSY. The contents of the messages will be defined by SW.

Since the value of the cookie is copied to the event, if 8 bytes are enough for needed message, the driver may 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 7-216. Send to PF command (Opcode: 0x0801)

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.1.1 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 Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-19		
Reserved	20-23		
Data Address high	24-27	0x0	Address of data buffer
Data Address low	28-31	0x0	Address of data buffer

After posting the event to the PF admin receive queue, the FW will complete this command by updating the flags and return value (0 for success).

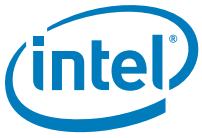
Table 7-217. Message from VF event (Opcode: 0x0801)

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.1.1 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.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-19		
Reserved	20-23		
Data Address high	24-27	0x0	Address of data buffer
Data Address low	28-31	0x0	Address of data buffer

7.10.13.2 Send Message to VF Admin Command

This command, together with the previous one, implements a communication channel between PFs and their FVs. The data in the external buffer is copied into an event on the VF receive queue. The command completes once the data is copied. If the VF queue is disabled or full the command returns -EBUSY. The contents of the messages will be defined by SW. A PF can only send messages to one of it's VFs.

Since the value of the cookie is copied to the event, if 8 bytes are enough for needed message, the driver may 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 7-218. Send to VF command (Opcode: 0x0802)

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.1.1 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 Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
VFID	16-19	VFID	ID of VF
Reserved	20-23		
Data Address high	24-27	0x0	Address of data buffer
Data Address low	28-31	0x0	Address of data buffer

After posting the event to the VF admin receive queue, the FW will complete this command by updating the flags and return value (0 for success).

Table 7-219. Message from PF event (Opcode: 0x0802)

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.1.1 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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-19		
Reserved	20-23		
Data Address high	24-27	0x0	Address of data buffer
Data Address low	28-31	0x0	Address of data buffer

7.10.13.3 Send Message to Peer PF Admin Command

This command may be used for messaging between PFs. Only PFs can use it. It is up to SW to define it's use. If the message can not be delivered -EBUSY will be returned.

Since the value of the cookie is copied to the event, if 8 bytes are enough for needed message, the driver may 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.

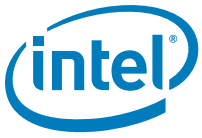
Table 7-220. Send to peer PF command (Opcode: 0x0803)

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.1.1 for details.
Opcode	2-3	0x0803	Command opcode
Length	4-5	buffer length	Length of message
Return value	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
PFID	16-19	PFID	ID of Peer
Reserved	20-23		
Data Address high	24-25	0x0	Address of data buffer
Data Address low	26-31	0x0	Address of data buffer

After posting the event to the PF admin receive queue, the FW will complete this command by updating the flags and return value (0 for success).

Table 7-221. Message from PF event (Opcode: 0x0803)

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.1.1 for details.
Opcode	2-3	0x0803	Event code
Length	4-5	buffer length	Length of message
Return value/VFID	6-7		reserved
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
PFID	16-19	PFID	ID of Peer (sender)
Reserved	20-23		
Data Address high	24-25	0x0	Address of data buffer
Data Address low	26-31	0x0	Address of data buffer



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

7.11 Statistics

The XL710 provides statistics for the following interfaces:

- Physical Ports
- Virtual Switch elements VEBs, PVs, and VSIs
- FCoE statistics are collected per PCI function.
- Flow director statistics.

A minimal set of Ethernet interface group statistics (RFC 2863) is provided by The XL710 at the switch sampling points, a fuller set of statistics is provided for physical ports.

The prefixes for the statistics counter names are listed in Table 7-222

Table 7-222. Counter Name Prefixes

Element	Register Prefix	Instances	Remark
Port	GLPRT	4	
PA or VEB	GLSW	16	
VEB per VLAN	GLVEBVL	128	
VEB per TC	GLVEBTC	16x8	Two dimensional array of 8 TCs for 16 VEBs
VSI	GLV	384	
FCoE Per Function Counters	GL_FCOE	144	Instances 0-127 are for VF 0 to 127, 128-143 - PF 0 to 15

7.11.1 Counter Implementation

Most of the counters in the XL710 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 XL710 does not provide clear-on-read statistics counters, since these would need to be duplicated per client.

As a general rule, the XL710 statistics counters should not be reset by software, because they are shared between more than one owner. Software needs to maintain a delta from the first value seen. If SW need 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, that means that writing any value clears the counter.

The XL710 provides 48-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.

48 bit counters are implemented as a two registers who’s names end with “high” and “low” containing the upper 16 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 HW indication tells the statistics block that this is the case. The statistics block preforms one read of the counter memory to satisfy both requests, thus providing atomicity.)

When 32 bit accesses are used to read the counters they each cause a separate read from the statistics block and therefore care must be taken to properly handle the possibility of one half warping 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.

7.11.2 Statistics Sample Points

Figure 7-96 shows sample points for Tx and Rx statistics. Error counters are in red. The figure shows the processing stages in which counters are updated.

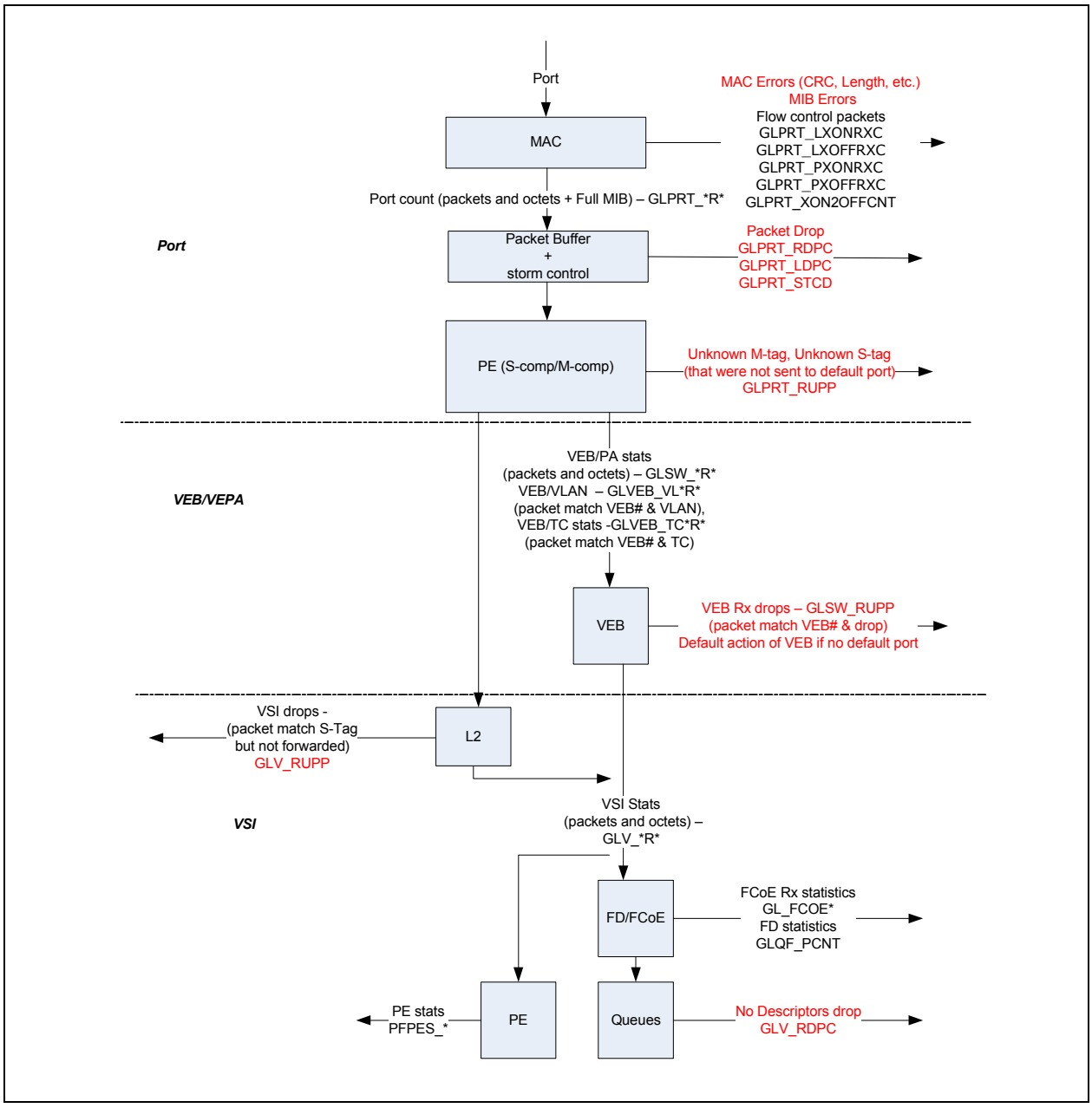
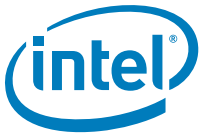


Figure 7-96. RX statistics sampling points.

7.11.3 Statistics Consistency Rules

- At each switch sample point, a packet is either good or error. That means that when a packet is discarded it is not counted in any of the none-error counters. It is also never counted by any of the



next processing stages. For example, an FCoE packet that is dropped at the port because it is too long is never counted in any FCoE counter. On the other hand, an FCoE packet dropped by the FCoE engine will be counted at the port as a good packet.

- All drops are counted. A packet with an error is only counted in one counter. . 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 DDP 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 will also count any packet that would have fit in it without the added S-TAGs and VLANs.

7.11.4 Supported MIBs

The XL710 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 XL710 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 (version 2.6)
- net_device_stats

The following section describes the match between the internal the XL710 statistic counters and the counters requested by the different standards.

7.11.5 Interface Statistics at VSIs and logical Interfaces

The XL710 provides Ethernet interface group statistics per RFC 2863 on each of the Virtual Station Interfaces (VSI), Port aggregators and Virtual bridges. Table 7-223 provides an illustration of interface counters in the XL710 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 element is that the RUPP (unknown protocol) counter is replaced by a missed packet counter RMPC.

Note that the counter names repeat themselves with a different prefix for the element type. See Table 7-222 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, the higher 12 bits rest of the bits are in a register ending with and H (denoted in Table 7-223 as {H,L}).

Table 7-223. Ethernet interface group statistics counters

Register Name	Width	Description
[PFX]GORC{H,L}	48	Incoming packet octets
[PFX]UPRC{H,L}	48	Incoming number of unicast packets
[PFX]MPRC{H,L}	48	Incoming number of multicast packets
[PFX]BPRC{H,L}	48	Incoming number of broadcast packets
[PFX]RDPC	32	Incoming packet discards
[PFX]RUPP	32	Incoming unknown protocol packets
[PFX]GOTC{H,L}	48	Outgoing packet octets
[PFX]UPTC{H,L}	48	Outgoing number of unicast packets
[PFX]MPTC{H,L}	48	Outgoing number of multicast packets
[PFX]BPTC{H,L}	48	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 can not happen at thoes elements in the current switch design. If software is queried about the value of these counters it should return zero (see Table 7-224).

Table 7-224. 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
GLV_TDPC	VSI	VSI transmit packets discarded count
GLSW_RDPC	PA or VEB	Switch receive packets discarded count
GLSW_TEPC	PA or VEB	Switch transmit error packet count

7.11.5.1 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 link interface statistics use the counters defined in Table 7-225. The only exception is the Unknown Protocol Packet counter, which has no meaning in the case of the physical port and therefor is absent.

In addition, the XL710 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 7-225.

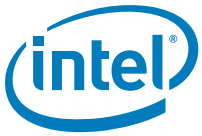


Table 7-225. Additional Per-Port Counters

Register Names	Width	Description
GLPRT_PRC64{H,L}	48	Packets Received [64 Bytes] Count
GLPRT_PRC127{H,L}	48	Packets Received [65–127 Bytes] Count
GLPRT_PRC255{H,L}	48	Packets Received [128–255 Bytes] Count
GLPRT_PRC511{H,L}	48	Packets Received [256–511 Bytes] Count
GLPRT_PRC1023{H,L}	48	Packets Received [512–1023 Bytes] Count
GLPRT_PRC1522{H,L}	48	Packets Received [1024 to 1522] Count
GLPRT_PRC9522{H,L}	48	Packets Received [1523 to Max Bytes] Count
GLPRT_PTC64{H,L}	48	Packets Transmitted (64 Bytes) Count
GLPRT_PTC127{H,L}	48	Packets Transmitted [65–127 Bytes] Count
GLPRT_PTC255{H,L}	48	Packets Transmitted [128–255 Bytes] Count
GLPRT_PTC511{H,L}	48	Packets Transmitted [256–511 Bytes] Count
GLPRT_PTC1023{H,L}	48	Packets Transmitted [512–1023 Bytes] Count
GLPRT_PTC1522{H,L}	48	Packets Transmitted [1024 to 1522] Count
GLPRT_PTC9522{H,L}	48	Packets Transmitted [1523 to Max Bytes] Count
GLPRT_LXONRXCNT	32	Link XON Received Count
GLPRT_LXOFFRXCNT	32	Link XOFF Received Count
GLPRT_LXONTXCNT	32	Link XON Transmitted Count
GLPRT_LXOFFTXCNT	32	Link XOFF Transmitted Count
GLPRT_PXONRXCNT[n]	32	Priority XON Received Count
GLPRT_PXOFFRXCNT[n]	32	Priority XOFF Received Count
GLPRT_PXONTXCNT[n]	32	Priority XON Transmitted Count
GLPRT_PXOFFTXCNT[n]	32	Priority XOFF Transmitted Count
GLPRT_PRRXON2OFFCNT[n]	32	Priority XON to XOFF count
GLPRT_CRCERRS	32	CRC Error Count
GLPRT_ILLERRC	32	Illegal Byte Error Count
GLPRT_MLFC	32	MAC Local Fault Count
GLPRT_MRFC	32	MAC Remote Fault Count
GLPRT_RLEC	32	Receive Length Error Count
GLPRT_RUC	32	Receive Undersize Count
GLPRT_RFC	32	Receive Fragment Count
GLPRT_ROC	32	Receive Oversize Count
GLPRT_RJC	32	Receive Jabber Count
GLPRT_LDPC	32	Loopback drooped packet count
GLPRT_STDC	32	Port Storm Control Discard Count
GLPRT_TDOLD	32	Transmit Packets Dropped On Link Down

7.11.5.2 VEB Statistics

The XL710 supports SMON statistics per RFC 2613 for each instance of the VEB. The XL710 supports the following set of smonVlanStats counters per 802.1Q VLAN and smonPrioStats counters per priority. See Table 7-226 and Table 7-227.

Table 7-226. VEB per VLAN Counters

Register Name	Width	Description
GLVEBVL_GORC{H,L}	48	Incoming packet octets
GLVEBVL_GOTC{H,L}	48	Outgoing packet octets
GLVEBVL_UPC{H,L}	48	number of unicast packets
GLVEBVL_MPC{H,L}	48	number of multicast packets
GLVEBVL_BPC{H,L}	48	number of broadcast packets

Table 7-227. VEB per TC Counters

Register Name	Width	Description
GLVEBTC_RPC{H,L}	48	Total number of packets received per priority
GLVEBTC_RBC{H,L}	48	Total number of octets received per priority
GLVEBTC_TPC{H,L}	48	Total number of packets transmitted per priority
GLVEBTC_TBC{H,L}	48	Total number of octets transmitted per priority

The XL710 provides virtual bridge port or interface statistics counters for VSIs and uplinks associated with a VEB instance. See Section 7.11.5 for additional details on VSI and uplink interface statistics.

7.11.5.3 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 per VLAN statistics. 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 will return 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 stat counters are not available, the allocation will fail. The allocating driver may retry the operation without requesting stats in that case.

7.11.6 FCoE Statistics

7.11.6.1 Per Function FCoE Statistics

FCoE statistics are collected in hardware per PCI function. See Table 7-228 and Figure 7-229 for details.

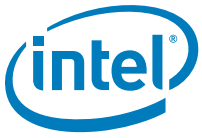


Table 7-228. FCoE Function Statistics

Name	Description	Size	Comment	In 82599
GL_FCOEPRC	FCoE Packets Received Count	32 bits	Number of good FCoE packets posted to the host.	Present
GL_FCOEDWRC {H,L}	FCoE DWORD Received Count	48 bits	Number of Dwords in those packet counted by GL_FCOEPRC. The counter includes all DWords starting after the FCoE encapsulation header up to including the FC CRC.	Present
GL_FCOEPTC	FCoE Packets Transmitted Count	32 bits	Number of FCoE packets transmitted by the host.	Present
GL_FCOEDWTC {H,L}	FCoE DWORD Transmitted Count	48 bits	Number of Dwords transmitted in those packet counted by GL_FCOEPTC. The counter includes all DWords starting after the FCoE exculpations header up to including the FC CRC.	Present
GL_FCOECRC	FCoE Receive CRC Error Count	32 bits	Number of good Ethernet FCoE packets received with bad FC CRC	Present, increased to 32 bits
GL_FCOERPDC	FCoE Receive Packets Dropped Count	32 bits	Number of FCoE RX packets dropped for any reason Note: These are FCoE packets not posted to either DDP or LAN queues Cases: (1) Packet does not have a DDP context, descriptors in LAN queue, and TS is drop (2) Packet went for DDP but DDP descriptors arrive with UR indication	Present
GL_FCOELAST	FCoE Last Error Count	32 bits	Number of FCoE packets received to valid FCoE DDP contexts and valid descriptors in the LAN receive queues while their DDP buffers are exhausted.	Present, increased to 32 bits
GL_FCOEDDPC	FCoE DDP Count	32 bits	Number of FCoE packets received using DDP. Note: packets with DIF errors that were processed by the DDP context are counted as well.	No

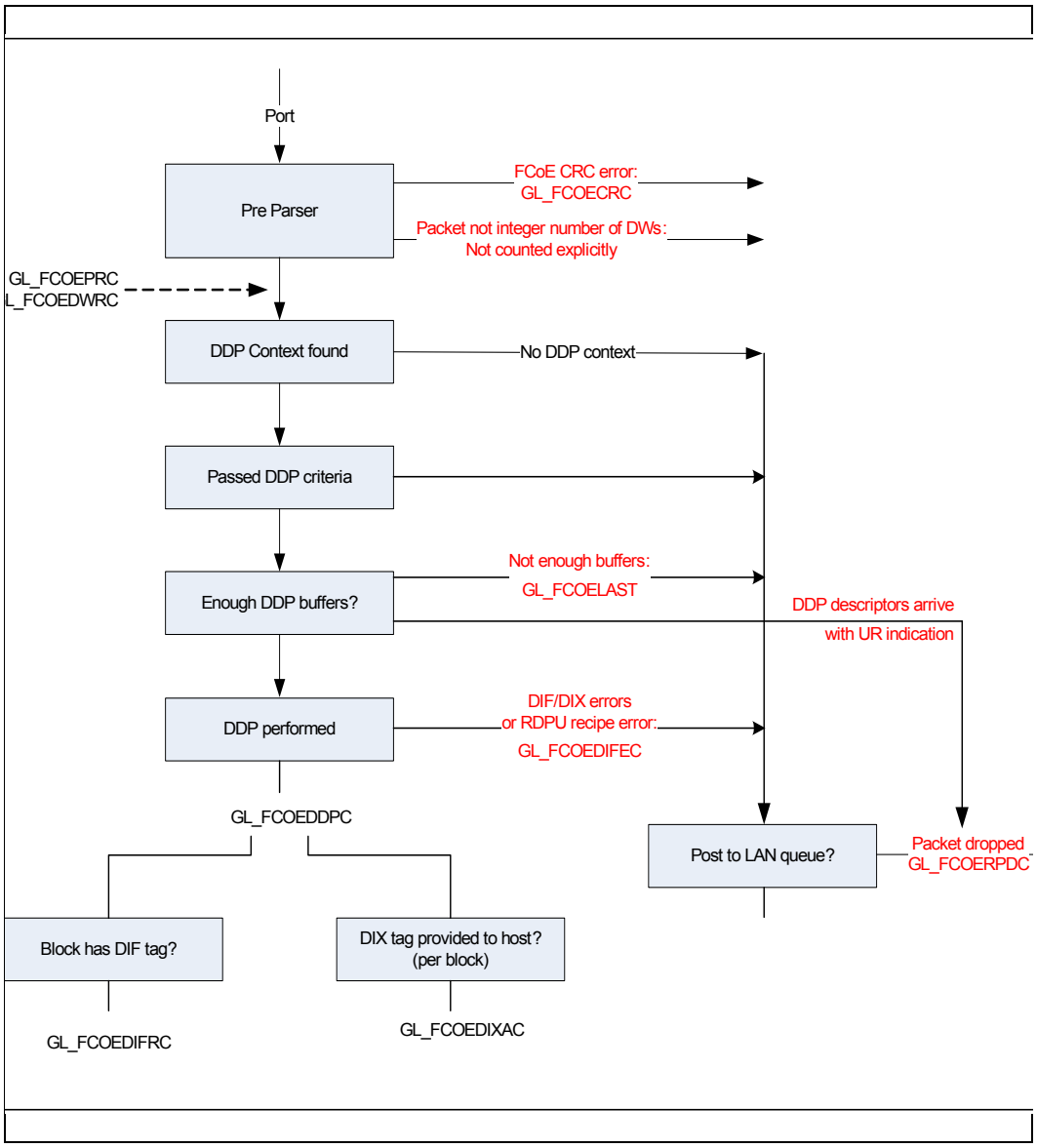


Figure 7-97. FCoE function statistics

7.11.7 Other Statistics

This section includes statistic counters that are not included in the previous sections.

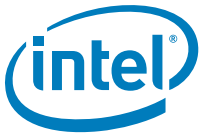


Table 7-229. Other Statistics

Name	Description	Size	Comment
GL_RXERR1 {H,L}	Receive Error Counter 1	64 bits	Count dropped packet due to one of the following exceptions: - Packet size is larger than RXMAX of the queue - FCoE packet received to a queue with inactive FCENA - Internal Receive queue context error (could be a result of a reset in progress) - Receive descriptor Unsupported Request on the PCI or internal Dummy completion (the Dummy completion is a result of a reset in progress)
GL_RXERR2 {H,L}	Receive Error Counter 2	64 bits	Count dropped packet due to one of the following exceptions: - Packets directed to invalid receive queues - Packets directed to disabled receive queues - Packets dropped by the switch filters (these packets are counted also by the switch statistics) - Packets dropped due to MAC errors or FC CRC errors - Packets dropped by the FD filter - Packets dropped due to VM reset, VF reset or PF reset

7.11.8 MFD Statistics

It is important that even in the MFD case, port statistics are reported in some manner to the OS. This is especially so for errors. There are contradicting vendor definitions for how to account port statistics when the OS is not aware of the existence of the internal switch and that several interfaces share a physical link.

One vendor dictates that the Ethernet port stats are accounted to the driver with the lowest PCI function number that is active. Another wants the statistics to be evenly distributed between all active functions.

It is expected that operating systems will become aware of the internal switch, when that happens the port stats can be reported as the switch uplink and the problem is solved.

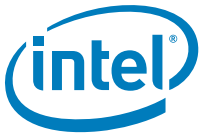
7.12 LLDP Protocol

Revision	Description/Changes
1.0Candidate	• Updated the Admin Queue commands that control LLDP operation
1.0	• Get LLDP MIB response, section for DCB added
1.1	• Per DCR17: — Stop LLDP Agent AQ command is per port — The shutdown LLDP Agent variant of the command is terminated by sending a last LLDP PDU with TTL=0 • Added comment on conditions to enable and disable LLDP in the device • Added subtypes for DCBx TLVs
1.2	• Added DCR 24 - Start LLDP Agent
1.3	• Update of Ingress forwarding rules • Removal of 1 command and addition of 4 commands
1.4	• Added note on End of LLDPDU TLV • Added note on support of a single LLDP neighbor • Added LLDP reset cases (DCR 70) • Various updates in fields of the Admin Queue commands • Removed the Format of the DCB section returned in the Get LLDP MIB response • Removed section on DCBX values in NVM. DCB default values are not loaded from NVM
1.5	• Editorial changes in section “LLDP Agent” • Updated text about CDCP and EVB handling • Added reinitDelay to the list of LLDP parameters • Small updates to AQ commands - in the MIB Type field, in status reporting
1.6	• Added DCR 86 • Clarified the RX forwarding rules • Added a clarification in the “Stop LLDP Agent” command
1.8	• None
1.9	• 2 documentation bug fixes. one in TTL TLV formula and the other in LLDP MIB event parameters.

7.12.1 Introduction

The XL710 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) and IEEE 802.3az - Energy Efficient Ethernet (EEE). 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 LLDPDU (LLDP Data Unit) is stored in two MIBs (Management Information Base) per physical LAN port, one for Nearest bridge and the other for non-TPMR. This information is used to configure the XL710 DCB and EEE features.



7.12.2 Scope

The XL710 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.

7.12.3 LLDP Agent

The following IEEE standards use LLDP to exchange configuration parameters with the link partner.

- Edge Virtual Bridging (EVB)
- Data Center Bridging (DCB)
- Energy Efficient Ethernet (EEE)

The IEEE 802.Qbg specification allows a service provider to support multipoint LAN services to customers over a single physical link by using multiple virtual channels, known as S-Channels.

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 XL710 supports a single instance of DCB and EEE logic per physical LAN port. Therefore, the XL710 DCB logic is associated with the component connected to the physical LAN port, either C or S depending on the use case/configuration.

The XL710 supports an LLDP agent that exchanges LLDP MIB with its peer. In MFP mode, the LLDP agent exchanges LLDP MIB with an S-VLAN Bridge and in single function mode, 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

1. During pre-boot operations including S5 (D0u and D0a)

2. During OS present mode unless SW explicitly turns off the agent using the “Stop LLDP Agent” AQ command

During DCBX exchange, the LLDP agent sets the “willingness” bit and accepts recommended setting from its link partner.

The LLDP agent is enabled during power-on by setting the NVM “LLDP Admin Status” field, (enabled separately per Ethernet port). It is disabled when the “Stop LLDP Agent” is executed (per Ethernet port). SW may transfer ownership of LLDP processing back to the device by issuing a “Start LLDP Agent” AQ command.

7.12.4 LLDP Processing

7.12.4.1 LLDPDU Addressing and Forwarding

7.12.4.1.1 Egress Rules

On the egress side, the LLDP agent uses the appropriate group MAC address as destination MAC and with Ether type 0x88cc:

- DCB and EEE - Nearest bridge address
- CDCP - Non-TPMR nearest bridge address

Source MAC address is the factory assigned SAN MAC address of the lowest PF in the port. All LLDPDU generated by the LLDP agent use this 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 (Section 8.4.2.2.1).Tagged (802.1Qbg or 802.1BR) LLDP packets are forwarded like regular packets. The driver may define different rules (forward to control VSI or drop) using the *Add Control Packet Filter* admin command (Section 7.4.9.5.8.9).

7.12.4.1.2 Ingress Rules

When an LLDP agent is enabled in the device, the following forwarding rules apply for RX LLDP packets:

Table 7-230. Forwarding rules for RX LLDP packets

Destination MAC address	S-tagged LLDP packets	Untagged LLDP packets
Nearest Bridge	Forward to PF if programmed so by the PF. Drop otherwise	Forward to EMP
Non-TPMR	Forward to PF if programmed so by the PF. Drop otherwise	Forward to EMP
Nearest Customer Bridge	Forward to PF if programmed so by the PF. Drop otherwise	SFP mode - Forward to PF if programmed so by the PF. Drop otherwise MFP - drop

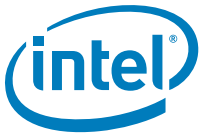
7.12.4.2 Supported TLV

The LLDP agent should include following TLV as part of the LLDPDU.

Chassis ID TLV: Is one of 4 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: Is one of 4 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, then this TLV contains the permanent/factory assigned MAC address of the LAN port.

TTL TLV: Is one of 4 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. Please refer to LLDP Configuration section for default values.



End of LLDPDU TLV: Is one of 4 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 which do not send it. The device therefore does not require that an LLDPDU ends with this TLV.

OEM Device type TLV: Is one of the 3 OEM required TLVs. This TLV uses TLV type value of 127 with subtype of 1. Please refer to OEM specific LLDP specifications for OUI and contents of the TLV.

OEM Firmware Version TLV: Is one of the 3 OEM required TLVs. This TLV uses TLV type value of 127 with subtype of 3. Please refer to OEM specific LLDP specifications for OUI and contents of the TLV.

OEM Port Capabilities TLV: Is one of the 3 OEM required TLVs. This TLV uses TLV type value of 127 with subtype of 4. Please refer to OEM specific LLDP specifications for OUI and contents of the TLV.

DCBX ETS Configuration TLV: This TLV uses 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 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 4 octet string with 4 bits per entry.
- Bandwidth table is a string of 8 octet string 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 8 octets with 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 indicating that this station is willing to accept configuration from remote station.
- MBC is set to 0 since MACsec is not processed by XL710.
- 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.

EEE TLV: EEE TLV is included in the LLDPDU only if both link partners advertise EEE capability for resolved PHY type during auto negotiation.

This TLV uses type value of 127 with OUI of IEEE 802.3 which is 00-12-0F. Subtype is 5. EEE TLV information string is 10 octets long. TLV information string carries timing information following Low Power Idle (LPI) exit.

First six octets contain TransmitTw, ReceiveTw and FallbackTw. Default values for TransmitTw and Receive Tw values are loaded from NVM. Value of FallbackTw is same as ReceiveTw (FallbackTw = ReceiveTw). TransmitTw and ReceiveTw can be changed at runtime by using admin queue commands

from a physical function. (Note: There is a single LLDP agent for all LAN ports and each LAN port has multiple Physical functions. Therefore, it is likely that these variables can overwritten by multiple admin queue commands. Last admin command wins.)

Echo Transmit Tw and Echo Receive Tw values are of the remote link partner. If the link partner has not transmitted an LLDPDU then HW default value for the resolved PHY type.

CDCP TLV: this TLV is over Ethernet frames with nearest non-TPMR bridge group address.

7.12.4.3 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, 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.

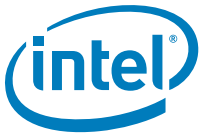
The XL710 supports a single LLDP neighbor (i.e. a single remote MSAP). Therefore, if the received LLDPDU is from a new neighbor, the existing information within the current remote systems MIB is discarded in order to make space for the new information received in the LLDPDU.

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 (e.g. 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

      Exception: See Section 7.7.3.1 for Flow Control configuration
  - — 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

      Exception: See Section 7.7.3.1 for Flow Control configuration
- 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 cause Link Down (i.e. EMP Reset, Global Reset, PCI Resets that cause link down)
    - Device performs an Internal GLOBR that brings the link down momentarily
    - Continue as in Link Down case
  - — LLDP ownership transfer from SW to FW
    - SW issues a Global Reset to the device



    - SW sends a "Start LLDP Agent" AQ command to start LLDP agent in FW
    - Continue as in Link Down case
- Events not visible to other end
  — LLDP ownership transfer from FW to SW
    - LLDP agent terminated in device
    - No automatic change in device configuration
    - EMP waits for SW commands
  — Device Resets that do not cause Link Down (i.e. 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)

7.12.4.4 LLDP Protocol Variables

Unless otherwise specified the LLDP agent operational state variables are set to values recommended by 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. Please refer to Section 7.12.5.2 for LLDP protocol variables that can be configured via NVM.

7.12.4.5 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 - This variable defines the time interval in timer ticks between transmissions during normal transmission periods.

msgTxHold - This variable is used, as a multiplier of msgTxInterval, to determine the value of txTTL that is carried in LLDP frames transmitted by the LLDP agent.

reinitDelay - This parameter 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 - This variable defines the time interval in timer ticks between transmissions during fast transmission periods.

txFastInit - This value determines the number of LLDPDUs that are transmitted during a fast transmission period.

adminStatus - This variable 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 following statistics are maintained by LLDP agent per port:

RemTableLastChangeTime, RemTableInserts, RemTableDeletes, RemTbleDrops, RemTableAgeouts, TxPortFramesTotal, RxPortFramesDiscarded, RxPortFrameErrors, RxPortFramesTotal, RxPortTLVsDiscardedTotal, RxPortTLVsUnrecognizedTotal, RemTooManyNeighbors.

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 XL710 provides LSAP services to the OS provided MSAP services that are used by LLDP agents hosted by system processors

7.12.5 Initialization and configuration

7.12.5.1 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. LLDP agent transitions to 'ready' state and waits for LAN link up before transmitting the first LLDPDU.

DCBX agent operates in slave mode and sets “willingness” bit and accept recommendations from link partner.

7.12.5.2 LLDP Configuration

7.12.5.2.1 LLDP Protocol Variables

The following LLDP timers can be configured via NVM:

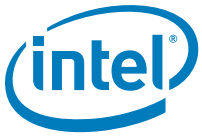
msgFastTx This variable defines the time interval in timer ticks between transmissions during fast transmission periods The default value of msgFastTx is 1. This value can be changed by management to any value in the range 1 through 3600.

msgTxInterval variable 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 variable is 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 value determines maximum number of LLDPDUs that can be sent per second. This value defaults to 5. This value can be changed by management to any value in the range 1 through 10.

txFastInit value 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 This parameter indicates the amount of delay from when adminStatus becomes 'disabled' until re-initialization is attempted. Default value is 2.

7.12.5.2.2 EEE TLV Variables

The following EEE timers can be configured via admin queue commands and defaults are stored and loaded from NVM:

EEE TransmitTw value determines the time (expressed in microseconds) that the transmitting link partner will wait before it starts transmitting data after leaving the Low Power Idle (LPI) mode. This value is platform dependent and depends on the OEM platform. NVM has a default value that is loaded by the LLDP agent. However, this variable can be configured via admin queue interface.

EEE ReceiveTw 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. NVM has a default value that is loaded by the LLDP agent. However, this variable can be configured via admin queue interface.

7.12.5.2.3 LLDP Admin Queue Commands

The following commands are supported by the XL710 to manage the LLDP agent and provide information to the drivers.

Table 7-231. LLDP Admin Queue Commands

Command	OpCode	Description
Get LLDP MIB	0x0A00	Fetch latest LLDP MIB.
Configure LLDP MIB Change Event	0x0A01	Request and deliver a notification that the peer has sent an updated LLDP MIB.
Add LLDP TLV	0x0A02	Add a new TLV to the local LLDP MIB.
Update LLDP TLV	0x0A03	Update an existing TLV in the local LLDP MIB.
Delete LLDP TLV	0x0A04	Delete a TLV from the local LLDP MIB.
Stop LLDP Agent	0x0A05	Used to stop or shutdown LLDP agent.
Start LLDP Agent	0x0A06	Start an LLDP agent running

7.12.5.2.3.1 Get LLDP MIB

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, FW 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, SW may request the local MIB, the remote MIB, or both MIBs.

FW 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, FW always writes the Local MIB first and the Remote MIB immediately following the Local MIB.

Table 7-232. Get LLDP MIB Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Admin Queue section for details.
Opcode	2-3	0x0A00	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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Type	16	MIB Type	Bit 0-1: Direction 0=Local MIB (sent by device) 1=Remote MIB (received by device) 2=Both Local and Remote MIBs 3=Reserved Bits 2-3: Bridge Type 0=Nearest Bridge 1=Non-TPMR Bridge 2=Reserved 3=Reserved Bits 4-7: Reserved
Reserved	17-23	Reserved	Reserved
Data Address high	24-27		Address of response buffer
Data Address low	28-31		Address of response buffer

Table 7-233. Get LLDP MIB Response

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Admin Queue section 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 - FW will return this value if any requested LLDP MIB doesn't exist EPERM - FW will return this value if SW has taken control of LLDP processing EFBIG - FW will return this value when size of LLDPDU is larger than size of the response buffer EINVAL - FW will return this value when SW asks for a bad request (For ex: invalid bridge type or direction)

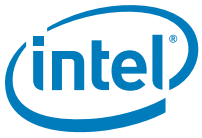


Table 7-233. Get LLDP MIB Response

Name	Bytes.Bits	Value	Remarks
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Type	16	MIB Type	Bit 0-1: Direction 0=Local MIB (sent by device) 1=Remote MIB (received by device) 2=Both Local and Remote MIBs 3=Reserved Bits 2-3: Bridge Type 0=Nearest Bridge 1=Non-TPMR Bridge 2=Reserved 3=Reserved Bits 4-7: Reserved
Reserved	17	Reserved	Reserved
Local MIB Length	18-19	Len	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	Len	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		Address of response buffer

Note: If only one MIB is present, it will be 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 will always be written first (i.e. offset=0), and the remote MIB will be written immediately following the local MIB (i.e. offset=LocalMIBLength).

7.12.5.2.3.2 Configure LLDP MIB Change Event

This command, posted on the ATQ, is a direct AQ command. The driver uses this command to request that FW 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 FW stop posting this event to the ARQ.

FW writes back the status of the request to the "Return Value" field.

Table 7-234. Configure LLDP MIB Change Event Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Admin Queue section 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.

Table 7-234. Configure LLDP MIB Change Event Command

Name	Bytes.Bits	Value	Remarks
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Command	16	Cmd	Bit 0: Command 0=Enable Event 1=Disable Event Bits 1-7: Reserved
Reserved	17-31	Reserved	Reserved

Table 7-235. Configure LLDP MIB Change Event Response

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Admin Queue section 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 - FW will return this value if SW has taken control of LLDP processing
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31	Reserved	Reserved

7.12.5.2.3.3 LLDP MIB Change Event

This event is posted on the ARQ to indicate to SW that any LLDP MIB associated with the physical interface has changed. The LLDP MIB Change Event shall also be posted if a multiple peers condition is detected or if a TLV is aged out.

FW provides the status of the event, the length of the LLDP MIB in bytes, the type of LLDP MIB which has changed, and copies the Cookie value from the associated “Configure LLDP MIB Change Event”.

The response buffer includes the entire MIB which has changed. The formatting is identical to the response for the "Get LLDP MIB" command.

Note that the OpCode for this event is the same as the OpCode for the related command on the ATQ which enabled this event to be sent to SW.

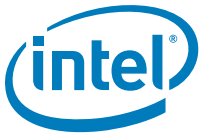


Table 7-236. LLDP MIB Change Event

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Admin Queue section for details.
Opcode	2-3	0x0A01	Event code
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.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Type	16	MIB Type	Bit 0-1: Direction 0=Local MIB (sent by device) 1=Remote MIB (received by device) 2=Reserved 3=Reserved Bits 2-3: Bridge Type 0=Nearest Bridge 1=Non-TPMR Bridge 2=Reserved 3=Reserved Bits 4-5: Miscellaneous 0=Port's TX active 1=Port's TX suspended and drained. 2=Reserved 3= 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		Address of response buffer

7.12.5.2.3.4 Add LLDP TLV

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 which allow multiple instances. SW can only add TLVs to the Local LLDP MIB.

FW will add the new TLV to the Local LLDP MIB just before the "End of LLDPDU TLV". FW 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 7-237. Add LLDP TLV Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section Section 7.10.5.2.1 for details.
Opcode	2-3	0x0A02	Command opcode
Datalen	4-5	Len	Length of indirect buffer.
Return value/ VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Type	16	MIB Type	Bit 0-1: Reserved Bits 2-3: Bridge Type 0=Nearest Bridge 1=Non-TPMR Bridge 2=Reserved 3=Reserved Bits 4-7: Reserved
Reserved	17	Reserved	Reserved
Length	18-19	Len	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		Address of indirect buffer

Table 7-238. Add LLDP TLV Response

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.2.2 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 – FW will return this value if there is not enough space available to add the new TLV. EINVAL – FW will return this value if the MIB Type associated with this command does not exist. EPERM - FW will return this value if SW has taken control of LLDP processing
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command

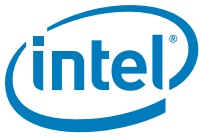


Table 7-238. Add LLDP TLV Response

Name	Bytes.Bits	Value	Remarks
Type	16	MIB Type	Bit 0-1: Reserved Bits 2-3: Bridge Type 0=Nearest Bridge 1=Non-TPMR Bridge 2=Reserved 3=Reserved Bits 4-7: Reserved
Reserved	17-23	Reserved	Reserved
Data Address high	24-27		Address of response buffer
Data Address low	28-31		Address of response buffer

7.12.5.2.3.5 Update LLDP TLV

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 may be updated by the driver.

FW must match the entire original TLV in the MIB before performing an update. There are some TLV types which may appear more than once, and so matching based only on Type/Subtype is not sufficient.

FW 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 7-239. Update LLDP TLV Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0A03	Command opcode
Datalen	4-5	Len	Length of indirect buffer.
Return value/ VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Type	16	MIB Type	Bit 0-1: Reserved Bits 2-3: Bridge Type 0=Nearest Bridge 1=Non-TPMR Bridge 2=Reserved 3=Reserved Bits 4-7: Reserved
Reserved	17	Reserved	Reserved

Table 7-239. Update LLDP TLV Command

Name	Bytes.Bits	Value	Remarks
Length1	18-19	Len	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	Len	Length of updated TLV in the indirect buffer.
Data Address high	24-27		Address of indirect buffer
Data Address low	28-31		Address of indirect buffer

Table 7-240. Update LLDP TLV Response

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.2.2 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 – FW will return this value if the requested MIB does not exist.ENOXIO – FW will return this value if the "original" TLV does not exist in the requested MIB. ENOMEM – FW will return 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 - FW will return this value if SW has taken control of LLDP processing
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Type	16	MIB Type	Bit 0-1: Reserved Bits 2-3: Bridge Type 0=Nearest Bridge 1=Non-TPMR Bridge 2=Reserved 3=Reserved Bits 4-7: Reserved
Reserved	17-23	Reserved	Reserved
Data Address high	24-27		Address of response buffer
Data Address low	28-31		Address of response buffer

7.12.5.2.3.6 Delete LLDP TLV

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.

FW 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.

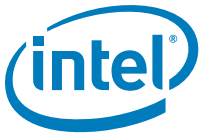


Table 7-241. Delete LLDP TLV Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0A04	Command opcode
Datalen	4-5	Len	Length of indirect buffer.
Return value/ VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Type	16	MIB Type	Bit 0-1: Reserved Bits 2-3: Bridge Type 0=Nearest Bridge 1=Non-TPMR Bridge 2=Reserved 3=Reserved Bits 4-7: Reserved
Reserved	17	Reserved	Reserved
Length	18-19	Len	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		Address of indirect buffer

Table 7-242. Delete LLDP TLV Response

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 7.10.5.2.2 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 – FW will return this value if the requested MIB does not exist. ENOXIO – FW will return this value if the TLV to be deleted does not exist in the requested MIB. EPERM - FW will return this value if SW has taken control of LLDP processing
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command

Table 7-242. Delete LLDP TLV Response

Name	Bytes.Bits	Value	Remarks
Type	16	MIB Type	Bit 0-1: Reserved Bits 2-3: Bridge Type 0=Nearest Bridge 1=Non-TPMR Bridge 2=Reserved 3=Reserved Bits 4-7: Reserved
Reserved	17-23	Reserved	Reserved
Data Address high	24-27		Address of response buffer
Data Address low	28-31		Address of response buffer

7.12.5.2.3.7 Stop LLDP Agent

This command, posted on the ATQ, is a direct AQ command. The driver uses this command to request that FW 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. FW then directs all untagged ingress LLDP frames received on the port to the default queue of the S-component Control VSI.

FW 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. SW should register for events again once 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 Control Packet Filter* command (Section 7.4.9.5.8.9).

Table 7-243. Stop LLDP Agent Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Admin Queue section 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, will be copied by the FW into the completion of this command

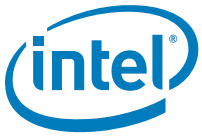


Table 7-243. Stop LLDP Agent Command

Name	Bytes.Bits	Value	Remarks
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Command	16	Cmd	Bit 0: Command 0=Stop LLDP Agent 1=Shutdown LLDP Agent Bits 1-7: Reserved
Reserved	17-31	Reserved	Reserved

Table 7-244. Stop LLDP Agent Response

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Admin Queue section 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 - FW will return this value if SW has taken control of LLDP processing EMODE - FW returns this value in MFP modes
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31	Reserved	Reserved

7.12.5.2.3.8 Start LLDP Agent

It is expected that CORER has occurred before this command is issued. CORER will cause the EMP to reload LLDP forwarding rules from NVM and re-initialize per NVM settings.

This command, posted on the ATQ, is a direct AQ command. In response to this command EMP re-enables LLDP agent over all ports enabled by the NVM "LLDP Admin Status" word. It is expected that only pre-boot SW will issue this command to start the LLDP agent.

If the driver defined LLDP forwarding rules previous to sending this command, these rules should be removed using the Remove Control Packet Filter admin command (Section 7.4.9.5.8.10) before sending this command.

Table 7-245. Start LLDP Agent Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Admin Queue section for details.
Opcode	2-3	0x0A06	Command opcode
Datalen	4-5	0	Direct command; no response buffer.

Table 7-245. Start LLDP Agent Command

Name	Bytes.Bits	Value	Remarks
Return value/ VFID	6-7		Return value. Zeroed by driver. Written by Firmware.
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Command	16	Cmd	Bit 0: Command 0 = Do not start the LLDP agent (**Note: this value should not be used**) 1= Start LLDP agent Bits 1-7: Reserved
Reserved	17-31	Reserved	Reserved

Table 7-246. Start LLDP Agent Response

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Admin Queue section 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 FW
Cookie High	8-11	Cookie	Opaque value, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31	Reserved	Reserved

8.0 LAN Engine

8.1 LAN Cache and Private Host Memory

The LAN transmit and receive queue contexts as well as FCoE contexts are stored in memory pages in the Function Private Memory (FPM) space detailed in Section 7.9. The LAN Queue context, FCoE context as well as the programmed filters pages of both the PF and its VFs reside in private memory spaces of the PF. A conceptual description of these structures (assuming each "object" starts at a new PD) is illustrated in the Figure 8-1.

The FPM is considered a private memory of the hardware; software is not expected to access it other than in LAN queue contexts for initial programming (as described in this chapter). During run time, the hardware fetches the queue context and filters from the FPM to its cache according to internal needs.

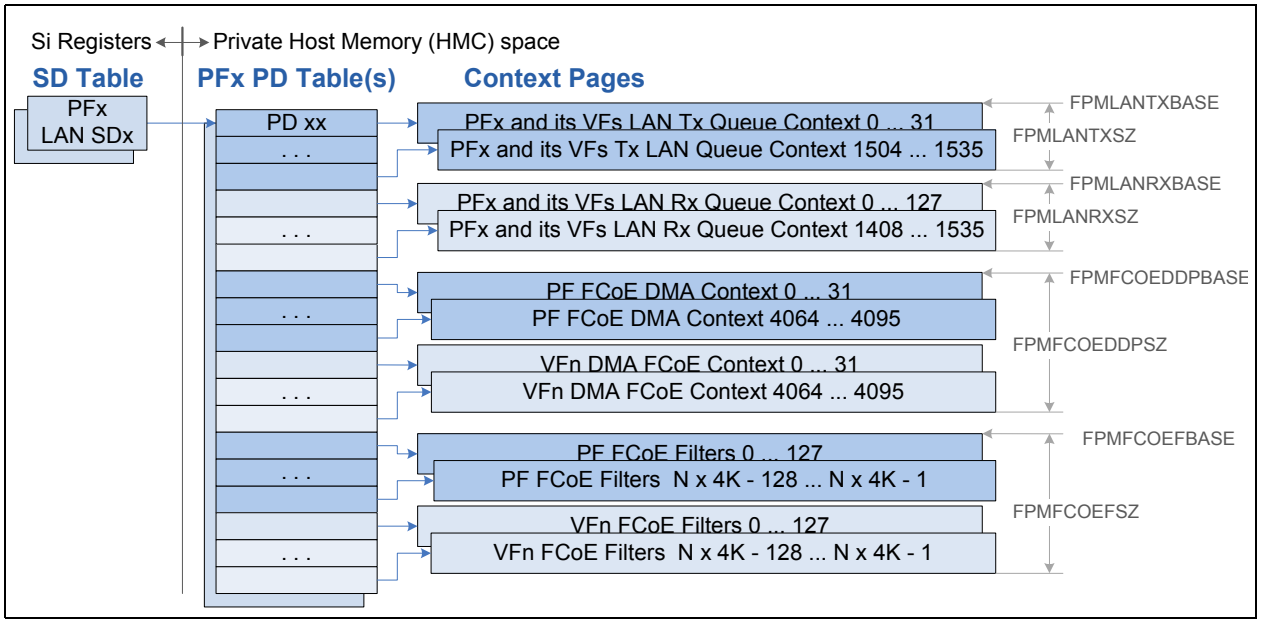


Figure 8-1. PF LAN Private Host Memory (FPM)

8.2 Shared LAN Data Path

8.2.1 LAN Queue Pairs Allocation

8.2.1.1 LAN Queue Pairs Allocation to PFs

Queues are allocated in pairs of transmit and receive queues, called LAN queue pairs (LQP) in this section. LQPs are statically allocated to PFs in a flexible manner according to PFLAN_QALLOC registers (loaded from the NVM):

- Active PFs must have LQPs enabled by the VALID flag.
- LQPs of a PF start at the absolute LQP index defined by the FIRSTQ field and end at the absolute LQP index defined by the LASTQ field.

The allocated LQPs for the PFs must be defined monotonically. This means that LQPs of PF(n) must be allocated after the LQPs of PF(n-1) in the absolute physical device space. See an example of 4 PFs in Figure 8-2 .

Note: Based on the PFLAN_QALLOC values, the PF software allocates the LQPs for its usage and for its VFs as detailed in the following subsections.

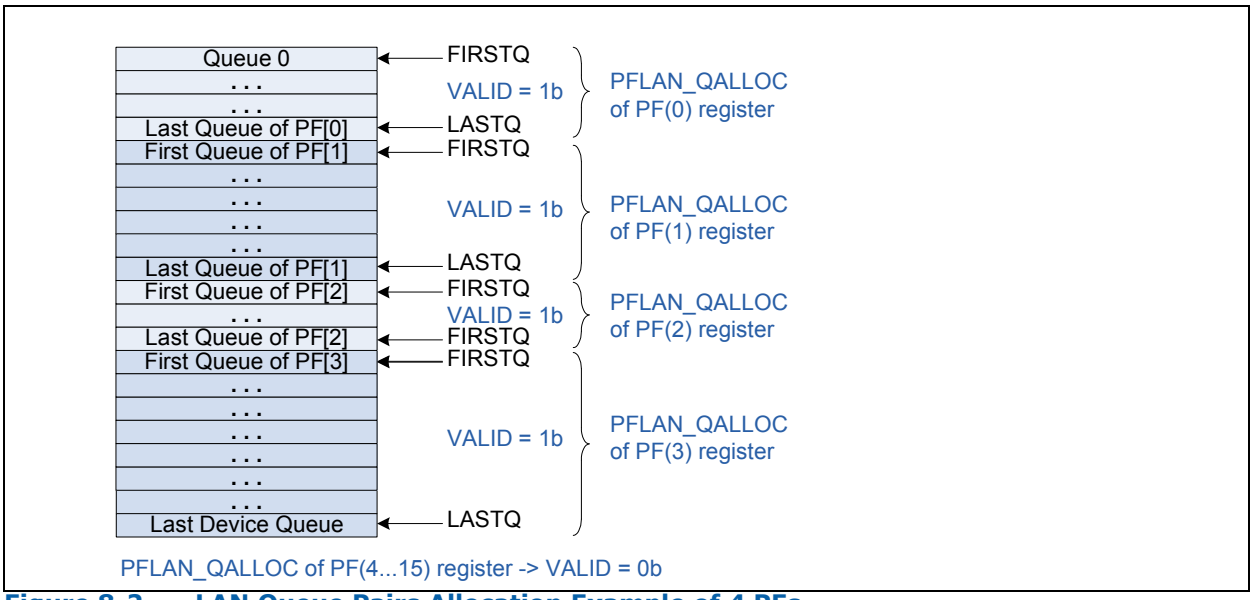


Figure 8-2. LAN Queue Pairs Allocation Example of 4 PFs

8.2.1.2 LAN Queue Pairs Allocation within a PF

PF software is expected to read the values of the PFLAN_QALLOC register to determine the LQPs owned. The PF allocates these LQPs using the add VSI AdminQ command which program the following registers: VPLAN_QTABLE's, VSILAN_QBASE and VSILAN_QTABLE's. An example of LQP allocation

within a PF to its VSIs is illustrated in Figure 8-3. Note the following:

- Receive queue zero of the PF is reserved for Flow Director Filter Programming Status Descriptor (see Section 8.3.2.2.3). For software simplicity it is possible to avoid including LQP zero of the PF to any VSI.
- VFs and VMDq2 VSIs are allocated up to 16 "scattered" LQPs in the PF space by the VPLAN_QTABLE and VSILAN_QTABLE registers. The VSILAN_QTABLE registers enable scattered LQPs 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_QTABLE.
  - These tables are not readable for the VFs. Instead, the VF is notified by the PF about the number of its allocated LQPs (using a software API or the MailBox). The PF on its side, must allocate the LQPs starting at LQP zero and setting contiguous entries in these tables.
  - Mapping of a VSILAN_QTABLE[m] to a VF is the same for all other VSI registers defined by the VSI_VSI2F registers.
- 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 LQPs in the PF space can get any number of LQPs.
  - The size of the contiguous VSI is not enforced by hardware setting; it is a software responsibility 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 PF software responsibility to define the receive classification filters so that the queues do not exceed the VSI space.
- Contiguous vs. "scattered" VSI is controlled by the VSIQTABLE_ENA flag in the VSILAN_QBASE register.
- It is the PF software responsibility to define "free" LQPs that are within the space of the PF.

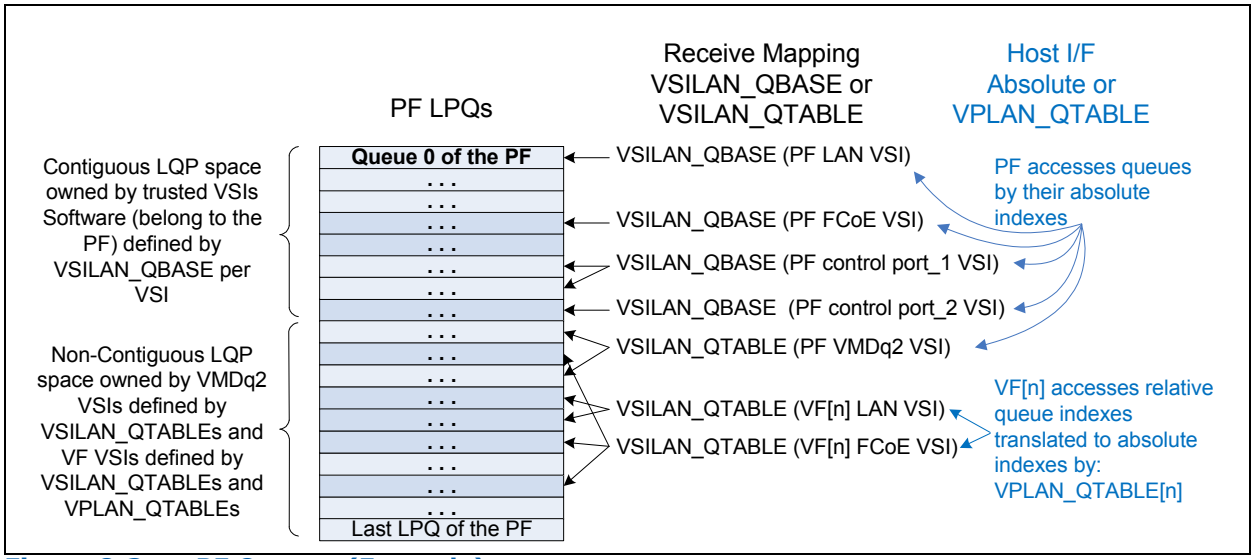


Figure 8-3. PF Queues (Example)

8.2.1.2.1 Software Access to the Queues of the Functions

PF software accesses its LQPs by their index within the PF space. Queue index 'm' of PF 'n' equals to 'm + PFLAN_QALLOC[n].FIRSTQ' in the device space. Accessing LQPs 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 LQPs using an index within the VF space. LQPs indexes are translated to indexes in the PF space by the VPLAN_QTABLE registers. The VPLAN_QTABLE supports up to 16 LQPs while a VF might get less than all 16 LQPs. A VF software attempt to access LQPs above the allocated ones does not impact device functionality (the same as described above for the PF).

Note that a function (PF or VF) might have multiple VSIs. Still, software accesses all LQPs by their index in the function space rather than index in the VSI spaces.

8.2.1.2.2 Associating Received Packets to VSI Queues

XL710 associates received packets to VSIs by the embedded switch as described in Section 7.4. It then associates packets to queues using classification filters.

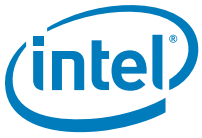
- As opposed to the software interface, the classification filters assign a queue index in the VSI space. The queue index 'n' is mapped to the PF space as follows:
  - Queue index 'n' of a contiguous VSI is mapped to queue index 'n + VSILAN_QBASE.VSIBASE'
  - Queue index 'n' of a "scattered" VSI is mapped 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 to 0x3FF or if 'n' is greater or equal than 16.
- Packets received to invalid queues are dropped and counted by the GLV_REPC counter of the VSI.

8.2.1.2.3 Dynamic Queue Allocation

XL710 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. Queues can be de-allocated 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.

The steps below relate to LQPs of VFs while it is similar to LQPs of VMs:

- PF software communicates to the VF that it needs to give up a specific LQP(s).
- The PF removes the queue(s) from the VPLAN_QTABLE and the VSILAN_QTABLE. As a result, the VF can no longer access the queue(s).
- PF software disables the relevant queues, following "queue disable" flow described in Section 8.3.3.1.2 and Section 8.4.3.1.2. As part of the flow, the PF waits for the hardware indication that the queues are disabled with no further activity to host memory.
- The PF notifies the VF that it can release the host memory structures of the removed LQP(s).
- The PF remaps these LQP(s) to another VF as follows:
  - It programs the new Queue context parameters in the FPM and then enables these queues (see Section 8.3.3.1.1 and Section 8.4.3.1.2 for receive and transmit queue enablement flows).
  - It maps the LQP(s) to the new VF in the VPLAN_QTABLE and VSILAN_QTABLE and informs the VF about this action.



- The queues are ready to be used by the VF.

8.2.1.3 LAN Queue Pair Allocation Example

Allocate the LQPs to the VSIs of the PF by setting the VSILAN_QBASE registers, VSILAN_TABLE and VPLAN_TABLE. The table below shows an example of 5 VSIs while the PF owns 128 LQPs (as configured by PFLAN_QALLOC.FIRSTQ = 128 and PFLAN_QALLOC.LASTQ = 255):

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 FCoE 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, 102 and
			QINDEX_0 and QINDEX_1 in VSILAN_QTABLE[10,1...7] = 0x7FF
			Queues 100,102 in the PF space are absolute queues 228,230

100	VF[20]	4 LQPs	VSILAN_QBASE[100].VSIQTABLE_ENA = 1 // scattered range
			VSILAN_QTABLE[100,0].QINDEX_0 and QINDEX_1 = 90, 98
			VSILAN_QTABLE[100,1].QINDEX_0 and QINDEX_1 = 110, 112
			QINDEX_0 and QINDEX_1 in VSILAN_QTABLE[100,2...7] = 0x7FF
			VPLAN_QTABLE[20; 0,1,2,3,4:15].QINDEX = 90, 98, 110, 112, 0x7FF
			Queues 90,98,110,112 in the PF space are absolute queues 218,226,238,240

8.2.2 LAN 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:

- The LAN queue pairs (LQPs) are allocated to the PF by NVM setting loaded to the PFLAN_QALLOC registers following a core reset (CORER).
- The statistic counters are cleared only at power on reset (POR). As part of the PF driver init, 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.
- 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 8.2.2.1 and Section 8.2.2.2.

- 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:
  - LQPs mapping:
    - Clear the QTX_ENA, QRX_ENA, QTX_TAIL and QRX_TAIL registers of all the PF and its VFs LAN queue pairs.
    - 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. In MFP setup, these registers might be initialized only following the request of the first PF on this port.
- Allocate the LQPs to VFs and VSIs of the PF:
  - For each assigned VSI do the following as part of "create VSI" procedure:
    - Allocate the LQPs to the VSI (see programming example in Section 8.2.1.3).
    - Program the VSILAN_QBASE and VSILAN_QTABLE (use QBASE option for "contiguous" LQPs or the QTABLE option for "scattered" LQPs).
  - Program the VPLAN_QTABLE of each assigned VF.
- Enable individual receive and transmit queues of the PF and its VFs following the flow described in Section 8.3.3.1.1 and Section 8.4.3.1.1 respectively.

8.2.2.1 Clear PXE Mode Admin Command

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. The complete software and device response is followed in section 8.2.2.2.

Table 8-1. Clear PXE Mode Admin Command (Opcode: 0x0110)

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.1 for details.
Opcode	2-3	0x0110	Command opcodes
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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16	0x2	Reserved 0x2
Reserved	17-31		

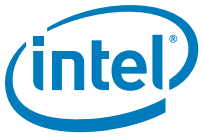


Table 8-2. Clear PXE Mode Admin Command Completion (Opcode: 0x0110)

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 7.10.5.2.2 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 7.10.9 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, will be copied by the FW into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the FW into the completion of this command
Reserved	16-31		Reserved

8.2.2.2 Transitioning Flow to non-PXE mode

8.2.2.2.1 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_ENA[0] and QRX_ENA[1] registers of the PFs.
   b. Wait till all receive queues are disabled by polling the QENA_STAT flag in the QRX_ENA registers of all above Rx queues. It is expected that all Rx queues are disabled within a few usec.
   c. Clear the PXE_MODE flag in the GLLAN_RCTL_0 register
   d. Flow is completed by posting a command completion with "No Error"

8.2.2.2.2 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.

8.2.3 Cloud Computing Support

The Cloud provides software, storage and compute infrastructure services over the Internet. Cloud platforms should provide isolation among tenants by creating private virtual networks over the physical network infrastructure and should be able to provision services on any physical machine within the data center. XL710 provides filtering and pruning functionality that supports these goals.

Cloud providers build virtual network overlays over existing network infrastructure that provide tenant isolation and scaling. Tunneling layers added to the packets carry the virtual networking frames over existing Layer 2 and IP networks. Conceptually, this is similar to creating virtual private networks over the Internet. Processing these tunneling layers by the hardware is a critical element of this solution.

The tunneling packet formats supported by XL710 are illustrated in the Figure 8-4. The fields in the packet used for switching functionality are described in Section 7.4.9.3.1. The switch filters are described in subsections of Section 7.4.4. See also transmit and receive descriptors for requested offloads on transmission and reported status on reception.

The XL710 should be set for tunneling mode or non tunneling mode by NVM image. NVM images are used to set the VSI filters and receive classification filters. There are 3 planned NVM images: (1) non-tunneling; (2) MAC in UDP tunneling; (3) IP in IP, IP in GRE and MAC in GRE tunneling.

An overview of the packet processing is described in Table 8-3.

Table 8-3. Tunneled Packet Processing

Packet Type	Accept / Reject packet	Forward the packet to specific VSI (based on...)	Identify, Insert / Strip L2 Tags	L4/IPv4 XSUM	TSO	Tunneling Header Split (on top of non tunneling options)	RSS (hash) and FD Filters
IP in IP	Based on outer L2 MAC and optional L2 Tags (including VLAN, QinQ or E/S Tag w/ wo VLAN)	Outer IP	L2 Tags in the outer L2 Headers Only single VLAN in the inner L2 header	Both inner and outer IP headers, Inner L4 header	Yes	Outer IP header	Yes (same as non tunneled packets)
IP in GRE		Outer IP w/wo GRE key				GRE Header	
MAC in GRE		Inner L2 MAC/VLAN w/ wo one of the following: GRE key; Outer IP; Outer MAC				GRE Header	
MAC in UDP		Inner L2 MAC/VLAN w/ wo VN Key words				Outer UDP header and its extension header	

The Figure 8-4 below illustrate the supported tunneled packet formats:

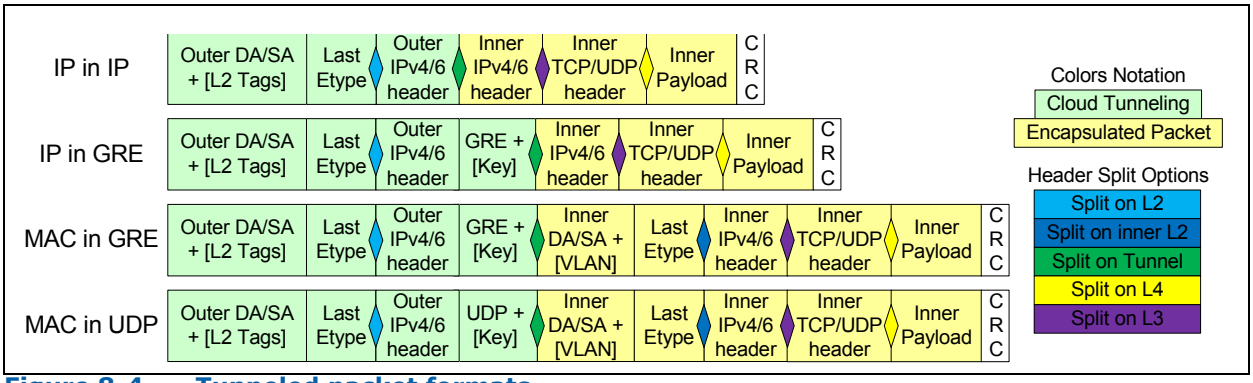
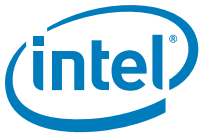


Figure 8-4. Tunneled packet formats



8.2.4 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. The following table 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 8-4. 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	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

8.3 LAN Receive Data-Path

The LAN Receive Data-Path includes the following major topics:

- Receive packets stored in system memory.
- Indicating the free memory structures to the hardware and indicating the completed structure back to the software.
- Receive descriptor queues which are called also “descriptor ring”
- Receive arbitration (covered in Section 7.7.1.1)
- Stateless Receive Offloads

8.3.1 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 8.3.2; these include pointers to the data buffers and status indications of the received packets. The Figure 8-5 shows two examples of receive packets in host memory composed of 2 buffers (indicated by 2 matched descriptors). XL710 fetches the receive descriptors (on demand) to an internal cache. A few rules relating receive packet posting to host memory are:

- Receive packets may span one to 5 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). These packets are counted by the GLPRT_MSPDC counter per LAN Port.

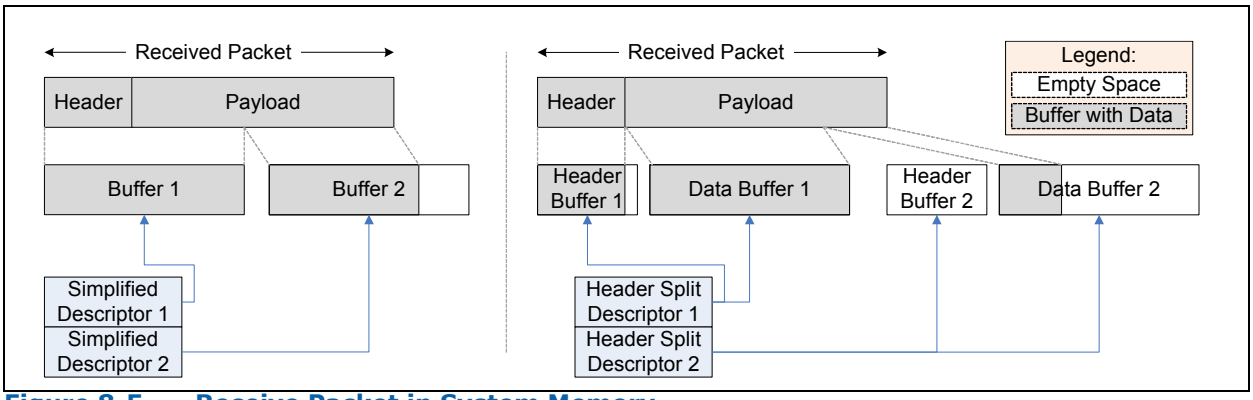


Figure 8-5. Receive Packet in System Memory

8.3.1.1 Receive Descriptor Cache

8.3.1.1.1 Descriptor Fetch Policy

XL710 fetches multiple receive descriptors at a time in order to minimize PCIe and memory bandwidth overhead; 8 or 4 descriptors when using 16 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 is set). PXE mode functionality and limitations are:
  - The receive queue should not be larger than 16 x 16 byte descriptors.
  - Software can bump the tail at descriptor granularity. Hardware fetches and writes back these descriptor at descriptor granularity as well.
  - Each packet may 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. This step is expected to occur during the PF software init procedure. In 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, software should bump the tail at whole 8 x descriptors granularity. In this mode, hardware fetches descriptors in whole cache lines (4 x 32 byte descriptors or 8 x 16 byte descriptors).

8.3.2 LAN Receive Descriptors

8.3.2.1 Receive Descriptor - Read Format

8.3.2.1.1 16 Byte Receive Descriptors Read Format

Described below is the 16 Bytes receive descriptor read format prepared by the software.

Table 8-5. 16 Byte Receive Descriptors Read Format

Quad Word	63 ... 0
0	Packet Buffer Address
1	Header Buffer Address

Packet Buffer Address (64)

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 (64)

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 that 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).

8.3.2.1.2 32 Byte Receive Descriptors Read Format

Described below is the 32 Byte receive descriptor read format prepared by the software.

Table 8-6. 32 Byte Receive Descriptors Read Format

Quad Word	63 ... 0
0	Packet Buffer Address
1	Header Buffer Address
2	Reserved (0x0)
3	Reserved (0x0)

The fields in first 16 Bytes Identical to the 16 Byte descriptors described in Section 8.3.2.1.1.

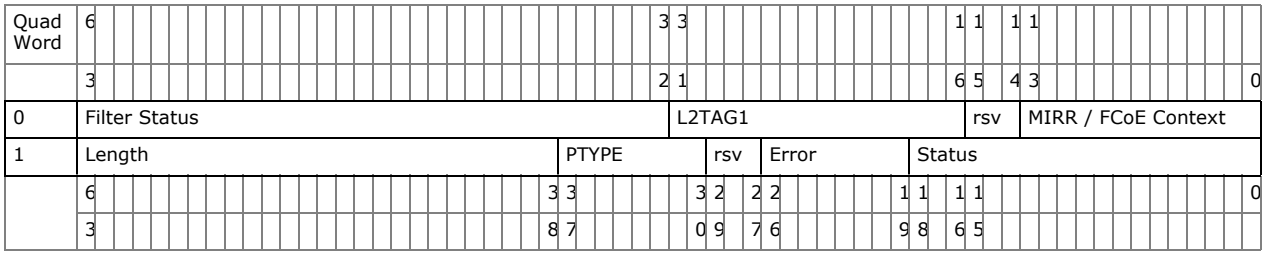
8.3.2.2 Receive Descriptor - Write Back Format

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 then all the fields are valid. See below 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) and 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) and HBO (Header Buffer overflow).
- All other fields are valid only in the last descriptor of a packet.

8.3.2.2.1 16 Byte Receive Descriptors WB Format

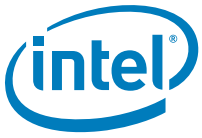
Described below is the 16 Bytes receive descriptor write back (WB) format.



RSV (7 bits) Reserved.

Status Field (Quad Word 1, 19 bits)

Bits	Name	Functionality
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_L2TAGxxx 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 L3 and L4 integrity check is processed by the hardware. In case of tunneled packet, the L3L4P relates to the inner IP header. For FCoE packets, this flag indicates that FC CRC integrity check is processed by the hardware. Processing is according to the packet type which is reported in the PTYPE field. See Section 8.3.4.3 for a description of the supported integrity checks.
4	CRCP	CRCP indicates that the Ethernet CRC is posted with data to the host buffer. Note that 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 HW computes the CRC are never posted with the CRC bytes regardless of the CRCStrip setting in the queue context.



Bits	Name	Functionality
5:7	TSYNINDX	Sampled 1588 timestamp index. • bit 7 is the TSYNINDX valid indication. Only when it is set to 1b the lower 2 bits are meaningful. • bit 5:6 is the index of the PRTTSYN_RXTIME Register that holds the packet reception timestamp. Note that 1588 packets over UDP can be sampled regardless of a possible checksum error.
8	Reserved	Reserved
9:10	UMBCAST	Destination Address can be one of the following: • 00b - Unicast • 01b - Multicast • 10b - Broadcast • 11b - Mirrored Packet Non-parsed packets are indicated by PTYPE equals to PAYLOAD (non identified MAC header).
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 RSSSTAT field below.
12:13	FLTSTAT	The FLTSTAT indicates the reported content in the "Filter Status" field. For non FCoE packets the FLTSTAT has the following encoding (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) For FCoE packet types (indicated by the PTYPE field) the FLTSTAT has the following encoding: 00b - No Data in the filter status field due to No FCoE Context Match 01b - The PARAM field from the matched context. The whole packet is posted in the LAN queue due to "Abnormal" exceptions as detailed in Section 9.4.3.2. 10b - Reserved 11b - The PARAM field from the matched context while the context is invalidated. It could be either good packet that invalidate the context or exception case (as reported by the FCE field in this descriptor). In any exception case or RSP packet, the whole packet is posted in the LAN queue. For good data packet that invalidate the context, its header is posted to the LAN queue.
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 (See additional details in Section 8.3.4.3.) If the packet contains two IPv6 headers (tunneling), the IPV6EXADD is a logic 'OR' function of the two IP headers.
16:17	Reserved	Reserved
18	INT_UDP_0	This flag is set for received UDP packets on which the UDP checksum word equals to zero. Note that 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 that for tunneled packets with UDP header, this flag relates to the checksum field in the inner UDP header.

Error Field (Quad Word 1, 8 bits)

Table 8-7. Error bits

Bits	Name	Functionality
0	RXE	The RXE error bit is an indication for any of the following MAC errors: CRC ; Alignment ; Oversize ; Undersize ; 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	RSV	Reserved
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. See Table 8-16 for details.
3:5	L3L4E / FCE	For FCoE packet types (indicated by the PTYPE field) the FCE field has the following encoding: 000b - No Errors 001b - FCoE protocol Error (FC CRC, FCoE Ver,… as detailed in Section 9.4.3.1) 010b - FCoE Filter Context Error (as detailed in Section 9.4.3.3) 011b - FCoE DMA Context Error (as detailed in Section 9.4.3.4) 100b - FCoE DMA Context Warning (as detailed in Section 9.4.3.5) Else - reserved 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 checksum error
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	RSV	Reserved

MIRR / FCoE Context (Quad Word 0, 14 bits)

- Multiplexed field between Mirroring and FCoE parameters as described in the tables below. Mirrored packets are not candidates for FCoE DDP offload so there is no conflict between the MIRR parameter and the FCoE context parameters. If the packet is not mirrored and is not processed by FCoE DDP offload then this field equals to zero.

Table 8-8. Mirroring Functionality

Bits	Name	Mirroring Functionality (UMBCAST field equals to 11b)
0:7	RSV	Reserved (equals to 0x7 for mirrored packets)
8:13	MIRR	Matched Mirror Rule ID that directed the packet to this queue.

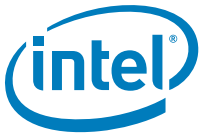
Table 8-9. FCoE Functionality

Bits	Name	FCoE Functionality (PTYPE reports FCoE packet)
0:11	FCoEINDX	Matched FCoE DDP context index (defined at the FCoE context programming). It is valid only for FCoE packets that match an FCoE context as reported by non zero FLTSTAT indication.
12:13	RSV	Reserved

L2TAG1 (Quad Word 0, 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).

Filter Status (Quad Word 0, 32 bits)



Multiplexed field between RSS (hash filter), FD Filter ID and FCoE PARAM 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 type is FCOE (reflected by the PTYPE field) and FLTSTAT is not zero, the Filter Status contains the PARAM field from the FCoE context. For the DDP context, it equals to the initial programmed PARAM value plus the total number of received data bytes.
- Else, if the packet matches a FD filter that enables its FD filter ID reporting while using a 16 byte descriptor, then FLTSTAT equals 01b and this field contains the programmed FD filter ID.
- Else, if the packet matches the Hash filter, then 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, 26 bits)

Bits	Name	Functionality
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. A step comment: This field is meaningful only when the HBO flag in the error field in this descriptor is cleared (inactive).
25	SPH	The Split Header flag is an indication that the device identified the packet header. See Section 8.3.4.2 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, 8 bits)

Packet Type field encode supported packet types as listed in the table below.

Note that in the table below, any UDP tunneling (Teredo, VXLAN) and GRE tunneling are reported as “GRENAT”.

Table 8-10. Packet Types

PTYPE	Description	PTYPE	Description
L2 Packet types			
0	Reserved	11	MAC, ARP
1	MAC, PAY2	12	MAC, FCOE, PAY3
2	MAC, TimeSync, PAY2	13	MAC, FCOE, FCDATA, PAY3
3	MAC, FIP, PAY2	14	MAC, FCOE, FCRDY, PAY3
4	Reserved	15	MAC, FCOE, FCRSP, PAY3
5	Reserved	16	MAC, FCOE, FCOTHER, PAY3
6	MAC, LLDP, PAY2	17	MAC, FCOE, VFT, PAY3
7	MAC, ECP, PAY2	18	MAC, FCOE, VFT, FCDATA, PAY3
8	Reserved	19	MAC, FCOE, VFT, FCRDY, PAY3
9	Reserved	20	MAC, FCOE, VFT, FCRSP, PAY3
10	MAC, EAPOL, PAY2	21	MAC, FCOE, VFT, FCOTHER, PAY3
Non Tunneled IPv4		**Non Tunneled IPv6**	
22	MAC, IPV4FRAG, PAY3	88	MAC, IPv6+IPV6FRAG, PAY3
23	MAC, IPV4, PAY3	89	MAC, IPV6, PAY3

Table 8-10. Packet Types (Continued)

PTYPE	Description	PTYPE	Description
24	MAC, IPV4, UDP, PAY4	90	MAC, IPV6, UDP, PAY4
25	Reserved	91	Reserved
26	MAC, IPV4, TCP, PAY4	92	MAC, IPV6, TCP, PAY4
27	MAC, IPV4, SCTP, PAY4	93	MAC, IPV6, SCTP, PAY4
28	MAC, IPV4, ICMP, PAY4	94	MAC, IPV6, ICMP, PAY4
IPv4 --> IPv4		**IPv4 --> IPv6**	
29	MAC, IPV4, IPV4FRAG, PAY3	36	MAC, IPV4, IPv6+IPV6FRAG, PAY3
30	MAC, IPV4, IPV4, PAY3	37	MAC, IPV4, IPV6, PAY3
31	MAC, IPV4, IPV4, UDP, PAY4	38	MAC, IPV4, IPV6, UDP, PAY4
32	Reserved	39	Reserved
33	MAC, IPV4, IPV4, TCP, PAY4	40	MAC, IPV4, IPV6, TCP, PAY4
34	MAC, IPV4, IPV4, SCTP, PAY4	41	MAC, IPV4, IPV6, SCTP, PAY4
35	MAC, IPV4, IPV4, ICMP, PAY4	42	MAC, IPV4, IPV6, ICMP, PAY4
IPv4 --> GRE/Teredo/VXLAN			
43	MAC, IPV4, GRENAT, PAY3		
IPv4 --> GRE/Teredo/VXLAN --> IPv4		**IPv4 --> GRE/Teredo/VXLAN --> IPv6**	
44	MAC, IPV4, GRENAT, IPV4FRAG, PAY3	51	MAC, IPV4, GRENAT, IPv6+IPV6FRAG, PAY3
45	MAC, IPV4, GRENAT, IPV4, PAY3	52	MAC, IPV4, GRENAT, IPV6, PAY3
46	MAC, IPV4, GRENAT, IPV4, UDP, PAY4,	53	MAC, IPV4, GRENAT, IPV6, UDP, PAY4,
47	Reserved	54	Reserved
48	MAC, IPV4, GRENAT, IPV4, TCP, PAY4	55	MAC, IPV4, GRENAT, IPV6, TCP, PAY4
49	MAC, IPV4, GRENAT, IPV4, SCTP, PAY4	56	MAC, IPV4, GRENAT, IPV6, SCTP, PAY4
50	MAC, IPV4, GRENAT, IPV4, ICMP, PAY4	57	MAC, IPV4, GRENAT, IPV6, ICMP, PAY4
IPv4 --> GRE/Teredo/VXLAN --> MAC			
58	MAC, IPV4, GRENAT, MAC, PAY3		
IPv4 --> GRE/Teredo/VXLAN --> MAC --> IPv4		**IPv4 --> GRE/Teredo/VXLAN --> MAC --> IPv6**	
59	MAC, IPV4, GRENAT, MAC, IPV4FRAG, PAY3	66	MAC, IPV4, GRENAT, MAC, IPv6+IPV6FRAG, PAY3
60	MAC, IPV4, GRENAT, MAC, IPV4, PAY3,	67	MAC, IPV4, GRENAT, MAC, IPV6, PAY3,
61	MAC, IPV4, GRENAT, MAC, IPV4, UDP, PAY4	68	MAC, IPV4, GRENAT, MAC, IPV6, UDP, PAY4
62	Reserved	69	Reserved
63	MAC, IPV4, GRENAT, MAC, IPV4, TCP, PAY4	70	MAC, IPV4, GRENAT, MAC, IPV6, TCP, PAY4
64	MAC, IPV4, GRENAT, MAC, IPV4, SCTP, PAY4	71	MAC, IPV4, GRENAT, MAC, IPV6, SCTP, PAY4
65	MAC, IPV4, GRENAT, MAC, IPV4, ICMP, PAY4	72	MAC, IPV4, GRENAT, MAC, IPV6, ICMP, PAY4
IPv4 --> GRE/Teredo/VXLAN --> MAC/VLAN			
73	MAC, IPV4, GRENAT, MACVLAN, PAY3		
IPv4 --> GRE/Teredo/VXLAN --> MAC/VLAN --> IPv4		**IPv4 --> GRE/Teredo/VXLAN --> MAC/VLAN --> IPv6**	
74	MAC, IPV4, GRENAT, MACVLAN, IPV4FRAG, PAY3,	81	MAC, IPV4, GRENAT, MACVLAN, IPv6+IPV6FRAG, PAY3
75	MAC, IPV4, GRENAT, MACVLAN, IPV4, PAY3,	82	MAC, IPV4, GRENAT, MACVLAN, IPV6, PAY3,
76	MAC, IPV4, GRENAT, MACVLAN, IPV4, UDP, PAY4	83	MAC, IPV4, GRENAT, MACVLAN, IPV6, UDP, PAY4
77	Reserved	84	Reserved

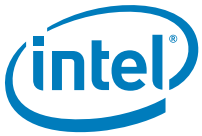


Table 8-10. Packet Types (Continued)

PTYPE	Description	PTYPE	Description
78	MAC, IPV4, GRENAT, MACVLAN, IPV4, TCP, PAY4	85	MAC, IPV4, GRENAT, MACVLAN, IPV6, TCP, PAY4
79	MAC, IPV4, GRENAT, MACVLAN, IPV4, SCTP, PAY4	86	MAC, IPV4, GRENAT, MACVLAN, IPV6, SCTP, PAY4
80	MAC, IPV4, GRENAT, MACVLAN, IPV4, ICMP, PAY4	87	MAC, IPV4, GRENAT, MACVLAN, IPV6, ICMP, PAY4
IPv6 --> IPv4		**IPv6 --> IPv6**	
95	MAC, IPV6, IPV4FRAG, PAY3	102	MAC, IPV6, IPv6+IPV6FRAG, PAY3
96	MAC, IPV6, IPV4, PAY3	103	MAC, IPV6, IPV6, PAY3
97	MAC, IPV6, IPV4, UDP, PAY4	104	MAC, IPV6, IPV6, UDP, PAY4
98	Reserved	105	Reserved
99	MAC, IPV6, IPV4, TCP, PAY4	106	MAC, IPV6, IPV6, TCP, PAY4
100	MAC, IPV6, IPV4, SCTP, PAY4	107	MAC, IPV6, IPV6, SCTP, PAY4
101	MAC, IPV6, IPV4, ICMP, PAY4	108	MAC, IPV6, IPV6, ICMP, PAY4
IPv6 --> GRE/Teredo/VXLAN			
109	MAC, IPV6, GRENAT, PAY3		
IPv6 --> GRE/Teredo/VXLAN --> IPv4		**IPv6 --> GRE/Teredo/VXLAN --> IPv6**	
110	MAC, IPV6, GRENAT, IPV4FRAG, PAY3	117	MAC, IPV6, GRENAT, IPv6+IPV6FRAG, PAY3
111	MAC, IPV6, GRENAT, IPV4, PAY3	118	MAC, IPV6, GRENAT, IPV6, PAY3
112	MAC, IPV6, GRENAT, IPV4, UDP, PAY4	119	MAC, IPV6, GRENAT, IPV6, UDP, PAY4
113	Reserved	120	Reserved
114	MAC, IPV6, GRENAT, IPV4, TCP, PAY4	121	MAC, IPV6, GRENAT, IPV6, TCP, PAY4
115	MAC, IPV6, GRENAT, IPV4, SCTP, PAY4	122	MAC, IPV6, GRENAT, IPV6, SCTP, PAY4
116	MAC, IPV6, GRENAT, IPV4, ICMP, PAY4	123	MAC, IPV6, GRENAT, IPV6, ICMP, PAY4
IPv6 --> GRE/Teredo/VXLAN --> MAC			
124	MAC, IPV6, GRENAT, MAC, PAY3		
IPv6 --> GRE/Teredo/VXLAN --> MAC --> IPv4		**IPv6 --> GRE/Teredo/VXLAN --> MAC --> IPv6**	
125	MAC, IPV6, GRENAT, MAC, IPV4FRAG, PAY3	132	MAC, IPV6, GRENAT, MAC, IPv6+IPV6FRAG, PAY3
126	MAC, IPV6, GRENAT, MAC, IPV4, PAY3	133	MAC, IPV6, GRENAT, MAC, IPV6, PAY3
127	MAC, IPV6, GRENAT, MAC, IPV4, UDP, PAY4	134	MAC, IPV6, GRENAT, MAC, IPV6, UDP, PAY4
128	Reserved	135	Reserved
129	MAC, IPV6, GRENAT, MAC, IPV4, TCP, PAY4	136	MAC, IPV6, GRENAT, MAC, IPV6, TCP, PAY4
130	MAC, IPV6, GRENAT, MAC, IPV4, SCTP, PAY4	137	MAC, IPV6, GRENAT, MAC, IPV6, SCTP, PAY4
131	MAC, IPV6, GRENAT, MAC, IPV4, ICMP, PAY4	138	MAC, IPV6, GRENAT, MAC, IPV6, ICMP, PAY4
IPv6 --> GRE/Teredo/VXLAN --> MAC/VLAV			
139	MAC, IPV6, GRENAT, MACVLAN, PAY3		
IPv6 --> GRE/Teredo/VXLAN --> MAC/VLAN --> IPv4		**IPv6 --> GRE/Teredo/VXLAN --> MAC/VLAN --> IPv6**	
140	MAC, IPV6, GRENAT, MACVLAN, IPV4FRAG, PAY3,	147	MAC, IPV6, GRENAT, MACVLAN, IPv6+IPV6FRAG, PAY3
141	MAC, IPV6, GRENAT, MACVLAN, IPV4, PAY3,	148	MAC, IPV6, GRENAT, MACVLAN, IPV6, PAY3,
142	MAC, IPV6, GRENAT, MACVLAN, IPV4, UDP, PAY4	149	MAC, IPV6, GRENAT, MACVLAN, IPV6, UDP, PAY4
143	Reserved	150	Reserved
144	MAC, IPV6, GRENAT, MACVLAN, IPV4, TCP, PAY4	151	MAC, IPV6, GRENAT, MACVLAN, IPV6, TCP, PAY4

Table 8-10. Packet Types (Continued)

PTYPE	Description	PTYPE	Description
145	MAC, IPV6, GRENAT, MACVLAN, IPV4, SCTP, PAY4	152	MAC, IPV6, GRENAT, MACVLAN, IPV6, SCTP, PAY4
146	MAC, IPV6, GRENAT, MACVLAN, IPV4, ICMP, PAY4	153	MAC, IPV6, GRENAT, MACVLAN, IPV6, ICMP, PAY4
Parser Aborted Code		**Else**	
255	Parser Aborted	Else	Reserved

8.3.2.2.2 32 Byte Receive Descriptors WB Format

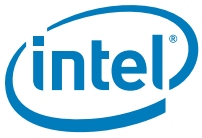
The 32-byte descriptor is composed of 4 quad words. The first two quad words are identical to the 16 byte descriptor write back other than the Flow Director filter ID that is moved from the "Filter Status" field to the "FD Filter ID / Flexible Bytes High" field.

Quad Word	63:44	43:33	32:22	21:11	10:0
0	Filter Status		L2TAG1	rsv	MIRR / FCoE Context
1	Length	PTYPE	rsv	Error	Status
2	L2TAG2 (2nd)	L2TAG2 (1st)	rsv		Ext_Status
3	FD Filter ID / Flexible Bytes High		Flexible Bytes Low		

	63:44	43:33	32:21	20:11	10:0

Ext_Status (Quad Word 2, 20 bits)

Bits	Name	Functionality
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_L2TAGxxx registers. If the stripped Tag is larger than one word, then the first word is posted to the "L2TAG2 (1st)" field and the second word is posted to the "L2TAG2 (2nd)" field. If the specified L2 tag is not present in the packet the L2TAG2P flag is cleared.
1	RSV	Reserved
2:3	FLEXBL_STAT	The FLEXBL_STAT indicates the content of the "Flexible Bytes Low" field as follows: 00b - The field is set to zero 01b - Flexible first 4 bytes defined by the FD filter are posted in the "Flexible Bytes Low" 10b - Reserved 11b - Reserved
4:5	FLEXBH_STAT	The FLEXBH_STAT 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 / Flexible Bytes High" field 10b - Flexible second 4 bytes defined by the FD filter are posted in the "FD Filter ID / Flexible Bytes High" field 11b - Reserved



Bits	Name	Functionality
6:8	RSV	Reserved
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 MAXFDBLEN parameter in the GLQF_CTL register.
10	FCOELONGB	The FCOELONGB flag is set if the matched FCoE context is found in the cache or the filter's index within its bucket is above threshold. If the packet is searched in the FCoE context filter and it is not found, the FCOELONGB reflects the bucket length relative to the same threshold. The threshold is defined by the MAXFCBLEN parameter in the GLQF_CTL register.
11	RSV	Reserved

L2 Tags (Quad Word 2, 32 bits)

Bits	Name	Functionality
0:15	L2TAG2 (1st)	Extracted L2 Tag 2 from the packet (see L2TAG2P flag).
16:31	L2TAG2 (2nd)	Extracted second word of the L2 Tag 2 from the packet (see L2TAG2P flag).

Flexible Bytes Low (Quad Word 3, 32 bits)

If the packet matches a FD filter that enables reporting flexible bytes from the packet, then FLEXBL_STAT equals 01b. Otherwise, FLEXBL_STAT equals 00b and this field is set to zero.

Bits	Name	FD filter flexible bytes (FLEXBL_STAT = 01b)
0:31	FLEXBL	This field contains 4 bytes from the "Flexible Payload" in the "Field Vector" (extracted from the packet). The word offset (in the "Field Vector") of these bytes are defined by the programmed FLEXOFF parameter of the FD filter. The reported bytes in this field are defined in "little endian" notation while bits 0:7 are the LS byte which is the last byte on the wire.

FD Filter ID / Flexible Bytes High (Quad Word 3, 32 bits)

Multiplexed field between the FD Filter ID and Flexible Bytes High, as follows:

- 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.
- Else if the packet matches a FD filter that enables reporting 8 flexible bytes from the packet, then FLEXBL_STAT equals 10b and this field contains the 4 bytes from the "Flexible Payload" in the "Field Vector" (extracted from the packet). The reported bytes in this field are defined in "little endian" notation while bits 24:31 are the MS byte of the whole 8-byte field which is the first byte on the wire.
- Else, FLEXBH_STAT equals 00b and this field is set to zero.

8.3.2.2.3 Programming Status Descriptor WB Format

The programming status descriptor provides an indication of FD filter programming and FCoE context programming as described in the table below. Programming status vs. packet's descriptor is identified by Length field set to 0x2000000. The programming status is provided by a 16 byte structure described below. When using 32 byte descriptors, only the first 16 bytes are meaningful while the last 16 bytes are reserved zeros.

Context Type	Action	Fail/OK	Programming Indication	Reported Receive Queue
FD Filter	Add Filter	OK	No reporting	N/A
FD Filter	Add Filter	Fail	Programming Status Descriptor	Queue zero of the PF
FD Filter	Remove Filter	OK	No reporting	N/A
FD Filter	remove Filter	Fail	Programming Status Descriptor	Queue zero of the PF
DDP Context	Add Context	OK	No reporting	N/A
DDP Context	Add Context	Fail	Programming Status Descriptor	Queue zero of the VSI
DDP Context	Invalidate Context	OK	Programming Status Descriptor	Queue zero of the VSI
DDP Context	Invalidate Context	Fail	Programming Status Descriptor	Queue zero of the VSI

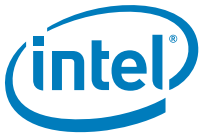
<table>
<tr><td>Quad Word</td><td colspan="2">63 … 32</td><td colspan="4">31 … 0</td></tr>
<tr><td>0</td><td colspan="2">Filter Status</td><td colspan="4">rsv</td></tr>
<tr><td>1</td><td>Length = 0x2000000</td><td colspan="2">rsv</td><td>Error</td><td>rsv</td><td>Status</td></tr>
<tr><td></td><td>63 … 38</td><td colspan="2">37 … 25</td><td>24 … 19</td><td>18 … 15</td><td>14 … 0</td></tr>
</table>

Status Field (Quad Word 1, 15 bits)

Status field

Bits	Name	Functionality
0	DD	Descriptor done indication flag.
1	RSV	Reserved
2:4	PROG_ID	The PROG_ID indicates the status reported by this descriptor as follows: 001 - FD filter status 010b - FCoE context programming status 100b - FCoE context invalidation status
5:14	RSV	Reserved

Error Field (Quad Word 1, 6 bits)



Bits	Name	Functionality
0	Failed FD Programming	FD filter programming failed. It could be due to no space in the FD table or since the function exhausted its quota.
1	Failed FD Removal	FD filter removal failed. It could be a result of an attempt to remove non-existent entry.
2	FCoE Table Full	FCoE context programming rejected since there is no room in the FCoE table.
3	FCoE Conflict / Failed FCoE Removal	The functionality of this flag depends on the PROG_ID value in the Status Field: For "FCoE context programming" - it is an indication of FCoE context programming rejected since it conflicts with existing active FCoE context. For "FCoE context invalidation" - it is an indication of failed context invalidation since a matched context was not found.
4:5	RSV	Reserved

Filter Status (Quad Word 0, 32 bits)

Equals to the “FD Filter ID” for FD filter programming status. For FCoE context programming status the lower 12 bits equals to the "DDP Context Index" and the higher bits of the Filter Status are reserved (equals to zero).

8.3.3 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”. XL710 supports up to 1536 receive queues allocated to PFs and VFs (see Section 8.2). Receive queues are defined by a set of parameters called the “queue context”. The main parameters are the queue pointers presented in Figure 8-6. The queue context includes additional parameters that define the queue functionality as detailed in Section 8.3.3.2. Part of these context parameters are kept in hardware (like the Tail register, queue enable / disable flags and interrupt related context). Most of the queue context parameters are stored in Function Private Memory (FPM), fetched to an internal cache when required.

The software interface to the queue for its initialization, during nominal operation as well as queue disable flow, is described in Section 8.3.3.1 . XL710 includes additional global setting option parameters for the whole device or per function that affect multiple queues described in Section 8.3.3.1.

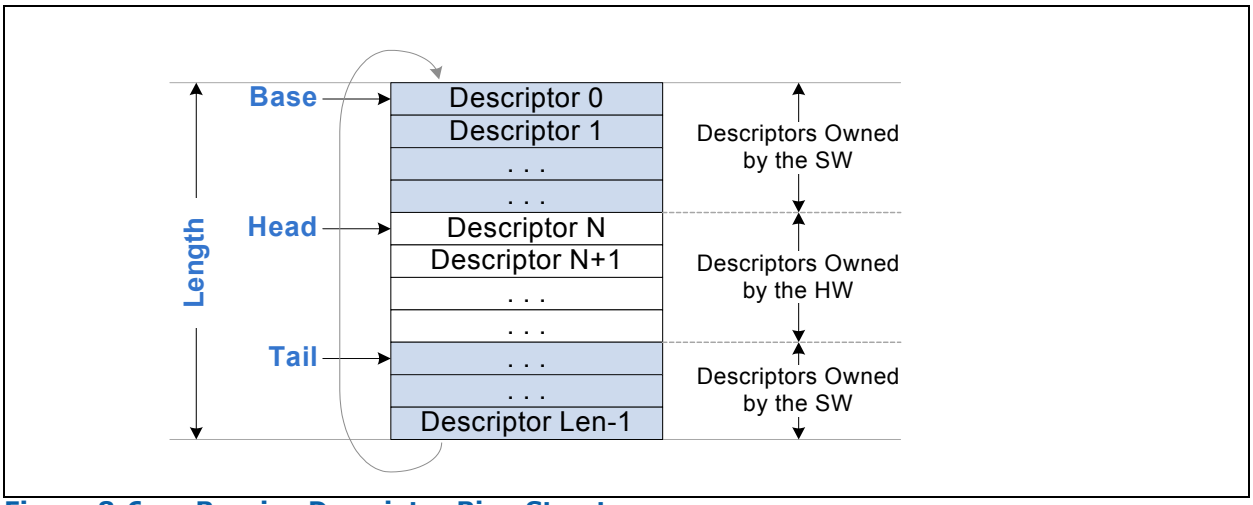


Figure 8-6. Receive Descriptor Ring Structure

8.3.3.1 Receive Queue Programming

Queue enable and disable flows are described in the following subsection and summarized in the Table 8-11 below.

Table 8-11. Receive Queue Enable / Disable Flags

QRX_ENA[n] register		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.

8.3.3.1.1 Receive Queue Enable Flow

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 8.2.1.2.

- Software steps
- 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.
- If the queue might have been disabled recently, then the software should wait at least 50msec from the completion of the queue disable flow before proceeding to the next step.
- Prepare the queue context in the FPM in the PF memory space. See a examples for queue context setting in Section 8.3.3.2.3. At pre-boot time, memory allocation might be tight and the FPM could exceed the budget. There is a mechanism to program the queue context directly to the device bypassing the FPM as described in Section 8.3.3.1.1.1.

- Clear the Tail pointer in the QRX_TAIL[n] register and then set the Tail pointer to the end of the descriptor ring (‘n’ is the queue index within the PF space).
- Set the QENA_REQ flag in the QRX_ENA[n] register (‘n’ is the queue index within the PF space).
- As a response, the hardware sets the QENA_STAT flag in the QRX_ENA[n] register. The QENA_STAT follows the QENA_REQ almost instantly and not more than 10usec after that. Once the QENA_STAT flag in the QRX_ENA[n] register is set, software can start using the queue.
- If the queue is targeted for a VF, PF software should also program the matched entry in the VFQ_TABLE. Then it is expected to inform the VF software that the queue is enabled. Note that the VFQ_TABLE is shared for the receive and transmit queues. So the PF software should enable the transmit queue before communicating to the VF.

8.3.3.1.1.1 Direct Queue Context Programming

This section describes direct queue context programming, bypassing the FPM for pre-boot flow.

Context Programming Flow:

- Write the four PFCM_LANCTXDATA registers with a total of 16 bytes of context data.
- Write the PFCM_LANCTXCTL register identifying the queue and the specific sub-line to be programmed.
- Poll the CTX_DONE flag in the PFCM_LANCTXSTAT register till “done” is indicated.
- Loop back to the first step until all sub-lines of the queue context are programmed.

Context Invalidation Flow (this flow is not needed in nominal operation):

- Write the following fields in the PFCM_LANCTXCTL register: QUEUE_NUM and QUEUE_TYPE identify the queue and the OP_CODE fields should be set to “Invalidate”.
- Poll the CTX_DONE flag in the PFCM_LANCTXSTAT register till “done” is indicated.

8.3.3.1.2 Receive Queue Disable Flow

Queue disable flow is executed by PF software for PF queues as well as for the queues of its VFs.

- Steps for receive queue which is used for FCoE traffic:
  - The state of a receive queue (which is used for FCoE traffic) affects the state of all FCoE contexts that are associated with it. Disabling such a receive queue will also abort all active FCoE contexts that are associated with the queue. These FCoE contexts are aborted without any indication to software about the progress of the abortion flow. Therefore, before disabling the queue, software should first disable all associated FCoE contexts.
- Remove the queue from the interrupt linked list as described in Section 7.5.3.1.3.
- The PF software clears the QENA_REQ flag in the QRX_ENA[n] register, while ‘n’ is the queue index within the PF space.
- The hardware generates a “queue disable” marker to the receive “pipe”.
- 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.
- The hardware waits for completion of all outstanding requests from the specific queue on the PCIe bus.
- The queue context is invalidated from the internal cache without updating the FPM and the QENA_STAT flag in the QRX_ENA[n] register is cleared.
- Once the QENA_STAT flag in the QRX_ENA[n] register is cleared, software can release all memory structures of the queue.

8.3.3.1.3 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 allthis automatically .

- It is assumed that a VFR was initiated by the PF software and the matched VFRD flag in the VPGEN_VFRSTAT 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.
- The PF software sets the FAST_QDIS flag (and clear the QENA_REQ flag) in the QRX_ENA[n] register where 'n' is the queue indexes in the PF space for all VF or VM queues.
- The queue contexts are invalidated instantly from the internal cache and the QENA_STAT flag in the matched QRX_ENA[n] register is cleared.

8.3.3.1.4 Software Fast Path Programming

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 descriptor index incremented the next time).
- 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 Table 8-12.

8.3.3.2 Receive Queue Context Parameters

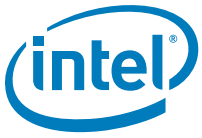
This section describes setting options of LAN receive queue parameters called the "queue context". Some queue context parameters reside in dedicated hardware registers and some are stored in host memory (FPM). Active queues are kept in cache.

8.3.3.2.1 Receive Queue Context in Hardware Registers

The queue context parameters that software accesses during fast past programming as well as its enablement reside in hardware registers:

- Queue enablement flags in the QRX_ENA[n] registers while 'n' is the queue index of the PF. Queue enable and disable flow by the PF are explained in Section 8.3.3.1.
- The TAIL pointer in the QRX_TAIL[n] registers for the PF and VFQRX_TAIL[n] registers for the VFs while 'n' is the queue index of the function (PF or VF). The usage of the Tail register is detailed in Section 8.3.3.1.3

8.3.3.2.2 Receive Queue Context in FPM



The receive queue context parameters that are stored in host memory (FPM) are fetched for the active queues to the internal cache. These parameters are described in the Table 8-12 below. The Queue context is a contiguous vector of 256 bits (32 bytes).

Table 8-12. LAN Rx Queue Context in the Private Host Memory(174 bits = 25 Byte)

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 (as shown in Figure 8.1). 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.
RSV	11	21	31	N/A	0x0	Reserved
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 8 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 Up to 16 PFs, QLEN can be set to: 8 descriptors
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 1K bytes and up to 16K 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 2KB minus 64B.
DType	2	114	115	Static	DType	Descriptor Type as defined in Table 8-13.
DSize	1	116	116	Static	DSize	Descriptor Size flag. '0' for 16B descriptors and '1' for 32B descriptors
CRCStrip	1	117	117	Static	CRCStrip	CRC Strip. Strip the Ethernet CRC bytes before the packet is posted to host memory. Note that no CRC Strip option may work properly only if the whole packet is posted to the data buffer(s) in host memory with no other strip option.
FCENA	1	118	118	Static	FCENA	Enable FCoE packet reception to the queue. If the FCENA flag is cleared, received FCoE and FIP packets to this queue are dropped and counted by the GLV_REPC counter of the VSI.
L2TSEL	1	119	119	Static	0x0	The L2TSEL defines the reported L2 Tags in the receive descriptor. When the L2TSEL flag is cleared, 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. When the L2TSEL flag is set to 1b, the above L2 tags are switched between the L2TAG1 and L2TAG2 fields.
HSPLIT_0	4	120	123	Static	HSPLIT_0	Header Split 0 control as described in Table 8-14
HSPLIT_1	2	124	125	Static	HSPLIT_1	Header Split 1 control as described in Table 8-15

Table 8-12. LAN Rx Queue Context in the Private Host Memory(174 bits = 25 Byte)

Alias	Width [bits]	LS bit	MS bit	Type	SW Init	Description
RSV	1	126	126	N/A	0x0	Reserved
SHOWIV	1	127	127	Static	SHOWIV	The VLAN in the inner L2 header is stripped to the receive descriptor if enabled by this flag.
RSV	46	132	173	Static	0x0	Reserved
RXMAX	14	174	187	Static	RXMAX	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 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 5 buffers). Received packet larger than RXMAX is dropped and counted by the GLV_REPC counter of the VSI. Note that packets larger than MFS defined per MAC are dropped by the MAC even before it gets to the receive queue.
RSV	5	188	192	Static	0x0	Reserved
TPHRDesc	1	193	193	Static	TPHRDesc	Read Descriptor TPH Ena (descriptor fetch).
TPHWDesc	1	194	194	Static	TPHWDesc	Write Descriptor TPH Ena (descriptor writeback).
TPHData	1	195	195	Static	TPHData	Packet Data TPH Ena.
TPHHead	1	196	196	Static	TPHHead	Packet Header TPH Ena.
RSV	1	197	197	Static	0x0	Reserved
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.
RSV	55	201	255	Static	0x0...01	Reserved

8.3.3.2.3 Examples for Receive Queue Context Setting

Listed below are several examples for receive queue context settings.

Control Port Receive Queue Context Example =

DW Offset: 0 1 2 3 4 5 6 7

Bit offset: 31 0 63 32 95 64 127 96 159 128 191 160 223 192 255 224

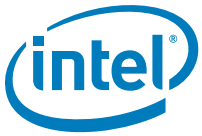
0x00000000 001579A0 00000000 00200301 00000000 01800000 0000021E 00000000

BASE = 0x001579A0	DSize = 0 (16 bytes)	TPH = 0xF (enable all TPHxxx flags)
QLEN = 0x80 (128 descriptors)	HSPLIT = 0 (no header split)	LRXQTRESH = 0 (no low threshold)
DBUFF = 12 (1536 bytes)	RXMAX = 0x600 (1536)	CRCStrip = 1
HBUFF = 0		FCENA = 0
DType = 00b (no header split)		

FCoE Receive Queue Context Example =

LAN Receive Queue Context Example (no Jumbo) =

DW Offset: 0 1 2 3 4 5 6 7



Bit offset: 31 0 63 32 95 64 127 96 159 128 191 160 223 192 255 224

0x00000000 0BACD000 00000000 00B00449 00000000 02190000 0000029E 00000000

BASE = 0x00BACD000	DSize = 1 (32 bytes)	TPH = 0xF (enable all TPHxxx flags)
QLEN = 0x400 (1024 descriptors)	HSPLIT = 0 (no split|)	LRXQTRESH = 2 (128 descriptors)
DBUFF = 17 (2176 bytes)	RXMAX = 0x864 (2148 bytes)	CRCStrip = 1
HBUFF = 0		FCENA = 1
DType = 00b (no split)		

8.3.3.2.4 Examples for Receive Buffer Size Setting

Listed below are the most common used buffer size settings used by LAD drivers.

OS	Queue type	HSPLIT	DBUFF	HBUFF	Notes
Linux	Standard	0x0	1536	-	
	Standard	enabled	2048	256	
	Jumbo	0x0	multiple of 1K	-	unlikely to use this configuration
	Jumbo	enabled	2048	256	
	FCoE	0x0	3072	-	
Windows	Any	0x0	multiple of 1K	-	up to 10K
	FCoE	0x0	3072	-	
	VMQ	enabled	2048	64	L2 header split
FreeBSD	Any	0x0	2K, 4K, 8K, or 10K	-	
	Any	enabled	2K, 4K, 8K, or 10K	256	
ESX NPA	Any	enabled	4096	1984	max header buffer
Solaris	Any	0x0	256, 1K, 2K	-	
XEN					same as Linux
KVM					same as Linux

8.3.4 Stateless Receive Offloads

8.3.4.1 Strip Ethernet CRC bytes

See Section 7.2.1.2.

8.3.4.2 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 size 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 may apply to these buffers or different rules for TPH enablement. Header split is enabled per receive queue by the DTYPE and HSPLIT_0 and HSPLIT_1 fields in the receive queue context as described in the Table 8-13, Table 8-14 and Table 8-15 and illustrated in Figure 8-7. Split between the header buffer and the payload buffers and the status reporting is detailed in Table 8-16. 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 that 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 on 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, then the packet is split according to HSPLIT_1 regardless of HSPLIT_0 setting.
- 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, then 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 buffers. If the header buffer is smaller than the received header, the header is posted together with the packet payload. See Table 8-16.
- 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 including the fragmented extension header.
- When a packet is replicated to multiple receive queues, the packet can be split differently on each according to the queues' settings.
- Header split is supported for packets received from the network as well as local VM to VM traffic.

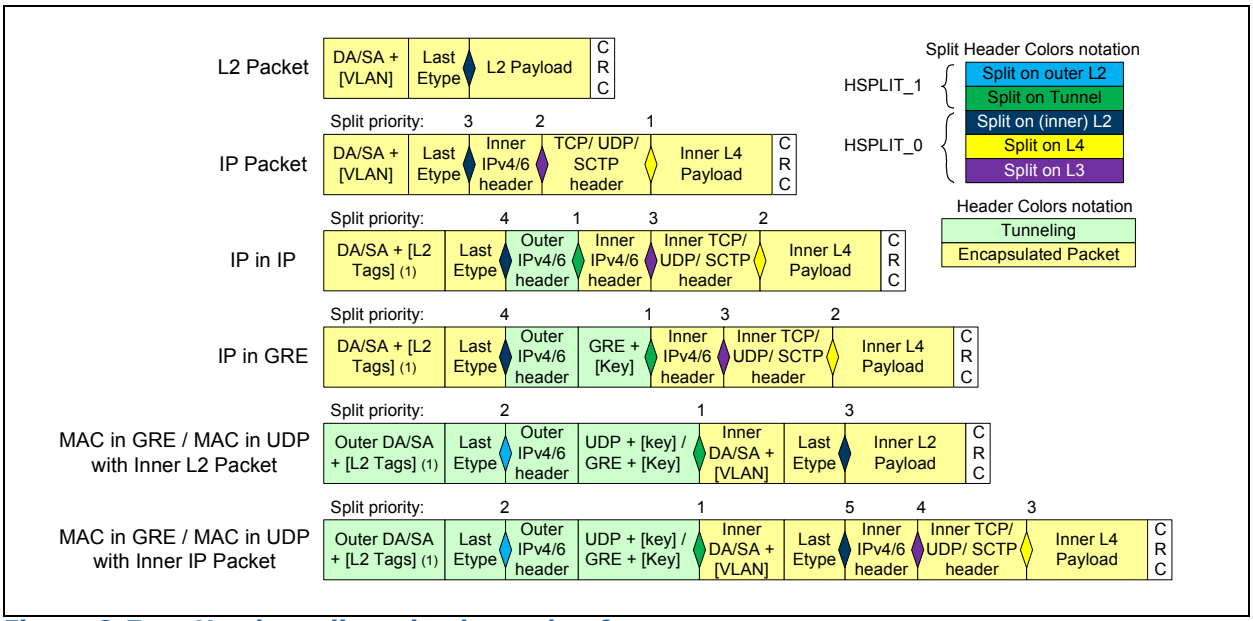


Figure 8-7. Header split option by packet formats

Table 8-13. 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 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 8-16 for rules of the header and payload buffers usage.
10b Split always	Header split always is enabled regardless of the HSPLIT setting. If the packet header is identified as defined by the HSPLIT, the packet content up to including the most inner header enabled by the HSPLIT is posted to the header buffer. See Table 8-16 for rules of the header and payload buffers usage.
11b	Reserved

Table 8-14. 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
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 8-15. Split Headers enabled by the HSPLIT_1 field

HSPLIT_1	Functionality
00b	No split on tunneling headers.
~~1x~~x1b	Enable split after the outer (tunneling) L2 header.
1xb	Enable split on the tunneling header as follows: - Non tunneled packet - No impact - IP in IP - Enable split after the outer IP header - IP in GRE or MAC in GRE - Enable split after the GRE header - MAC in UDP - Enable split after the tunneling UDP header (including its private UDP header)

Table 8-16. Header Split vs. packet and buffer sizes

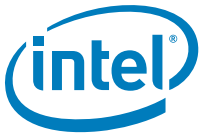
DTYPE	Condition	Header and Payload DMA	SPH	HBO	PKTL (1)	HDRL (1) (2)
00b - Single buffer	None	Post the whole packet to the packet buffer(s)	0	0	Size of the whole packet	0x0
01b - Header split	Header is not identified	Post the whole packet to the packet buffer(s)	0	0	Size of the whole packet	0x0
	Header size <= HBUFF (2)	Post header to header buffer and payload to the packet buffer(s)	1	0	Size of the packet payload	Header size
	Header size > HBUFF (2)	Post the whole packet to the packet buffer(s)	1	1	Size of the whole packet	Header size
10b Split Always	Packet length <= HBUFF	Post the whole packet to the header buffer(s)	0	0	0x0	Packet length
	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
	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
	Header size > HBUFF	Post the whole packet to the header + packet buffer(s)	1	1	Size of the whole packet minus HBUFF	Header size

Note:

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.
2. For the sake of the split conditions, the packet and header lengths are taken from the received packets including all L2 tags (VLAN for example) as well as the CRC bytes. Regardless if these tags are posted to the buffers or posted to the receive descriptors or extracted from the packet. Orthogonal to this condition, the reported HDRL parameter represents the actual data posted to the buffer.

8.3.4.3 Receive L3 and L4 Integrity Check Offload

The XL710 offloads the following L3 and L4 integrity checks: IPv4 header(s) checksum, Inner TCP or UDP checksum and SCTP CRC integrity (see Table 8-17). The XL710 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 checksum error are reported by the IPE, L4E and the EIPE error flags in the receive descriptor respectively.



Some rules for integrity check offload are listed below. 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 may 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 erroneous value. XL710 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 - XL710 parse fragmented receive packets up to including the IP header (for IPv4) or up to 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 may have any option headers that fit within the maximum header size (60 bytes).
- Header with fixed size: IPv6, UDP, SCTP common header.
- VM to VM loopback traffic is processed by the hardware for L3/L4 integrity check as any other packet received from the network.

Table 8-17 below 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. Zero UDP checksum for IPv6 packet is illegal and is reported by the hardware as L4 checksum error.

Table 8-17. Integrity Offload check for receive packet types

Packet Type	Supported Integrity Offload	Reported L3L4P
IP -> [data / Unknown / fragmented]	IP checksum offload	1 (for IPv4) / 0 (for IPv6)
IP -> L4	IP and L4 checksum offload	1
IP -> IP -> L4	IP and L4 checksum offload	1
IP -> [tunnel header] -> IP -> L4 (2)	IP and L4 checksum offload (1)	1
IP -> [tunnel header] -> IP -> data / Unknown / fragmented (2)	Only IP checksum offload.	1

Table 8-17. Integrity Offload check for receive packet types

Packet Type	Supported Integrity Offload	Reported L3L4P
[MAC in UDP / MAC in GRE] -> IP -> L4	IP and L4 checksum offload (1)	1
[MAC in UDP / MAC in GRE] -> IP -> data / unknown / fragmented	IP checksum (relevant only for IPv4)	1 (for IPv4) / 0 (for IPv6)
(1) The offloaded L4 protocol is the inner header. XL710 does not check the integrity of the tunneling UDP or GRE headers. The L4 checksum covers the following: - 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 (2) Tunneling headers could be one of the following: GRE, Teredo, VXLAN UDP header		

8.4 LAN Transmit Data-Path

The LAN Transmit Data-Path section covers the following major topics:

- Transmit packets stored in system memory and indicated to the hardware by descriptors
- Transmit descriptor queues which are called also “descriptor rings” and Transmit arbitration queue lists
- Stateless transmit offloads

8.4.1 Transmit Packet in System Memory

Transmit packets are made up of data buffers in host memory indicated to the hardware by descriptors (16 byte structures described in Section 8.4.2.2). These descriptors include pointer and length pairs to the data buffers as well as control fields for the transmit data processing. Figure 8-8 shows an example of a transmit packet in host memory composed of 2 buffers (header buffer and payload buffer), indicated by 2 matched “data descriptors”.

A few rules related to the transmit packet in host memory are:

- 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 may span up to 8 buffers (up to 8 data descriptors per packet including both the header and payload buffers).
- The total number of data descriptors for the whole TSO (explained later on in this chapter) is unlimited as long as each segment within the TSO obeys the previous rule (up to 8 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.

8.4.2 Transmit Descriptors

This section describes the descriptors provided on the LAN transmit queues. The XL710 supports the following Transmit descriptor types: Transmit Data Descriptor; Context Descriptor; FCoE Data Descriptors in the LAN queue; FCoE context descriptors; FCoE Programming descriptors; Filters programming descriptors. The following sections describe these descriptors.

Note: For all descriptors: fields indicated as 'rsv' (reserved) should be set to zero by the software at programming time.

8.4.2.1 General Descriptors

8.4.2.1.1 Transmit Data Descriptor

Quad Word	63					0
0	Tx Packet Buffer Address					
1	L2 Tag 1	Tx Buffer Size	Offset	rsv	CMD	DTYP
	63 48	47 34	33 16	15 14	13 4	3 0

Descriptor Type - DTYP (Quad Word 1, bits 0:3) 0x0 stands for a Transmit Data Descriptor

Command Field - CMD (Quad Word 1, bits 4:13)

Bits	Name	Functionality
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 setting the DTYP field to 0xF or by head write back as configured by the HEAD_WBEN flag in the transmit context. The RS flag can be set only on a last Transmit Data Descriptor of a packet or last Transmit Data Descriptor of a TSO or last Transmit Data Descriptor of a filter or FCoE context programming.
2	RSV	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_L2TAGxxx registers. If the type of the L2 Tag requires more than 2 variable bytes, then 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: FCoE and 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.

Bits	Name	Functionality
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. 00 - non IP packet or packet type is not defined by software 01 - IPv6 packet 10 - IPv4 packet with no IP checksum offload 11 - IPv4 packet with IP checksum offload
7	FCOET	FCoE packet type indication: 1 - FCoE packet type 0 - Non-FCoE packet type
8:9	L4T / EOFT	For an FCoE packet, this field is the EOFT; for non FCoE packets, it is the L4T. L4T is the L4 packet type: 00b - Unknown / Fragmented Packet 01b - TCP 10b - SCTP 11b - UDP When the L4T is set to other values 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. EOFT is the EOF Tag to be used: 00b - EOFn01b - EOFt 10b - EOFni 11b - EOFa The EOFT field is meaningful only for an FCoE packets offload defined by an active FCOET flag. For a single packet send, the EOFT can be any of the above 4 values. In ETSO it must be set to EOFt. For TSO, it must be set to either EOFn or EOFt as follows: - EOFn when the "End Sequence" flag in the THeader is cleared - EOFt when the "End Sequence" flag in the THeader is set.

Header Offset Parameters - OFFSET (Quad Word 1, bits 16:33)

Bits	Name	Functionality
0:6	MACLEN	MAC Header Length defined in Words. In case of a tunnel packet, MACLEN defines the outer L2 header. For FCoE packets, MACLEN defines the L2 header length up to excluding the FCoE header and its Ethertype. For non FCoE packets, MACLEN defines the L2 header length up to including the Ethertype. If L2 tag(s) are provided in the data buffers, then they are included in MACLEN.
7:13	IPLEN / FCoELEN	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. For an FCoE packet, this field specifies the FCoE header length, including the FCoE Ethertype field, which must be set to 4 (16 bytes). If the IIPT and FCOET flags are cleared, this field should be set to zero.
14:17	L4LEN / FCLEN	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: For non FCoE packets: 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 equal or larger than 20 for TCP. For FCoE packets, FCLEN specifies the FC header length, defined in DWords. The FC Header length includes8 bytes/12bytes (optional) "Extended FC Headers" and the Basic (mandatory) FC header.

L2 Tag 1 - L2TAG1 (Quad Word 1, 48:63)

A 16-bit Tag to be inserted into the packet if the IL2TAG1 flag is set. If IL2TAG1 is cleared, L2TAG1 should be set by software to zero.

Transmit Buffer Description - BUFF (Quad Word 0, bits 0:63; Quad Word 1, bits 34:47)

Table 8-18. Transmit Buffer

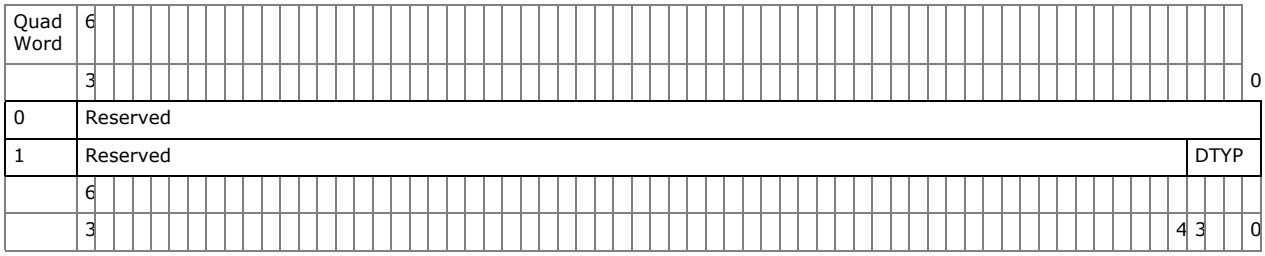
Bits	Name	Functionality
34:47	BSIZE	Buffer size in byte units from 1 byte up to 16KB minus 1
0:63	BADDR	Buffer address in byte granularity

8.4.2.1.2 RSVNOP Descriptor

A NOP descriptor can be used if there is a software need to align descriptors. A NOP descriptor does not cause any activity other then processing of the descriptor.

NOP descriptors are is implemented by "Null" setting of a context descriptor as follows: DTYP should be set to 0x1 (LAN Context Descriptor) and all other fields should be cleared. Note that NOP descriptors are permitted only between commands.

8.4.2.1.3 Transmit Descriptors Write Back Format



Descriptor Type - DTYP (Quad Word 1, bits 0:3)

The hardware indicates a completed descriptor by setting the DTYP field to a value of 0xF. The write back status is the same for all transmit descriptors used for LAN traffic, filter programming and FCoE context programming.

The hardware reports this status in the following two cases:

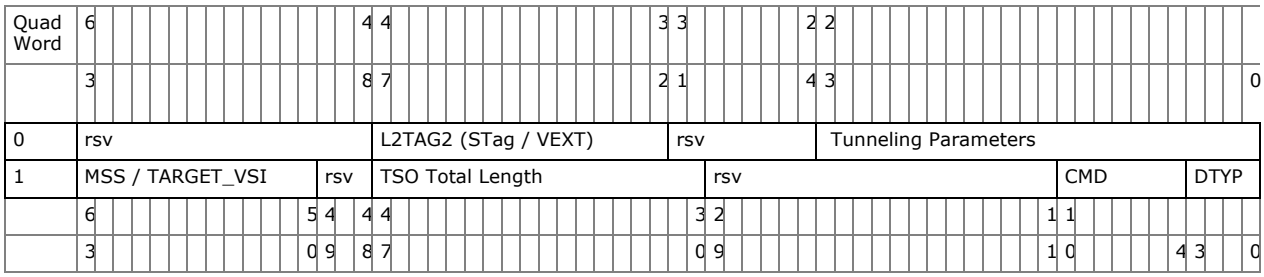
- For descriptors with the 'RS' bit set (Report Status) while the HEAD_WBEN flag in the transmit context is set to "Descriptor Write Back" mode. See rules for setting the 'RS' flag in the transmit descriptors in Section 8.4.2.1.1.
- For the last descriptor of a command that was executed before an interrupt of the queue is initiated.

8.4.2.2 LAN Transmit Context Descriptors

8.4.2.2.1 Transmit Context Descriptor

A context descriptor may have additional setting options for a single packet or TSO defined by the Data descriptor(s) that follows. Following the above packet or TSO, the context provided by this descriptor is "lost".

See Table 8.4.4 for field settings as a function of required offload.



Descriptor Type - DTYP (Quad Word 1, bits 0:3)

0x1 stands for a LAN Context Descriptor

Command Field - CMD (Quad Word 1, bits 4:10)

Bits	Name	Functionality
0	TSO	Transmit Segmentation Offload is activated when the TSO flag is set.
1	TSYN	1588 Time Stamp. When set, the hardware samples the packet transmission time in the PRTTSYN_TXTIME register. The TSYN flag is processed by the device only if TimeSync is enabled by the TSYNENA flag in the transmit queue context. Note that the TSYN flag is aimed only for a single send with no TSO.
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_L2TAGxxx 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 are mutually exclusive and the software may set only one of them. The IL2TAG_IL2H can be set to '1' only for L4 tunneled packets with non-zero L4TUNLEN.
4:5	SWTCH	Switch Control Tag (Can be set to non-zero only by control VSI as programmed by the "Allow 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 VSI's 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 Parameter (Quad Word 1, bits 30:47; bits 50:63)

Bits	Name	Functionality
30:47	TLEN	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 by software to zero.
50:63	MSS / TARGET_VSI	When the TSO flag is set, this field function as MSS. When the SWTCH field is set to 11b, this field function as TARGET_VSI. If the TSO flag is set then 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. In this case, the MSS should not be set to a lower value than 256 or larger than 9668B. 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)

Bits	Name	Functionality
0:1	EIPT	The External (outer) IP header type and its offload: 00 - no External IP header 01 - External IPv6 10 - External IPv4 with no checksum offload 11 - External IPv4 with checksum offload
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 (Teredo / 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 Else - reserved
11	RSV	Reserved zero
12:18	L4TUNLEN	L4 Tunneling Length (Teredo / GRE header / VXLAN header) defined in Words (field must be set to zero if L4TUNT equals to zero). • For standard Teredo headers with no additional header payload it should be set to 4 which equals to 8 bytes. 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 that this field represents the length of data provided in host memory buffers. If the L4TUNT is cleared, this field must be set to zero.
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.
23	Reserved	

L2 Tag 2 (Quad Word 0, bits 32:47)

Bits	Name	Functionality
32:47	L2TAG2	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 4 variable bytes or if the IL2TAG_IL2H flag is set. If the the above conditions are not met, the L2TAG2 should be set by software to zero.

RSV Reserved bits that must be set to zero.

8.4.2.3 Filter Programming Descriptor

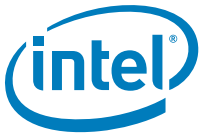
The Filter Programming Descriptor should be followed by data descriptor(s) as illustrated in Figure 7-3.

Quad Word	6 3	3 2	2 1	1 1	1 1	1 1	1 0	0
0	Reserved	DEST_VSI	PCTYPE		FLEX OFF	QINDEX		
1	Flow Director Filter ID	rsv	CNT_INDEX	CMD				DTYP

Quad Word 0

Bits	Name	Functionality
0:10	QINDEX	Destination Queue Index for this filter. The queue index is defined in the range of the VSI.
11:13	FLEXOFF	Flexible offset. This is a word offset in the “Flexible Payload” in the “Field Vector” relative to the beginning of the “Flexible Payload” space from which bytes are extracted to the 32 byte receive descriptor. The FD_STATUS flag enables either 4 or 8 bytes or none. The (first) 4 bytes indicated by FLEXOFF are posted to the “Flexible Bytes Low” field in the receive descriptor and the (last) 4 bytes are posted (if enabled) to the “Flexible Bytes High” field. Note that the FLEX bytes are supported only by 32 byte receive descriptors. The FLEXOFF must not be larger than 6 when 4 byte are enabled and must not be larger than 4 when 8 byte are enabled.
14:16	RSV	Reserved
17:22	PCTYPE	Matched packet type as defined in Table 7-5. When the FD_AUTO_PCTYPE flag in the GLQF_CTL register is cleared, this field defines the PCTYPE of the programmed FD filter (see Table 7.7 for PCTYPE values). When the FD_AUTO_PCTYPE flag in the GLQF_CTL register is set, this field is meaningless and should be set to zero. In this case the device concludes the PCTYPE by the packet used to program the FD filter.
23:31	DEST_VSI	Expected VSI Index. 0x0 - 0x17F Matched VSI Index of the received packet. Else Reserved Note that the hardware does not check the DEST_VSI at programming time while it is the software responsibility to program a DEST_VSI which is owned by the PF.
22:63	RSV	Reserved

Quad Word 1



Bits	Name	Functionality
0:3	DTYPE	0x8 for a FD filter programming Descriptor
4:19	CMD	Described in the table below.
20:28	CNT_INDEX	Statistic counter index for packets that match this filter. This field is meaningful only when the CNT_ENA flag is set in this descriptor.
29:31	RSV	Reserved
32:63	FDID	Flow Director Filter ID. Note that the FDID is opaque to the hardware. The programmed FDID can be reported back in the received descriptor of a matched received packet (if enabled by the FD_STATUS parameter in this descriptor). If so, it can be used by the software to associate the received packet with its matched filter.

Command Field - CMD (Quad Word 1, bits 4:19)

Bits	Name	Functionality
0:2	PCMD	Filter Programming Command: 001b - Add Filter (*) 010b - Remove Filter Else - Reserved (*) If the filter entry does not exist, the filter is added. If the filter already exists, the filter parameters are updated.
3:4	DEST	Matched packet destination control. 00b - Drop packet 01b - Direct packet to LAN queue defined by the QINDEX 10b - Direct packet to LAN while the queue is defined by other filters 11b - Reserved
5:6	RSV	Reserved (must be set to 00b)
7	CNT_ENA	Statistic Counter Enable flag (relevant only for FD filter). This flag enables packet counting by the Statistic counter defined by the Statistic Counter Index in this command.
8	RSV	Reserved
9:10	FD_STATUS	The FD_STATUS flag enables status indication of the FD filter ID and extracting flexible bytes in the receive descriptor as follows: 00b - FD ID and Flexible bytes are not enabled 01b - FD ID is enabled 10b - FD ID and 4 Flexible bytes are enabled 11b - 8 Flexible bytes are enabled
11:15	RSV	Reserved

8.4.2.4 FCoE Descriptors

8.4.2.4.1 FCoE Data Descriptor

FCoE data descriptor is the same transmit data descriptor used for regular LAN traffic as described in Section 8.4.2.1.1. Multiple data descriptors can be used the same as regular LAN packets.

8.4.2.4.2 FCoE Single Send Descriptors

In most cases the Data descriptor is sufficient to indicate a single FCoE packet. An "FCoE Transmit Context Descriptor" is needed as well if it is required to insert double L2 tags from the descriptors. In this case, the context descriptor is defined first, followed by the data descriptor(s).

FCoE TSO and ETSO Descriptors

FCoE TSO and ETSO descriptors include the following context descriptors followed by regular Transmit data descriptors for the actual sent packets. Note that the FCoE header must be contained in a single data descriptor (as opposed to regular LAN header that can span on up to 3 descriptors). The ~~context~~TSO descriptors must be programmed in the following ordering.

- FCoE Transmit Context Descriptor (described in Section 8.4.2.4.5)
- Transmit Data Descriptor(s) (described in Section 8.4.2.2.1)

8.4.2.4.3 FCoE DDP Descriptors

FCoE DDP context is programmed by the following descriptors listed in the same order they should be programmed. A DDP context for a read exchange is programmed with a Read request packet. The "exchange context" flag in the FC header is expected to be cleared. A DDP context for a write exchange is programmed with a RDY packet. The "exchange context" flag in the FC header is expected to be set. The DDP programming descriptors must be initiated from the same transmit queue associated with the FCoE traffic class.The DDP buffers are organized in a dedicated queue. That queue is indicated by the DDP queue context descriptor indicated below. The hardware process requests to add a context only if the FDENA flag in the transmit queue context is set. Otherwise (programming is not enabled), the packet indicated by the context is transmitted as requested without programming any context.

- DDP Context Descriptor (described below in Section 8.4.2.4.8)
- DDP Queue Context Descriptor (described below in Section 8.4.2.4.6)
- DDP Filter Context Descriptor (described below in Section 8.4.2.4.7)
- Transmit Data Descriptor(s) (described in Section 8.4.2.1.1)

8.4.2.4.4 FCoE Context Invalidation

A context is invalidated by the FCoE Transmit context descriptor (described below in section Section 8.4.2.4.5) followed by a data descriptor. The "OPCODE" field is set to "DDP Context invalidation".

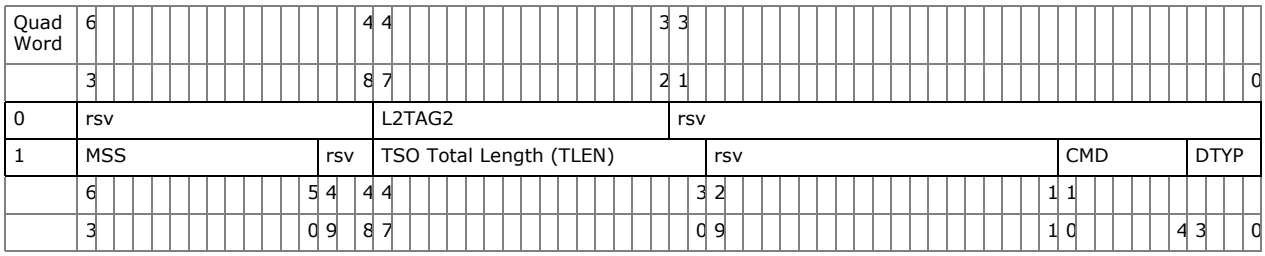
The data transmit descriptor indicates an FCoE packet structure with the same header parameters that were used for the context programming. The FCoE packet can be optionally transmitted according to the "DUMMY" flag in the data descriptor. The DDP context invalidation descriptors must be initiated from the same transmit queue associated with the FCoE traffic class.

The context invalidation status indication is provided in a "Programming Status" descriptor after it is guaranteed that no additional DMA accesses are made to the FCoE buffers.

The "Programming Status" descriptor is indicated in the LAN receive queue that is defined in the invalidated DDP context. Software must wait for the invalidation status indication before it releases the DDP memory structures.

8.4.2.4.5 FCoE Transmit Context Descriptor

The FCoE Transmit context descriptor is used for TSO, ETSO and DDP.



Descriptor Type - DTYP (Quad Word 1, bits 0:3)

0x2 stands for a FCoE Context Descriptor

Command Field - CMD (Quad Word 1, bits 4:10)

Bits	Name	Functionality
0:3	OPCODE	Opcode of this context descriptor as follow: 0000b - Single send context 0X01b - TSO offload for FC Class 2 or FC Class 3 when 'X' equals to '0' or '1' respectively 0X10b - ETSO offload for FC Class 2 or FC Class 3 when 'X' equals to '0' or '1' respectively 1000b - DDP Context Invalidation (the packet used for this operation must be a single send) Else - reserved
4	RELOFF	Relative Offset flag. When set, the PARAM field in the FC header is an offset field dynamically updated in TSO, ETSO . When cleared, the PARAM field remains constant as provided in the THeader (THeader is the packet header template described in the FCoE Engine chapter). This flag is meaningless when SEG equals Context invalidation.
5	CLRSEQ	Clear Sequence Count flag. When set, the sequence count field is cleared on the RSP packet of ETSO.
6	RSV	Reserved.
7	IL2TAG2	Insert the L2 tag from the L2TAG2 field in this descriptor. 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_L2TAGxxx registers.

Segmentation Parameters - SEGPAR (Quad Word 1, bits 30:47; bits 50:63)

Bits	Name	Functionality
30:47	TLEN	TSO Total Length. The total length defines the FC payload bytes. If the OPCODE is not TSO nor ETSO, this field must be set to zero.
50:63	MSS	The Max Segment Size for non single send option. MSS defines the maximum FC payload size of segmented packets in byte units. If the OPCODE is not TSO nor ETSO, this field must be set to zero.

8.4.2.4.6 FCoE DDP Queue Context Descriptor

Quad Word	63 … 12					11 … 0
0	FCoE DDP Descriptor List Base Address					DDP Context Index
1	rsv	rsv	rsv	rsv	TPH	Descriptor List Length
	63 … 48	47 … 32	31 … 28	27 … 16	15 … 13	12 … 0

Quad Word 0

Bits	Name	Functionality
0:11	DDPINDX	DDP Context Index. The context index in this descriptor defines the programmed context index. This field must be identical to the DDPINDX field in the DDP Filter Context Descriptor.
12:63	FBASE	FCoE DDP Descriptor List Base Address defined in 4KB units

Quad Word 1

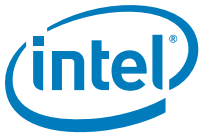
Bits	Name	Functionality
0:12	FLEN	FCoE DDP Descriptor List Length defined in descriptors units (note that each DDP descriptor is 8 bytes long)
13	TPHRDESC	TPH Enablement for descriptor fetch. The CPU ID is taken from the LAN Tx queue context used for the programming of the FCoE context.
14	TPHDATA	TPH Enablement for FC Data payload access in the data buffers (Write to the DDP buffers). Same CPU ID as above.
15	RSV	Reserved.

8.4.2.4.7 FCoE DDP Filter Context Descriptor

Quad Word	63 … 52	51 … 48	47 … 32	31 … 0
0	DDP Context Index	rsv	Expected Sequence Number	Expected PARAM

Quad Word			
1	LANQ Index	rsv	flags
	63 … 53	52 … 14, 13 … 8	7 … 0

Quad Word 0



Bits	Name	Functionality
0:31	PARAM	Expected PARAM field in the received packet for the DDP
32:47	SEQN	Expected Sequence Number of the first packet in the DDP
48:51	RSV	Reserved
52:63	DDPINDX	DDP Context Index. This field must be set to the same value of the DDPINDX field in the DDP Queue Context Descriptor.

Quad Word 1

Bits	Name	Functionality
0	CTYP	Context Type: 0b - DDP 1b - Reserved
1	ENODE	End Node Type: 0b - Initiator 1b - Responder
2	FC_CLASS	FC Class: 0b - Class 2 1b - Class 3
3:7	RSV	Reserved
14:52	RSV	Reserved
53:63	LANQ	LAN Receive Queue Index within the VSI space. Upon packet reception, the hardware associates the LAN queue number equals to LANQ.

8.4.2.4.8 FCoE DDP Context Descriptor

Quad Word	6 3	3 2	1 1	0
	3	2 1	2 1	
0	rsv			
1	rsv	Last Buffer Size	rsv	First Buffer Offset

Quad Word	6 4	4 3	3 2	1 1	0
	3 6 5	2 1	0 9	2 5	8 7
1	rsv	rsv	First Buffer Offset	rsv	CMD	DTYP

Quad Word 1

Bits	Name	Functionality
0:3	DTYP	0x9 stands for a DDP Context Descriptor
4:5	BSIZE	DDP Buffer Size: 00b - 512B 01b - 4KB 10b - 8KB 11b - 16KB
6	RSV	Reserved.

Bits	Name	Functionality
7	LASTSEQH	When this flag is set, the header of the last packet in a sequence is posted to the LAN receive queue. The receive queue index is defined in the DMAINDX parameter in the FCoE context descriptors.
16:29	FOFF	Byte Offset in the first buffer on which DDP starts. The FOFF must be smaller than BSIZE. Failing to do so the whole DDP is not accepted by the HW.
32:45	LSIZE	Last DDP Buffer size in byte units. The LSIZE must be smaller or equal than BSIZE. In case of a DDP that has only a single buffer, the LSIZE must be larger than FOFF. Setting the LSIZE to 0x0 means it equals to 16KB.

8.4.3 LAN Transmit Queue (Ring)

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 "descriptor ring". The XL710 supports up to 1536 transmit queues allocated to PFs and VFs as described in Section 8.2.

Transmit queues are defined by a set of parameters called the "queue context". The main parameters are the queue pointers presented in the Figure 8-8. The queue context includes additional parameters that define the queue functionality as detailed in Section 8.4.3.4. Part of these context parameters are kept in hardware (like the Tail register, queue enable flag and interrupt related context). Most of the queue context parameters are stored in Function Private Memory (FPM), fetched to an internal cache when required.

The software interface to the queue (for initialization, nominal operation, and queue disable flow) is described in Section 8.4.3.1. The XL710 has additional global parameters for the whole device or per function parameters that affect multiple queues. See Section 8.3.3.1.

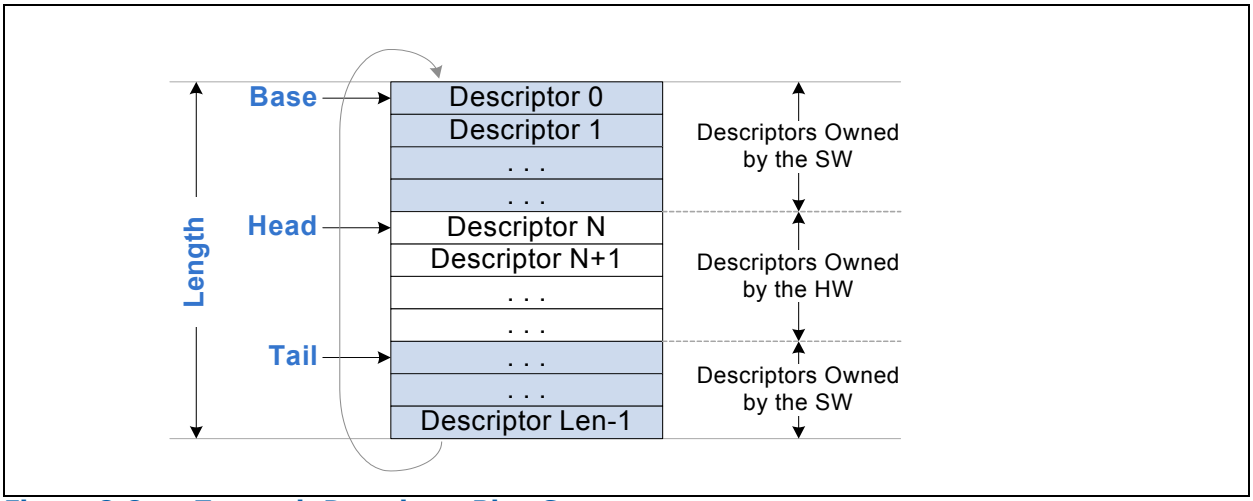


Figure 8-8. Transmit Descriptor Ring Structure

8.4.3.1 Transmit Queue Programming

Queue enable and disable flows are described in the following section and summarized in the Table 8-19.

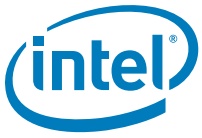


Table 8-19. Transmit Queue Enable / Disable Flags

QTX_ENA[n] register		Transmit 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.

8.4.3.1.1 Transmit Queue Enable Flow

Queue enable flow is executed by the PF software for PF queues as well as for the queues of its VFs.

- The function that owns the queue (PF or VF) allocates contiguous memory in its own memory space for the transmit ring.
- If the queue might have been disabled recently then the software should wait at least 50msec from the completion of the queue disable flow before proceeding to the next step.
- The software should clear the queue disable flag by setting the CLEAR_QDIS flag in the matched GLLAN_TXPRE_QDIS[n] register. The QINDX field in the GLLAN_TXPRE_QDIS[n] register should be set to the absolute queue index (equals to the queue index within the PF plus the FIRSTQ field in the PFLAN_QALLOC register of the PF)
- The software prepares the queue context in the FPM in PF memory space, clears the QTX_HEAD[n] register and then sets the QENA_REQ flag in the QTX_ENA[n] register, while 'n' is the queue index within PF space. Before that, the PF also defines the function that owns the queue in the PFVF_FUN; PF_INDX; VFVM_INDX parameters in the matched QTX_CTL[n] register. See several examples for queue context settings in Section 8.4.3.4.3. At pre-boot time, memory allocation might be tight and the FPM could exceed the budget.

Note: There is a mechanism to program the queue context directly to the device bypassing the FPM as described in Section 8.3.3.1.1.1.

- As a response, the hardware sets the QENA_STAT flag in the QTX_ENA[n] register. The QENA_STAT follows the QENA_REQ almost instantly and not more than 10usec after that.
- Once the QENA_STAT flag in the QTX_ENA[n] register is set, software can start using the queue.
- If the queue is targeted for a VF, PF software should program also the matched entry in the VFQ_TABLE. Then it is expected to inform the VF software that the queue is enabled. Note that the VFQ_TABLE is shared for the receive and transmit queues. So the PF software should enable also the receive queue before communicating it to the VF.

8.4.3.1.2 Transmit Queue Disable Flow

Queue disable flow is executed by PF software for PF queues as well as for the queues of its VFs.

- Remove the queue from the interrupt linked list as described in Section 7.5.3.1.3.
- The PF software sets the SET_QDIS flag in the matched GLLAN_TXPRE_QDIS[n] register. The DINDX field in the GLLAN_TXPRE_QDIS[n] register should be set to the absolute queue index (equals to the queue index within the PF plus the FIRSTQ field in the PFLAN_QALLOC register of the PF). Software should guarantee a gap of at least 100usec at 2x10G and above or 400usec at 1x10G before the next step is executed. Note that software could execute this step for multiple transmit queues and then proceed to the next step for these queues concurrently.

- PF software clears the QENA_REQ flag in the QTX_ENA[n] register, while 'n' is the queue index within the PF space.
- The hardware schedules this event like it does for any other software update of the Tail. Further updates of the Tail register are ignored by the hardware.
- The hardware post a "queue disable" marker in the transmit "pipe".
- Any preceding commands in the transmit "pipe" are processed while their buffers are fetched from host memory.
- Eventually, the "queue disable" marker gets to the end of the "pipe". At this point it is guaranteed that the "pipe" does not contain any additional transmit commands that might require the host memory.
- The hardware invalidates the transmit queue from the internal cache and also clears the QENA_STAT flag in the QTX_ENA[n] register.
- Once the QENA_STAT flag in the QTX_ENA[n] register is cleared, the software can release all memory structures of the queue. If the QENA_STAT flag in the QTX_ENA[n] register is not cleared within a "reasonable time" it might be an indication for too long "pause" packets from the network. In such a case the software could follow the flow described in Section 7.7.1.2.8.

Note: The progress of the "queue disable" marker following clearing the QENA_REQ flag is subjected to scheduling in the transmit data path the same as any other packet. This means that processing latency might take longer time if the traffic class is paused.

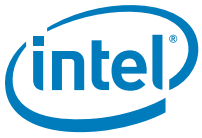
8.4.3.1.3 Fast Transmit Queue Disable Flow

Fast queue disable flow is executed by the hardware following a PFR, VFR or VMR resets. Once these reset are completed by the hardware, all transmit queues of the function or the VM are disabled. All queue contexts of the matched function or the VM are invalidated from the internal cache and their QENA_STAT flag in the matched QTX_ENA[n] register are cleared while the queue context is not updated in the FPM.

8.4.3.1.4 Software Fast Path Programming

During nominal operation, the function that owns the queue (PF or VF) accesses the hardware directly.

- Prepare transmit descriptors starting at the descriptor indicated by the TQTAIL pointer in the relevant QTX_TAIL register. Note that the software should update the TQTAIL at whole structures boundaries. For TSO, it is the whole TSO; for a single packet, it is the whole packet; and for filter or FCoE context programming, it is the end of the packet that is associated to the filter programming. Failing to do so might hang the activity of the queue.
- 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.
- The hardware reports completed descriptors only for those ones indicated by an 'RS' bit in the CMD field in the descriptor. 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 DTYP field in the completed descriptor to a 0xF (DTYP equals to 0xF is reserved for completed descriptor indication). It also sets the rest of the descriptors to all zero's.
  - Head Write Back: The hardware indicates the head pointer of the next descriptor that follows a completed descriptors (with active 'RS' bit). The hardware post the 4 bytes of the head pointer at the address defined by the HEAD_WBADDR parameter in the queue context.



- Bump the TAIL to the last prepared descriptor plus one (the descriptor index to be added in the next time).

8.4.3.2 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 that 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.

8.4.3.3 Transmit Arbitration Queue Set

Transmit queues are grouped into arbitration "queue sets". The XL710 supports 1024 LAN "queue sets" (indicated by the RDYList parameter in the transmit queue context). Queue arbitration within the same "queue sets" is a simple round robin scheme. Arbitration between "queue sets" is described in Section 7.8.1.5. A queue context is fetched on demand (if not already in the cache) when it is scheduled. Then the hardware fetches the transmit descriptors (if not already in the cache) and then it fetches the transmit data.

8.4.3.4 Transmit Queue Context Parameters

This section describes the setting options of the LAN transmit queue parameters named as "queue context". Part of the queue context parameters reside in dedicated hardware registers and part of the context is stored in host memory (FPM) while the active queues are kept in cache.

8.4.3.4.1 Transmit Queue Context in Hardware Registers

The queue context parameters that software accesses during fast past programming as well as its enablement reside in hardware registers:

- Queue enablement flags in the QTX_ENA[n] registers while 'n' is the queue index of the PF. Queue enable and disable flow by the PF are explained in subsections of Section 8.4.3.1.
- The TAIL pointer in the QTX_TAIL[n] registers while 'n' is the relative queue index of the function (PF or VF). The usage of the Tail register is detailed in Section 8.4.3.1.4
- Association of the queue to a function (PF and its VF or VM) is programmed by the QTX_CTL[n] registers while 'n' is the relative queue index of the PF. The QTX_CTL includes the following parameters that are programmed by the PF:
  - PF_INDX is the absolute index of the PF (between 0 and 15)
  - PFVF_Q flag associate the queue to the PF, the EMP or to one of its VFs or its VMs.
  - VFVM_INDX is the absolute index of the VF (between 0 and 127) if the queue belongs to a VF or the absolute index of the VSI (between 0 and 383) if the queue belongs to a VM. Otherwise, the VFVM_INDX should be set to zero.

8.4.3.4.2 Transmit Queue Context in FPM

The transmit queue context parameters that are stored in host memory (FPM) are fetched for the active queues to the internal cache. These parameters are described in the Table 8-20 below. The Queue context is structured as contiguous lines of 128 bits (16 bytes) each, as follows:

Table 8-20. LAN Tx Queue Context in the Private Host Memory

Alias	Width [bits]	LS bit	MS bit	Type	SW Init	Description
Line 0						
HEAD	13	0	12	Dynamic	0x0	Transmit Queue Head. An index updated by the hardware relative to the beginning of the queue that defines the next descriptor to be used. All descriptors starting by the HEAD up to (excluding) the TTAIL are owned by the hardware and the rest are owned by software (as shown in Figure 8.1). At idle time the HEAD equals to the TTAIL.
RSV	16	13	29	Dynamic	0x0	Reserved
New_Context	1	30	30	Dynamic	0x1	New Context Flag. Should be set to '1' by software at queue context programming.
RSV	1	31	31	N/A	0x0	Reserved
BASE	57	32	88	Static	BASE	Transmit Queue Base Address. Indicates the starting address of the descriptor queue defined in 128 Byte units.
FCENA	1	89	89	Static	FCENA	FCoE Enablement (FCENA). Enable transmission of FCoE and FIP packets from this queue. When this flag is cleared FCoE packets and context programming on this queue are dropped silently.
TSYNENA	1	90	90	Static	TSYNENA	TimeSync Enable. If the TSYNENA flag is set, then setting the TSYN flag in the transmit context descriptor is processed by the hardware sampling the packet timestamp in the PRTTSYN_TXTIME register.
FDENA	1	91	91	Static	FDENA	Enable FD filter programming by this queue.
ALT_VLAN	1	92	92	Static	ALT_VLAN	Alternate VLAN tag selects. At '0' the PORT_TAG_ID tag in the VSI_TIR is disabled and at '1' the PORT_TAG_ID tag in the VSI_TAIR register is enabled.
RSV	3	93	95	N/A	0x0	Reserved
CPUID	8	96	103	Dynamic	CPUID	CPU Socket ID for TPH. The CPU socket ID is updated by the hardware
RSV	4	104	127	N/A	0x0	Reserved
Line 1						
THEAD_WB	13	0	12	Dynamic	0x0	Reflection of the reported HEAD at Head WB.
RSV	19	13	31	N/A	0x0	Reserved
HEAD_WBEN	1	32	32	Static	HEAD_WBEN	This flag selects between Head WB and transmit descriptor WB: 0b - Descriptor Write Back 1b - Head Write Back
QLEN	13	33	45	Static	QLEN	Transmit Queue Length (QLEN). Defines the size of the descriptor queue in descriptors units from 8 descriptors (QLEN=0x8) up to 8K-32 descriptors (QLEN=0x1FE0). Q LEN restrictions: At smaller queue size than 32 descriptors the QLEN must be whole number of 8 descriptors. At larger size than 32 descriptors the QLEN must be whole number of 32 descriptors.
TPHRDesc	1	46	46	Static	TPHRDesc	Descriptor Read TPH Ena for descriptors fetch.
TPHRPacket	1	47	47	Static	TPHRPacket	Packet Content TPH Ena flag for the packet header and payload.
TPHWDesc	1	48	48	Static	TPHWDesc	Head WB / Descriptor Completion Status Write Back TPH Ena flag.
RSV	15	49	63	N/A	0x0	Reserved

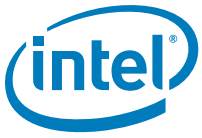


Table 8-20. LAN Tx Queue Context in the Private Host Memory

Alias	Width [bits]	LS bit	MS bit	Type	SW Init	Description
HEAD_WBADDR	64	64	127	Static	HEAD_WBADDR	When the HEAD_WBEN is set to '1', this field defines the ~~T~~HEAD WB Address. Must be set to 0x0 if HEAD_WBEN is cleared.
Line 2						
RSV	128	0	127	N/A	0x0	Reserved
Line 3						
RSV	128	0	127	Tx_Desc	0x0	Reserved for internal parameters
Line 4						
RSV	128	0	127	Internal	0x0	Reserved for internal parameters
Line 5						
RSV	128	0	127	Internal	0x0	Reserved for internal parameters
Line 6						
RSV	128	0	127	Internal	0x0	Reserved for internal parameters
Line 7						
RSV	84	0	83	Internal	0x0	Reserved
RDYList	10	84	93	Static	RDYList	Transmit arbitration "queue set". The RDYList index is absolute so it should be set to those RDYList allocated to the function.
RSV	34	94	127	Internal	0x0	Reserved

8.4.3.4.3 Examples for Transmit Queue Context Setting

Listed below are two examples of transmit queue context settings:

PF LAN transmit Queue Context Example =

Offset: Even_L.0 Even_L.1 Even_L.2 Even_L.3 Odd_L.0 Odd_L.1 Odd_L.2 Odd_L.3

Line 0, 1: 0x40000000 001579A0 08000000 00000000 00000000 0001C200 00000000 00000000

Line 2, 3: 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Line 4, 5: 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Line 6, 7: 0x00000000 00000000 00000000 00000000 FFFFFFFF 480FFFFF 00000000 00000000

BASE = 0x001579A0 (0x0ABCD000)	HEAD_WBEN = 0
FCENA = 0	QLEN = 512 (0x200)
TSYNENA = 0	TPH = 111b (enable all TPHxxx flags)
FDENA = 1	HEAD_WBADDR = 0x00...0
ALT_VLAN = 0	RDYList = 0x80 (128)

FCoE Transmit Queue Context Example =

Offset: Even_L.0 Even_L.1 Even_L.2 Even_L.3 Odd_L.0 Odd_L.1 Odd_L.2 Odd_L.3

Line 0, 1: 0x40000000 001779A0 12000000 00000000 00000000 0001C201 1BBCD000 00000000

Line 2, 3: 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Line 4, 5: 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Line 6, 7: 0x00000000 00000000 00000000 00000000 FFFFFFFF 481FFFFF 00000000 00000000

BASE = 0x001779A0 (0x0BBCD000)	HEAD_WBEN = 1
FCENA = 1	QLEN = 512 (0x200)
TSYNENA = 0	TPH ENA = 111b (enable all TPHxxx flags)
FDENA = 0	HEAD_WBADDR = 0x1BBCD000
ALT_VLAN = 1	RDYList = 0x81 (129)

8.4.4 Stateless Transmit Offloads

8.4.4.1 Insert Ethernet CRC bytes

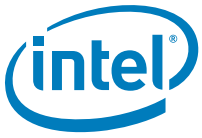
See Section 7.2.1.1.

8.4.4.2 Transmit L3 and L4 Integrity Offload

The XL710 can offload the following L3 and L4 integrity checks: IPv4 header(s) checksum for “simple” and tunneled packets, Inner TCP or UDP checksum and SCTP CRC integrity. Tunneling UDP headers and GRE header are not offloaded while the XL710 leaves their checksum field as is. If a checksum is required, software should provide it as well as the inner checksum value(s) that are required for the outer checksum. In order 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 8-9 and Table 8-21.

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 XL710.
- 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 the OS should not request checksum nor segmentation offload for packets with routing extension header type 2.
- IP Header Length Requirements (IPLEN and EIPLEN)
  - In case of 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 defined 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 L2 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).

The Table 8-21 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.

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
- 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.
- 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 8 bytes (bare Teredo 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.

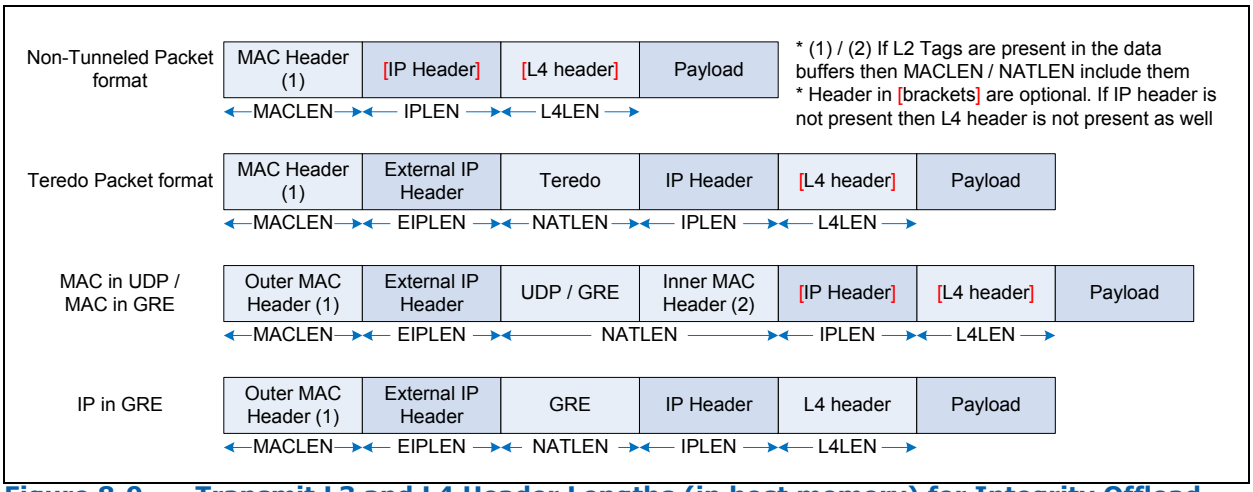


Figure 8-9. Transmit L3 and L4 Header Lengths (in host memory) for Integrity Offload

Table 8-21. 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 IPv4 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.
IP -> IP -> L4	IIPT = 10b or 11b for inner IPv4 IIPT = 01 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 = 01 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
IP -> (Fragmented IP or IP -> Unknown)	IIPT, IPLEN, EIPT, EIPLEN, L4TUNT = same as IP -> IP -> L4 L4T = 00b (unknown)	Inner IPv4 checksum if IIPT = 11b; Outer IPv4 checksum if EIPT = 11b; No L4 checksum

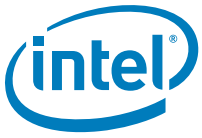


Table 8-21. Transmit Integrity Offload for Packet Types

Packet Type	Parsing hints and offload enablement in the transmit descriptors (1) (2)	Supported Transmit Checksum Offload
IP -> Teredo UDP / GRE -> IP -> L4	IIPT = 10b or 11b for inner IPv4 IIPT = 01 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 = 01 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 -> Teredo UDP / GRE -> (Fragmented IP or IP -> Unknown)	IIPT, IPLEN, EIPT, EIPLEN, L4TUNT, L4TUNLEN = same as IP -> IP -> L4 L4T = 00b (unknown)	Same as IP -> (Fragmented IP or IP -> Unknown)
IP -> GRE / UDP -> MAC (w/wo VLAN) -> IP -> L4	IIPT, IPLEN, L4T, EIPT, EIPLEN = same as IP -> IP -> L4 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
IP -> GRE / UDP -> MAC (w/wo VLAN) -> (Fragmented IP or IP -> Unknown)	IIPT, IPLEN, L4T, EIPT, EIPLEN = same as IP -> (Fragmented IP or IP -> Unknown) L4TUNT, L4TUNLEN = same as IP -> GRE / UDP -> MAC (w/wo VLAN) -> IP -> L4	Same as IP -> (Fragmented IP or IP -> Unknown)
Note 1. Common settings to all cases: When context descriptor is used, the TSO flag should be cleared. Note 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 may be needed for other offloads than checksum. The values indicated in this table assumes that the context descriptor is used.		

8.4.4.3 Transmit Segmentation Offload (also named as TSO or LSO)

Transmit Segmentation Offload (TSO, also called Large Send offload - LSO) enables the TCP/IP stack to pass to the network device a larger ULP datagram than the Maximum Transmit Unit Size (MTU). The XL710 divides the large ULP datagram to multiple segments according to the MTU size as illustrated in the Figure 8-10. 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 256KB (2^18) supporting TSO and “Giant” TSO.

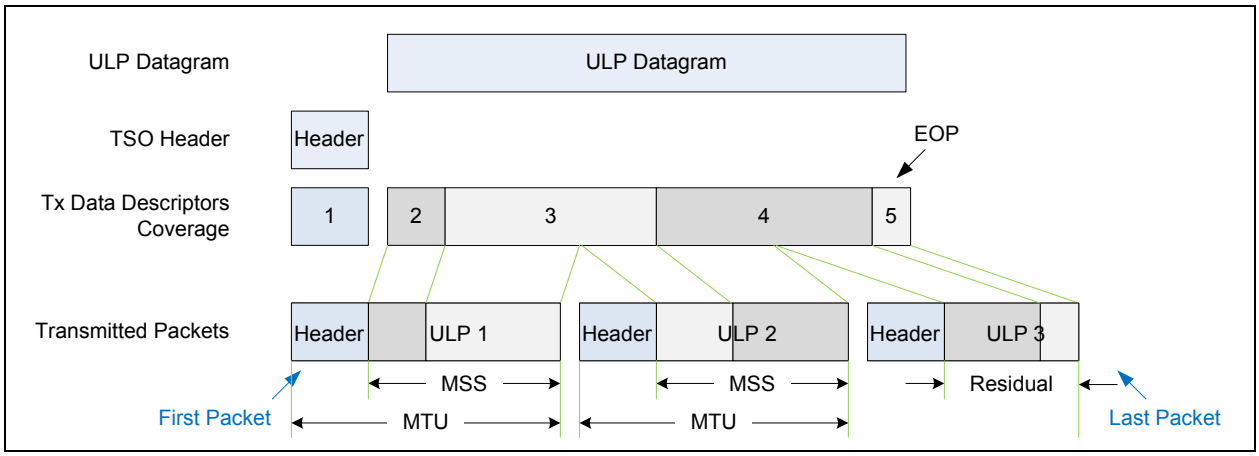


Figure 8-10. TSO Functionality (Example)

8.4.4.3.1 Frame Formats and Assumptions

The following packet formats are supported for TSO:

- IP -> L4// IP is IPv4 or IPv6 with/without Option/Extension headers. L4 is TCP.
- IP -> IP -> L4
- IP -> Teredo -> IP -> L4
- IP -> UDP Tunneling [Key] -> MAC -> [VLAN] -> IP -> L4
- IP -> GRE [Key] -> MAC -> [VLAN] -> IP -> L4
- IP -> GRE [Key] -> IP -> L4
- SNAP packet formats are not supported for TSO

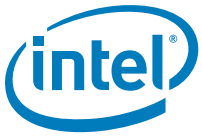
The following assumptions apply to the TCP segmentation implementation in the XL710:

- 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 8-21.
- 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).

8.4.4.3.2 Transmit Segmentation Flow

The TSO flow includes the following steps:

- The protocol stack receives from an application a ULP datagram to be transmitted.



- The protocol stack calculates the number of packets required to be transmitted based on the MTU.
- 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.
- 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.
- The hardware fetches the segmentation parameters as well as the data and header buffers description by the transmit context and data descriptors.
- The hardware fetches the header and data buffers and then transmits them by segments according to the TSO parameters.
- 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 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).
- 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 the 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 to: 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.
  — 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 then it is incremented by one for each transmitted segment.
  — The Identification field (in the external IP header) is taken from the TSO header in the first segment and then it is either incremented by one for each transmitted segment or remains constant depending on the EIP_NOINC setting in the transmit context descriptor.
- The header checksum is calculated after the other parameters in the IP header are updated. 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. So for the inner IP header (in case of tunneling) it is the 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 to: 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.
- 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, then it is processed the same as the last segment of a multiple 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 to the payload size plus the TCP header size).
- Dynamic fields in a tunneling UDP header (Teredo or MAC in UDP) 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).

8.4.4.3.3 Transmit Arbitration

The XL710 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 XL710 switches to the next queue in line at the end of the transmitted segment of the TSO. See the "Transmit Scheduling" chapter for a description of transmit arbitration.

8.4.4.3.4 Segmentation Indication to the Hardware

Software indicates a TCP segmentation by a transmit context descriptor just before the data descriptor with the following parameters:

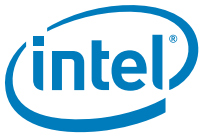
- TSO flag is set, indicating a TSO is requested.
- Insert S-Tag or External VLAN can be set only by the PF (the same as a single send).
- Switch control fields must not be enabled for TSO.
- MSS should be set to the required size of the L4 payload on each segment (MTU minus the size of the headers).
  - MSS outside the range (256B - 9674B) 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 8.4.4.3.1.

8.5 TimeSync (IEEE1588 and 802.1AS)

8.5.1 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 XL710 includes 1588 logic per LAN port; the logic is enabled by the TSYNENA flag in the PRTTSYN_CTL registers per port. The PFs can access only the 1588 registers of a port. All PFs on a given port can access the 1588 registers while only one PF is expected to control the logic. The PF ID that owns the 1588 logic of the port is reflected by the PF_ID in the PRTTSYN_CTL0 registers per port. All PFs are permitted to read the 1588 timer registers (on the port they belong).

Software Note: The PF_ID field has no impact on the hardware; it is used only by the software. The field is expected to be loaded from NVM or set by management agent. During nominal operation the PF software driver is not expected to change the field's setting.

The synchronized clock "PRTTSYN_TIME" is a 96-bit register (per LAN port). Out of it, the upper 64 bits are used for timestamp sampling and synchronized events described later in this section. Tune this register so that the upper 64 bits are defined in nsec units. The register then can define the absolute time relative to PTP "epoch" which is January 1st 1970 00:00:00 International Atomic Time (TAI).

1588 Registers:

All 1588 registers are supported per port.

Many 1588 registers are 64 bits wide. The XL710 provides a 32-bit interface. Therefore, accessing these registers require multiple slave accesses.

- Reading any 64-bit register, software should access the LS register first and then the MS register. Specifically relating to dynamic registers: reading the LS register, the hardware latches internally the value of the MS register till it is fetched as well. Included in this category are: PRTTSYN_TIME; PRTTSYN_RXTIME; PRTTSYN_TXTIME; PRTTSYN_INC; PRTTSYN_TGT and PRTTSYN_EVNT registers.
- Writing to any 64-bit registers. software should write the LS register first and then the MS register. Only after writing the MS register is the value latched internally. Included in this category are: the PRTTSYN_TIME, PRTTSYN_INC, PRTTSYN_TGT, and PRTTSYN_EVNT registers. Writing to the PRTTSYN_TIME (*_L and *_H) registers, the device also clears the additional internal lowest 32 bits (usually used for "sub nsec" units). Note that this "sub nsec" register is internal, and not exposed to the software.

8.5.2 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.

8.5.2.1 Initialization Phase

If enabled as a potential master (by a software setting), the software transmits sync packets that include the master's clock parameters periodically. 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 each slave, software switches back to the initialization phase until a new master is chosen. Note that there are more than one option for the above flow while there are other flows which are based on static master setting. The node can be set statically to a master or slave modes. While in a slave mode, the node can be tuned statically to a specific master.

8.5.2.2 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.

8.5.2.2.1 Clocks Calibration

The master sends SYNC packets periodically (about of 10 packets per second). 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 starts with an initial time calibration which can be used as the transmission delay between the master and the slave.

In order to minimize sampling inaccuracy, both master and slave sample the packets transmission and reception time at a location in the hardware that has as much as possible deterministic delay from the PHY interface.

Software hardware flow in the master

Sending the SYNC packet by the master, software indicates this packet to the hardware by setting the TSYN flag in the transmit context descriptor. Setting this flag, the hardware samples its transmission time using the PRTTSYN_TXTIME register. After transmission, software should read the PRTTSYN_TXTIME register and sent the transmission time in a Follow_Up packet (according to 2-step flow). Note that software is responsible to read the PRTTSYN_TXTIME register before initiating another packet with an active TSYN flag.

Software hardware flow in the slave

The SYNC packet is received by the slave and its reception time is sampled using one of four PRTTSYN_RXTIME registers. The index of the register that sampled the reception time is reported in the TSYNINDX field in the receive descriptor. The PRTTSYN_RXTIME register holds the reception timestamp until the software reads it. It then released for sampling another packet reception.

8.5.2.2.2 Time Synchronization Phase

The complete synchronization scheme is illustrated below in Figure 8-11. It relies on measured timestamp of Sync packets transmission and reception by the master and the slave. The scheme is based on t 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.

  The master's software sends periodic Sync packets to each slave followed by a Follow_Up packet (as explained in the "Software hardware flow in the master" for the "Clocks Calibration" phase). The slave samples the TSYN packet reception time in one of the PRTTSYN_RXTIME registers.

The slave responds, sending a Delay_Request packet to the Master. The hardware in the slave samples its transmission time in the PRTTSYN_TXTIME register and the hardware in the master samples its reception time in one of the PRTTSYN_RXTIME registers. The software in the slave checks the Delay_Request packet transmission time (reading the PRTTSYN_TXTIME register) and could optionally send a Follow_Up packet that includes the captured transmission time. The master responds back with the Delay_Response packet reporting the reception time of the Delay_Response to the slave. At this point, the slave has all the following parameters (the notations below match the notations used in Figure 8-11):

- **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 = - [(T2-T1) - (T4-T3)] / 2

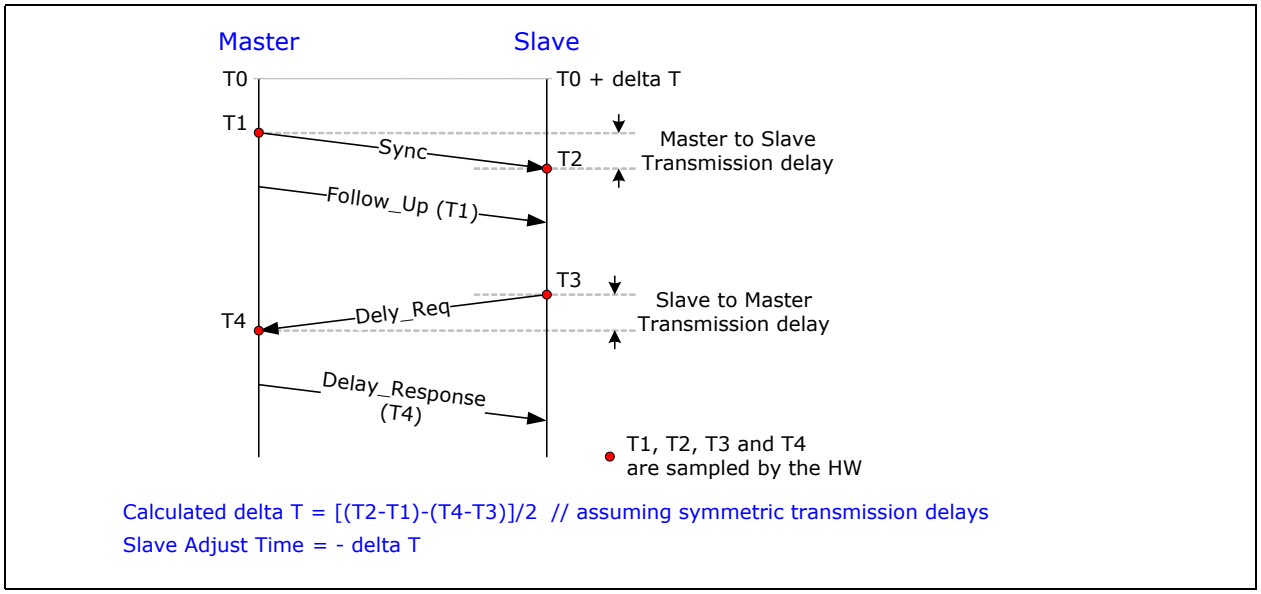


Figure 8-11. Sync Flow and Offset Calculation

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 when the master is changed. The process is usually enabled when dynamic master selection is enabled. It can operate asynchronous to the "Time Synchronization" flow described above. The Figure 8-12 below shows the Pdelay flow.

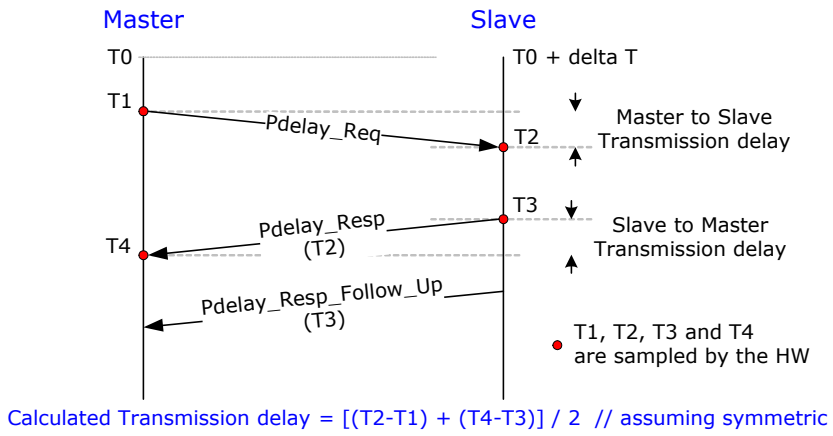


Figure 8-12. PDelay Flow

8.5.3 Timestamp Indication

Relevant packet transmission time (as Sync and Delay_Request) are sampled by the hardware at a deterministic as possible affinity to the PHY interface. The sampled time is taken at the beginning of the packet (packet size doesn't matter) as shown below. The latency between packet time sampling and packet on the "Ethernet Interface pins" is shown in Table 8-22.

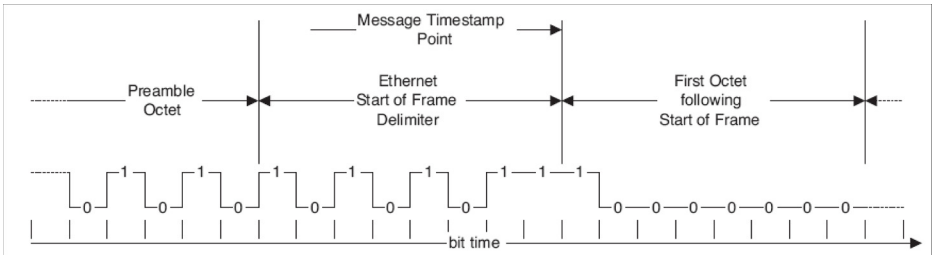
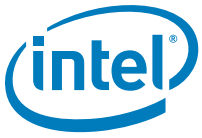


Figure 8-13. Time Stamp Point

Table 8-22. Packet Time-Stamp Latency

Link Speed	Interface	Time Stamp to Packet Transmission Latency	Packet Reception to Time Stamp Latency
40G	XLAUI / XLPPI		
40G	KR4 / CR4		
10G	XAUI / KX4		
10G	KR / SFI		
1G	KX / SGMII		



8.5.3.1 Transmit Timestamp

Software indicates to the hardware the packets to be sampled by setting the "TSYN" bit in the transmit context descriptor. The hardware samples the transmission time of packets with an active "TSYN" bit, enabled for the queue by the TSYNENA flag in the transmit queue context. The transmission timestamp is sampled in the PRTTSYN_TXTIME register. It is software's responsibility to read the PRTTSYN_TXTIME_L and PRTTSYN_TXTIME_H registers before sending another packet with active "TSYN" bit.

Note that in order to use the "TSYN" flag in the transmit descriptor, the TSYNENA flag should be set in the transmit queue context. Software should set the TSYNENA flag only for PF queues so VFs can not interfere the Timesync functionality. Software should limit this enablement on a single PF per port which is assigned to control the 1588 logic.

8.5.3.2 Receive Timestamp

XL710 identifies received PTP "event" packets as described in Section 8.5.8. PTP packet reception time is sampled by one of 4 PRTTSYN_RXTIME[n] registers per port which is found "free". Such packets are posted to LAN queues with active an "1588 TS Index" (TSYNINDX) field in the receive descriptor.

TSYNINDX is the index of the PRTTSYN_RXTIME register that contains the packet reception timestamp. Once a PRTTSYN_RXTIME register samples a packet reception time, the register is "locked" until the software reads its value. When accessing this register the software should read first the PRTTSYN_RXTIME_L[n] register and then the PRTTSYN_RXTIME_H[n] register. Following read accesses to the PRTTSYN_RXTIME register, the "lock" is released and the register becomes "free" again.

Note that PRTTSYN_RXTIME registers are not accessible to any of the VFs (so they cannot interfere the proper functionality). The PRTTSYN_RXTIME are 64-bit registers that capture the upper 64 bits of the PRTTSYN_TIME register that belong to the LAN port on which the PTP packet was received (accessible to software by 2 x 32-bit accesses).

8.5.4 1588 Clock Registers

The clock driving the time sync elements is the MAC clock on the port. The MAC clock frequency depends on the link speed (see Table 8-23). Upon change in link speed the PRTTSYN_INC register (shown below) should change accordingly.

Table 8-23. Proposed PRTTSYN_INC as a function of MAC Clock Frequency

Link Speed	MAC Clock Frequency	Sampling Clock Precision	Recommended PRTTSYN_INC defining the 64 MS bits of PRTTSYN_TIME in nsec units
40 Gb/s	625 MHz	± 0.8 ns	6871947673 (0x199999999)
10 Gb/s	312.5 MHz	± 1.6 ns	13743895347 (0x333333333)
1 Gb/s	31.25 MHz	± 16 ns	137438953472 (0x2000000000)

The **PRTTSYN_TIME** is a 96-bit register holding the synchronized clock while its upper 64 bits are meaningful to the software accessible by 2 x 32-bit accesses. Each MAC clock, the PRTTSYN_TIME is increment by PRTTSYN_INC. The lower 32 bits of the PRTTSYN_TIME register are internal, not accessible to software.

PRTTSYN_INC is a programmable 38-bit register accessible to software by 2 x 32-bit accesses (the upper 26 bits of this register are reserved and set to zero). When updating this register, software should set first the PRTTSYN_INC_L register and then the PRTTSYN_INC_H register. Hardware updates the internal value of the PRTTSYN_INC only after these two write cycles. Setting the PRTTSYN_INC by the recommended values in Table 8-23, the upper 64 bits of the PRTTSYN_TIME are defined in nsec units and the lower 32 bits of the PRTTSYN_TIME hold a fraction of a nsec.

Software in slave node adjusts PRTTSYN_TIME in order to track the 1588 timer of the master using one of two methods:

- Direct write access to the upper 64 bits of the PRTTSYN_TIME register. This step could be useful for initial setting of the time during the "Clocks Calibration" phase.
- Fine adjustment during the 1588 "Time Synchronization Phase" using the **PRTTSYN_ADJ** register. The PRTTSYN_ADJ register is composed of a SIGN flag and TSYNADJ field that defines the absolute value of the adjustment. The TSYNADJ value is defined in the units of the PRTTSYN_TIME_L. The PRTTSYN_TIME is updated iteratively by the PRTTSYN_ADJ together with the nominal INC value on each MAC clock as shown below:

```
If (TSYNADJ = 0) then:

PRTTSYN_TIME += PRTTSYN_INC

Else then:     // TSYNADJ is not equal to zero

If positive adjust (SIGN=0) then: PRTTSYN_TIME += PRTTSYN_INC + 2^32

Else : PRTTSYN_TIME += PRTTSYN_INC - 2^32

TSYNADJ -= 1
```

8.5.5 Synchronized Auxiliary Events

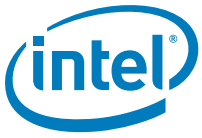
XL710 supports the following global auxiliary events for all ports:

- Synchronized events to global IO signals are described in Section 8.5.5.1.
- Synchronized events to PCI slave access are described in Section 8.5.5.2

8.5.5.1 Auxiliary 1588 IO Signals

XL710 supports 30 global GPIO signals that are controlled by 2 PRTTSYN_AUX[n] registers (n = 0,1). GPIO 'n' functionality is controlled by the GLGEN_GPIO_CTL[n] register (expected to be loaded from the NVM). GPIO's can be assigned to 1588 of a specific LAN port by the PRT_NUM and PIN_FUNC fields in the GLGEN_GPIO_CTL register. Setting the PIN_FUNC field to "Timesync 0" or "Timesync 1" assigns the GPIO to PRTTSYN_AUX[0] or PRTTSYN_AUX[1] respectively.

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 is sampled by the PRTTSYN_EVNT[n] registers ('n' is the matched index to the PIN_FUNC field). When set to output signal the output event time is defined by the PRTTSYN_TGT[n] register ('n' is the matched index to the PIN_FUNC field).



There are several output modes of operation described below. In any of them, the initial state of the GPIO output level can be controlled by setting the INSTNT and the OUTLVL flags in the PRTTSYN_AUX[n] register. All the output modes are enabled by the OUT_ENA flag in the PRTTSYN_AUX[n] register. There are also several input modes of operation controlled by the EVNTLVL field in the PRTTSYN_AUX[n] register.

Synchronized Level Output

This mode is selected by setting the PRTTSYN_AUX[n].OUT_ENA, setting the PRTTSYN_AUX[n].OUTMOD field to "Output Level Mode" and the event time is defined by the matched PRTTSYN_TGT[n] register. When the upper 64 bits of the PRTTSYN_TIME crosses the value of the PRTTSYN_TGT[n], the assigned GPIO transits to the requested level defined by the PRTTSYN_AUX[n].OUTLVL.

Synchronized Flipped Output Signal

Defined by the PRTTSYN_TGT[n] and PRTTSYN_AUX[n] registers similar to the Synchronized Level Output mode. The only difference is that for Flipped Output, the OUTMOD field should be set to "Flipped Output Mode". In this case, each time the upper 64 bits of the PRTTSYN_TIME crosses the value of the PRTTSYN_TGT[n] the IO signal flips its output level.

Synchronized Output Pulse

Defined by the PRTTSYN_TGT[n] and PRTTSYN_AUX[n] registers similar to the Synchronized Level Output mode. The only difference is that the OUTMOD field should be set to "Output Pulse Mode". In this case, each time the upper 64 bits of the PRTTSYN_TIME crosses the value of the PRTTSYN_TGT the GPIO signal flips its output state for 16 x PRTTSYN_AUX.PULSEW plus 1 MAC clocks and then reverts back to its previous level.

Synchronized Clock Output

Defined by the PRTTSYN_TGT[n] and PRTTSYN_AUX[n] registers similar to the Synchronized Level Output mode with the following changes: The OUTMOD field should be set to "Output Clock Mode" and the matched PRTTSYN_CLKO[n] register should be set to 50% of the required output clock duration. Each time the upper 64 bits of the PRTTSYN_TIME crosses the value of the PRTTSYN_TGT the GPIO signal flips its output level. Then, the PRTTSYN_TGT register is reloaded by the hardware to PRTTSYN_TGT[n] plus PRTTSYN_CLKO[n] (while PRTTSYN_CLKO is padded by 32 MS bit zero's). Note that for proper operation the PRTTSYN_CLKO must be larger than PRTTSYN_INC + 2^{32} and for reasonable accuracy the PRTTSYN_CLKO should be significantly larger than PRTTSYN_INC + 2^{32}.

Sampling Input Event

XL710 can capture the time on which an event is sensed on any of the 1588 auxiliary signals. The event time is captured by one of two matched 64 bit event time registers named PRTTSYN_EVNT[n]. The sampled input event type is defined by the EVNTLVL field in the PRTTSYN_AUX[n] register, equals to one of the following options: Disable; Rising Edge; Falling Edge; Any Transition. When the defined transition is sampled by the hardware (synchronized to the MAC clock), the upper 64 bits of the PRTTSYN_TIME are latched by the matched PRTTSYN_EVNT[n] register. Once the event is sampled, the PRTTSYN_EVNT register is locked, keeping the event time till the software reads it.

8.5.5.2 1588 PCI Slave Access

Following a write access to the PRTTSYN_AUX[n] while setting the SAMPLE_TIME flag, the hardware samples the PRTTSYN_TIME to the matched PRTTSYN_EVNT[n] register (the upper 64 bits of the PRTTSYN_TIME). This functionality can be useful to synchronize between a system timer accessible to the CPU and the PRTTSYN_TIME.

8.5.6 Interrupts

XL710 can trigger an interrupt on any GPIO regardless of whether it is enabled for 1588 functionality (see Section 7.5.2.4). In addition, XL710 can generate a 1588 interrupt for one of the following events (if enabled by the TIMESYNC flag in the PFINT_ICR0_ENA register):

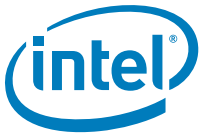
- Packet transmission time is latched by the PRTTSYN_TXTIME registers while this interrupt is enabled by the TXTIME_INT_ENA flag in the PRTTSYN_CTL0 register.
- Input event is latched by one of the PRTTSYN_EVNT registers while this interrupt is enabled by the EVENT_INT_ENA flag in the PRTTSYN_CTL0 register.
- One of the time registers has expired while this interrupt is enabled by the TGT_INT_ENA flag in the PRTTSYN_CTL0 register (a target time register expires when PRTSYN_TIME register is larger or equal than the target time register).

Regardless of interrupt enablement, the above events are reported in the PRTTSYN_STAT_0 and PRTTSYN_STAT_1 registers. Following an interrupt assertion, Software should read these registers to identify the cause. The registers are cleared on read, so they report an updated status since the last time they were read by the software.

8.5.7 1588 Initialization Flow

This section describes the PF software initialization flow required to activate the 1588 logic.

- Check the PF_ID in the PRTTSYN_CTL0 register if the PF is expected to control the 1588 logic.
- The 1588 CSRs 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:
  - Set PRTTSYN_CTL0 and PRTTSYN_CTL1 to the required 1588 packet type as well as required interrupt functionality.
  - Clear any optional residuals reported in the PRTTSYN_STAT register.
  - Read all receive timestamps releasing them for nominal operation (PRTTSYN_RXTIME registers).
  - Set the PRTTSYN_AUX register as required for the AUX functionality.
  - The timer and the increment values are expected to be programmed as part of the nominal operation (PRTTSYN_TIME and PRTTSYN_INC registers).
- Assign a transmit queue for 1588 transmission and enable TSYNENA flag in its queue context.
- Optionally assign a receive queue. The EtherType Classification filter can be used to direct Layer 2 or UDP 1588 packets to a specific LAN receive queue.



8.5.8 PTP Packet Recognition

The time sync implementation supports both the 1588 Version 1 (V1) and Version 2 PTP frame formats (V2). The V1 structure is expected only as UDP payload over IPv4 while the V2 is expected over L2 with its Ethertype or as a UDP payload over IPv4 or IPv6. Note that XL710 accepts the expected packet formats but does not enforce these UDP/IP rules.

The 802.1AS uses only the layer 2 V2 format. Note that PTP frame structure is not supported in the XL710 for tunneled packets. The Layer 2 packet format and UDP packet format are shown in Figure 8-14. The PTP Message V1 and V2 formats are shown in Figure 8-25. The packet format supported by XL710 for receive timestamp is controlled by the parameters detailed in Figure 8-24.

Note: Correct UDP checksum is not a criteria for PTP packets recognition by the XL710. Still, such a packet is reported to software with checksum error. The software can ignore the packet; but should read the register that carries the timestamp to release it for upcoming packets.

Note: the XL710 does not recognize tunneled PTP packets

Table 8-24. Receive TimeStamp Control Parameters

Parameter	Register	Functionality
TSYNTYPE	PRTTSYN_CTL1	Controls which packets are sampled by the 1588 logic. It can be one of the following: • L2 Version 2 packets while message type is defined by the PRTTSYN_CTL1 register • UDP Version 1 packets while the UDP ports are defined by UDP_ENA and message type is defined by the PRTTSYN_CTL1 register • L2 & UDP Version 2 packets while UDP ports and message type are defined as above • L2 & UDP Version 2 event packets while UDP ports are defined by the UDP_ENA and the Message Type < 8
UDP_ENA	PRTTSYN_CTL1	Enable the UDP port numbers that are recognized as 1588 packets. It could be one of the following options: • No UDP packet recognition • UDP port number equals to 0x013F • UDP port number equals to 0x0140 • UDP port numbers equals to either 0x013F or 0x0140
V1MESSTYPE0, V1MESSTYPE1	PRTTSYN_CTL1	Controls the PTP V1 Message Type for sampled timestamp
V2MESSTYPE0, V2MESSTYPE1	PRTTSYN_CTL1	Controls the PTP V2 Message Type for sampled timestamp

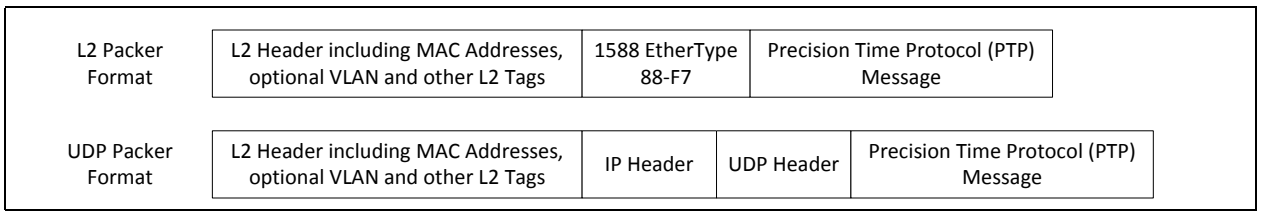


Figure 8-14. 1588 Packet Format

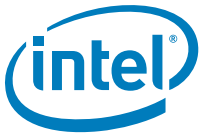
Table 8-25. PTP Header Format (V1 and V2 formats)

Offset in Bytes	V1 Fields	V2 Fields	
		bits: 7 . . . 4	bits: 3 . . . 0
0	versionPTP	transportSpecific[1]	Message Type
1		Reserved	versionPTP
2 - 3	versionNetwork	messageLength	
4	Subdomain	SubdomainNumber	
5		Reserved	
6 - 7		flags	
8 - 15		Correction Field	
16 - 19		reserved	
20	Message Type	Source Port ID	
21	Source communication technology		
22 - 27	Sourceuuid		
28 - 29	sourceportid		
30 - 31	sequenceId	sequenceId	
32	control	control	
33	reserved	logMessagePeriod	
34 - 35	flags	n/a	

1. Should all be zero.

Table 8-26. PTP V2 and V1 Message Types

Message Type	Message Class	V2 Message Type	V1 Message Type
Sync	Event	0x0	0x00
Delay_Req	Event	0x1	0x01
Pdelay_Req	Event	0x2	N/A
Pdeley_Resp	Event	0x3	N/A
Reserved	-	0x4 - 0x7	N/A
Follow_Up	General	0x8	0x02
Delay_Resp	General	0x9	0x03
Pdelay_Resp_Follow_Up	General	0xA	N/A
Announce	General	0xB	N/A
Signaling	General	0xC	N/A
Management	General	0xD	0x04
Reserved	-	0xE, 0xF	0x05 - 0xFF



8.6 Software Timer

XL710 includes a 32-bit counter in GLVFGEN_TIMER register which is based on a free running 1 usec clock. The counter is cleared by Core Reset (CORER) and increments after being cleared. The timer wraps around in about 70 minutes.

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

9.0 FCoE Engine

9.1 Introduction

Fibre Channel (FC) is the predominant protocol used in Storage Area Networks (SAN). Fibre Channel over Ethernet (FCoE) is based on a simple layer 2 Ethernet encapsulating of FC frames. Due to its simplicity it enables smooth migration of FC to Ethernet.

The FC protocol is based on high reliability of the communication link between the initiator and the storage target. It assumes an extremely low error rate of 10-12 and no packet drop. DCB extends Ethernet through class-based flow control and Quantized Congestion Notification (QCN) in such a way that FC-like no-drop is guaranteed as required by FC. Figure 9-1 shows a connection between an FCoE initiator and both legacy FC and FCoE targets.

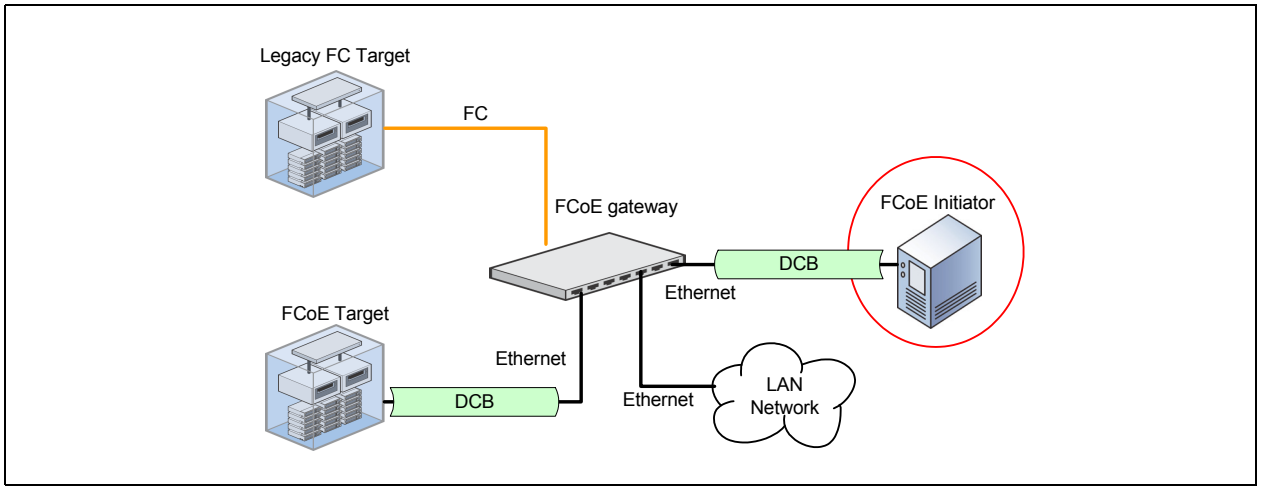


Figure 9-1. Connecting an FCoE Initiator to FC Targets

Existing FC HBAs provide full offload of the FC protocol to maximize storage performance. In order to compete with this market, XL710 offloads the main data path of I/O Read and Write commands detailed in this chapter.

9.1.1 Terminology

Useful background on FC framing and its Ethernet encapsulation can be found in Section 18.5 FCoE Framing. More comprehensive material can be found in the FIBRE CHANNEL FRAMING AND SIGNALING-2 (FC-FS-2) specification as well as the Fibre Channel Framing and Signaling - 3 (FC-FS-3). Following are some of the most common terms used extensively in the sections that describe the FCoE functionality.

Initiator - The node that initiate the transaction (FC read or write) with the target. It is denoted also as "Originator". In most cases the initiator is the server requesting storage services from the disk.

Target - The node that accepts the transaction (FC read or write) from the Initiator. It is denoted also as "Responder". In most cases the target is the remote storage disk.

FC Exchange - Complete FC read or FC write flow. It starts with a read or write request by the initiator (the host system) until it receives a completion indication from the target (the remote disk).

FC Sequence - An FC exchange is composed of multiple FC sequences. An FC sequence can be single or multiple frames that are sent by the initiator or the target. Also, each FC sequence has a unique Sequence_ID. As part of ETSO offloads (explained later on in this section), the hardware auto-increments the Sequence_ID. There are some targets that do not support the same Sequence_ID value on any exchange with a specific initiator. There are other targets with relaxed restrictions that do not support the same Sequence_ID value only within the same exchange. It is software responsibility to choose the initial Sequence_ID that will not break the target rules when incremented by the hardware.

FC Frame - FC frames are the smallest units sent between the initiator and the target. The FC-FS-2 specification defines the maximum frame size as 2112 bytes. Each FC frame includes an FC header and optional FC payload. It also may include extended headers and FC optional headers. Extended headers other than Virtual Fabric Tagging (VFT) are not expected in an FCoE network and FC optional headers are not used in most cases as well.

Data Frame - FC frames that carry read or write data.

FCP XFER_RDY Frame - FC control frame sent from the target to the initiator. It defines the write data that the target is ready to accept from the initiator. In this document these packets are called simply "RDY" packets.

FCP_RSP Frame - FC control frame sent from the target to the initiator. It defines the completion of an FC read or write exchange. In this document these packets are called simply "RSP" packets.

Single Send - The hardware gets from software a single packet for transmission. Supported "Per Packet" offloads are described in Section 9.2.1.

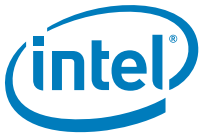
TSO and ETSO - Segmentation offloads enabling the software to initiate multiple packet transmission in a single instruction. TSO is described in Section 9.2.2 and ETSO is described in Section 9.2.3.

Single Receive - The hardware post to software each packet in a dedicated structure composed of a single or multiple receive descriptors. Supported "Stateless Receive" offloads are described in Section Section 9.3.1.

DDP - Direct Data Placement handle multiple packet reception directly to receive buffers provided by the storage stack. DDP is described in Section 9.3.2.

9.1.2 FCoE support - Summary

- FC CRC offload



  - — Integrity check on receive and generate plus insert on transmit
- Packet Filtering
  - — Identify FCoE packets by EtherType value equals to 0x8906
  - — Identify FIP packets by EtherType value equals to 0x8914. FIP packet are handled as L2 packets directed to receive queues by the EtherType filters.
  - — Filtering to VSI by MAC/VLAN filters
  - — Filtering to FCoE context by: Exchange ID, Optional VLAN (if header present), FC Destination ID, Source MAC address
  - — Filtering to LAN queues by OX Exchange ID (supporting multiple LAN receive queues for CPU load balance).
- Segmentation offload
  - — Transmit Segmentation (TSO) as provided by XL710's predecessor
  - — Enhanced TSO (ETSO) for targets offloading the RSP packet transmission
- State-full offload
  - — Direct data placement (DDP) as provided by XL710's predecessor. Offload completely processing of received data packets while posting its payload directly to the application buffers.
  - — Up to 4K contexts and 4K filters per function (DDP)
  - — Context programming interface via LAN Tx queue. Bad status reporting on the LAN Rx queue.
  - — Context programming commands: Add / Remove context

9.1.3 FCoE Enablement and Cross Functionality

The table below lists FCoE offload functionality combined with other device capabilities and network configurations.

Tx/Rx	Cross Function	Requirements
Tx/Rx	Global FCoE offload Enablement	FCoE offload is enabled in the GLGEN_STAT.FCOEN bit
Rx	FCoE reception	FCoE and FIP packet reception is enabled per queue by the FCENA flag in the receive queue context
Rx	FCoE DDP filter Enablement	FCoE context filters are enabled per VSI by the FCOE_ENA flag. This functionality is enabled only for the PFs. Software should not set the FCOE_ENA flag on VFs and VM VSI's.
Rx	LAN Rx queue for DDP	For any DDP context there is an associated LAN Rx queue. If the associated LAN Rx queue is not enabled, then the DDP functionality for the specific DDP context is disabled as well.
Tx/Rx	TC and UP for FCoE DDP	The traffic class and user priority for the Tx LAN queue that is used for the context programming must be the same as the TC and UP of the received traffic. In addition, all DDP programming descriptor (add / remove DDP context) must be initiated from the same Tx queue.
Tx/Rx	Jumbo frames	Maximum expected clear text FC frame size is 2140 bytes (FC header + FC payload + FC CRC). In order to enable FCoE traffic, jumbo packet reception should be enabled.
Tx/Rx	Traffic rate control	FC traffic relies on a high quality link that guarantees no packet loss. It is expected that any loss-less traffic protocols supported by the network are enabled by XL710 as well.
Tx/Rx	FC and PFC	
Tx/Rx	TCP/IP and UDP/IP offload	FCoE traffic is L2 traffic (not over IP). Any setting of TCP/IP and UDP/IP offload capabilities are not applicable and do not impact FCoE offload functions.

Tx/Rx	Cross Function	Requirements
Tx/Rx	Virtualization	FCoE offloads is supported only for the PFs. It is not supported for VMs nor VFs.
Rx	Header Split option	Header split should not be enabled in the LAN queues used for FCoE traffic. Both HSPLIT_0 and HSPLIT_1 parameters in the queue context should be cleared.
Rx	Ethernet CRC check	Packets with any MAC error (Ethernet CRC error, short packet, etc.) are not processed by the FCoE offload logic. Such packets are posted to the legacy Rx queues (if save bad frames is enabled).
Tx/Rx	VLAN header	It is assumed that any FCoE has a VLAN header. In the case of double VLAN mode, the packet must have the two VLAN headers.
Tx/Rx	SNAP packet	XL710 does not provide FCoE offload for FCoE frame over SNAP.
Tx/Rx	Security Offload MACsec and FC-SP (ESP)	When packets carry MACsec encapsulation, any FCoE offloads are not enabled. When packets carry ESP encapsulation only FC CRC is enabled and receive classification to FCoE LAN queues.
Rx	Ethernet padding extraction	There is no enforcement on the Ethernet padding extraction. When DDP is enabled, hardware posts the FC payload to the user buffers. When DDP is not enabled the entire packets are posted to the legacy receive queues with or without the Ethernet padding according to the device setting.

9.1.4 Read and Write Exchanges Flows

The following diagrams (Figure 9-2 and Figure 9-3) show examples for Fibre Channel class 3 read and write exchange flows. These diagrams illustrate a high level description of potential "stream based" offloads supported by XL710 which includes: DDP, TSO and Enhanced TSO. These offload as well as any "per packet" processing are detailed in the following sections.

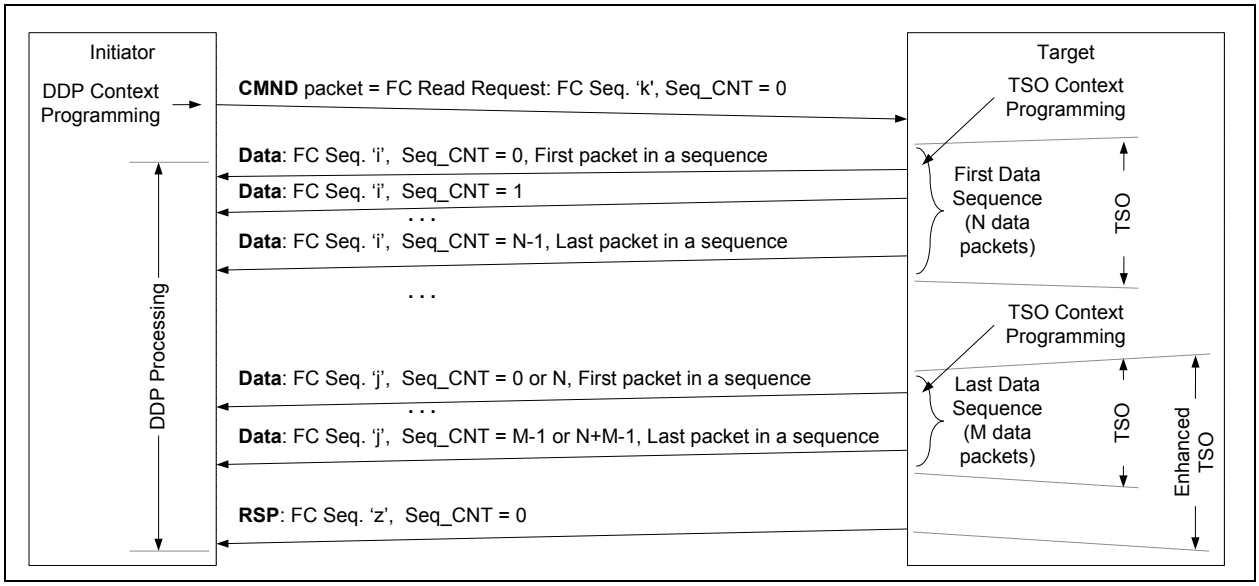


Figure 9-2. Example for FC Class 3 Read Exchange Flow

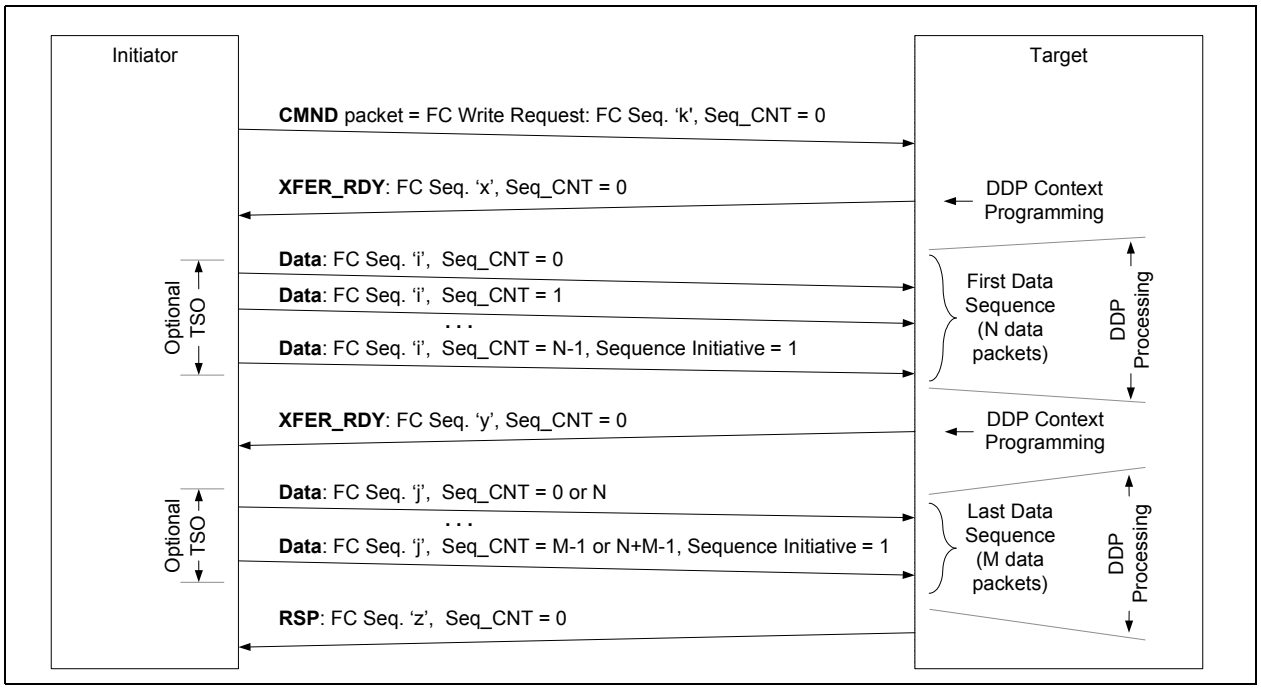


Figure 9-3. Example for FC Class 3 Write Exchange Flow

9.2 FCoE Transmit Operation

XL710 supports the following transmit offload capabilities described in the following sections:

- Per Packet FCoE Transmit Offload (see Section 9.2.1)
- Transmit segmentation - TSO (also named LSO, see Section 9.2.2)
- Enhanced segmentation for target applications (See Section 9.2.3)

9.2.1 Stateless Per Packet FCoE Transmit Offload

Device processing for any transmitted FCoE packet is described the following sub sections:

- Layer 2 offloads like any other LAN traffic (See Section 9.2.1.1)
- FC padding insertion (See Section 9.2.1.2)
- SOF flag placement (See Section 9.2.1.3)
- EOF flag insertion (See Section 9.2.1.4)
- FC CRC calculation and insertion (See Section 9.2.1.5)

Any FCoE transmit offload is enabled by setting the FCOET flag in the transmit data descriptor. The FCOET flag must be set as well for a packet that is used only for FCoE context programming . See Section 8.4.2.4 for the transmit descriptors of a single send.

9.2.1.1 Layer 2 offload

Single send is indicated to the hardware by the transmit data descriptor. Listed below are some notes related to FCoE packets:

- L2 Tag can be inserted from the data descriptor as any LAN packets by setting the IL2TAG1 flag and providing the L2 tag in the L2TAG1 field. Similarly, a second L2 Tag can be inserted from the context descriptor by setting the IL2TAG2 flag and providing the L2 tag in the L2TAG2 field.
- The FCOET flag should be set and the IIPT field should be cleared
- The EOFT defines the EOF flag as explained in Section 9.2.1.4
- FC CRC offset parameters (MACLEN, FCoELEN and FCLEN) are explained in Section 9.2.1.5

For single send on which two L2 Tags offload are required by the driver, the FCoE Transmit context descriptor should be used as follow:

- The OPCODE field should be set to Single send
- The IL2TAG2 flag should be set and the L2 tag is defined by the L2TAG2 field

The L2 Tag 2 can be added to TSO and ETSO as well (TSO and ETSO are described later on in this chapter). In these cases the L2 Tag offload is defined by the FCoE Transmit context descriptor (the same as the single packet).

9.2.1.2 FC Padding Insertion

FC frames always consist of a whole number of four bytes. If the user data is not composed of a whole number of four bytes, then the FC frames contain padding bytes with a zero value. The length of the padding bytes can be between zero to three to complement the missing bytes.

The length of the padding bytes is indicated by the Fill_Bytes field in the FC header. This field is used only by the receiving end node to extract these bytes and not by the transmission hardware. Instead, the hardware checks the transmit buffer(s) size indicated by the software. The length of the padding bytes added by hardware equals to the 2’s complement of the 2 LS bits of the total size of the data buffers. In case of TSO the hardware checks for the required padding only on the last packet. All packets but the last one must be whole number of 4 bytes (even without padding).

9.2.1.3 SOF Flag Placement

During a single send, the SOF field is taken as is from the FCoE header in the data buffer.

9.2.1.4 EOF Flag Insertion

In a single send, TSO, ETSO XL710 inserts the End of Frame field according to the EOFT setting in the transmit data descriptor. The EOFT flag in the transmit data descriptor define the EOF codes as listed in the Table 9-1 below. Following the EOF flag the hardware inserts additional 3 zero bytes according to the FCoE frame format.

Table 9-1. EOF Codes in Single Send

EOF Bits in the Context Descriptor	Transmitted EOF Code
00	EOFn (0x41)
01	EOFt (0x42)
10	EOFni (0x49)
11	EOFa (0x50)

9.2.1.5 FC CRC Insertion

FC CRC calculation is one of the most CPU intensive tasks in large transactions. XL710 offloads the FC CRC calculation for a single send TSO and ETSO. XL710 calculates and adds the FC CRC before packet transmission but after the required FC padding bytes are already added.

The CRC polynomial used by the FC protocol is the same one as used in FDDI and Ethernet as shown below while the CRC bytes are transmitted in little endian byte ordering (LS byte first on the wire):

X32+X26+X23+X22+X16+X12+X11+X10+X8+X7+X5+X4+X2+X+1.

The size of FC Frame on which FC CRC is calculated is defined for single send and TSO in the context and data descriptors as shown in the Figure 9-4 below.

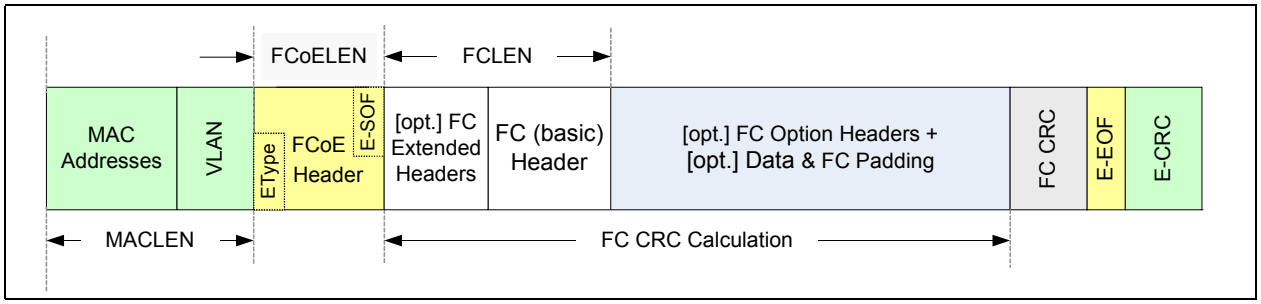


Figure 9-4. FCoE Frame and Relevant Transmit Descriptor Parameters

FC CRC Calculation Beginning

FC CRC calculation starts after the FCoE header. It equals to byte offset of MACLEN + FCoELEN.

FC CRC Calculation End FC CRC calculation ends at the end of the FC Payload LEN (eight bytes before the Ethernet CRC).

9.2.2 FCoE Transmit Segmentation Offload (TSO)

FCoE segmentation enables the FCoE software to initiate a transmission of multiple FCoE packets up to a complete FC sequence with a single header in host memory (by a single instruction).

9.2.2.1 Transmit Segmentation Offload Flow

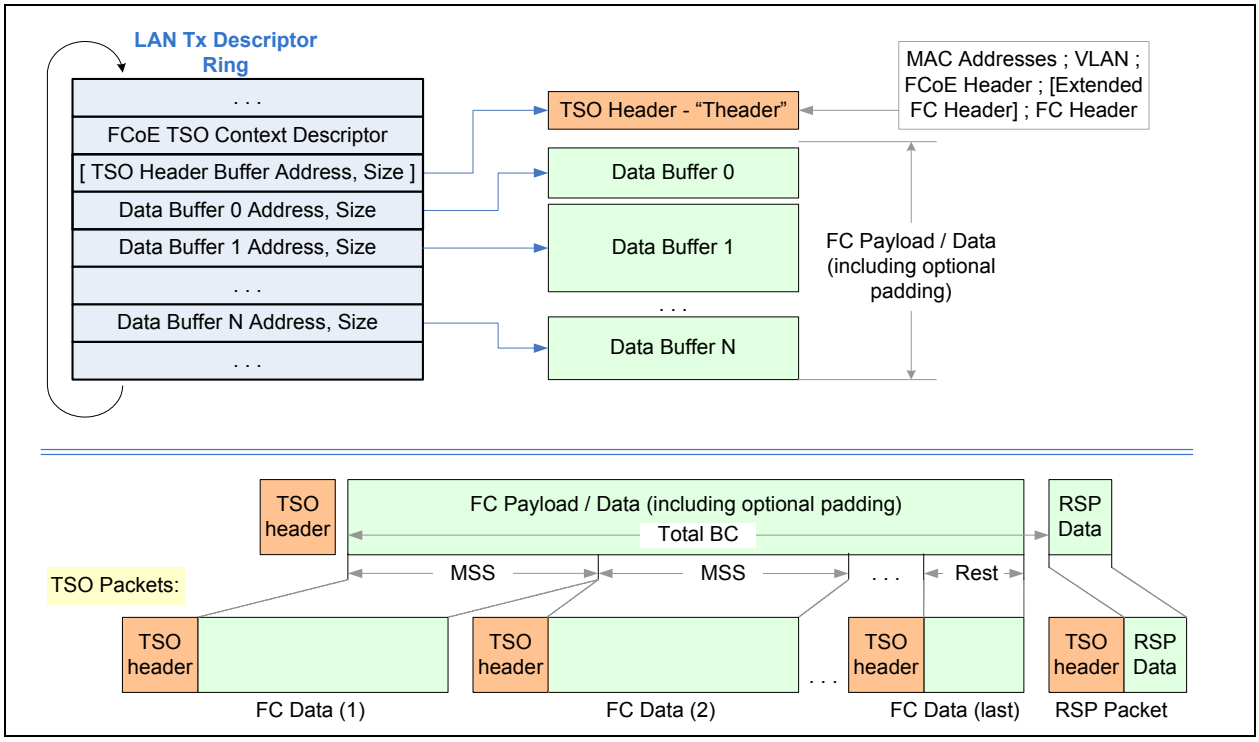
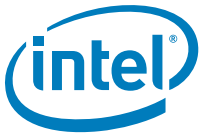


Figure 9-5. TSO Buffer List and the Transmitted packets

- The software prepares and lock the data buffers to be transmitted and prepares a descriptor list that indicate these buffers. These Data descriptors have the same structure as the LAN data descriptors described in Section 8.4.2.1.1.
- The TSO can be used to send a single packet and up to a block size of 256KB minus 1.
- The software indicates the above buffers and additional parameters listed below as illustrated above in Figure 9-5:
  - FCoE transmit context descriptors in the LAN Tx queue define the segmentation parameters.
  - The FCoE transmit context descriptor is described in Section 8.4.2.4.
  - The EOP flag should be set only on the last data descriptor (as done for the LAN TSO)
- The hardware updates the Head pointer of the LAN Tx queue after it fetches the FCoE transmit context descriptors and the descriptors of the data packet(s) depending on 'RS' bit setting in the transmit descriptor.
- The size of the whole data payload to be segmented is defined by the "TLEN" field in the FCoE transmit context descriptor and the size of each packet payload is defined by the "MSS" field in the FCoE transmit context descriptor.
  - XL710 rounds up the "TLEN" to whole number of DWords (4 byte). It enables the software to define data buffers excluding the FC padding bytes. If the "TLEN" is not set originally to whole number of DWords the hardware pads the required zero bytes at the end of the last data frame in the TSO as described in Section 9.2.1.2 for a single packet transmission.
  - The MSS must be whole number of DWords.



- All packets in the TSO are transmitted in a single FC sequence.
- The hardware calculates and inserts the FC CRC to each packet as described in Section 9.2.1.4.
- The transmitted packets' headers are based on the THeader while dynamic are described below.
  - The THeader must be defined in a single buffer while the payload may span on multiple buffers and optionally it could be on the same buffer as the THeader.
- Dynamic fields in the transmitted packets are described in the Table 9-2 below. Note to the following cases of TSO:
  - The TSO may start a sequence or may not. If it starts a sequence the software should set the SOF flag in the "THeader" to SOFi2 or SOFi3 depending the class of the exchange. Otherwise, the SOF in the "THeader" should be set to SOFn2 or SOFn3 depending the class of the exchange. The exchange class is defined by the OPCODE field in the FCoE Transmit context descriptor.
  - The TSO may end a sequence or may not. If it ends a sequence the software should set the "End_Sequence" flag in the "THeader". If the "End_Sequence" flag is set, the last packet in the TSO ends a sequence.
- Auto transmission of RSP packet can be enabled as described in Enhanced TSO flow in Section 9.2.3.

9.2.2.2 TSO Context

The TSO context is programmed by the FCoE context descriptors illustrated in the Figure 9-5 above and described in subsections of Section 8.4.2.4.

9.2.3 Enhanced Segmentation Offload (ETSO)

Enhanced TSO provides incremental step on top of plain segmentation described in Section 9.2.2.1. It is optimized for Target response to read exchange request from the initiator. It enabled the software on the Target to send back to the initiator both the last data sequence and the RSP packet by a single command to the hardware. The software activates the ETSO flow by setting the "OPCODE" field in the "FCoE FCoE Transmit Context Descriptor" to ETSO. In this case, following the last data packet in the TSO, the HW auto sends the RSP packet using the header of the Data frames in the TSO while modifying the fields described in the Table 9-2 below. It then increments the Head pointer in the LAN Tx descriptor ring to the next descriptor that follows all the data descriptors of the ETSO.

The payload of the RSP packet is assumed to be 24 bytes. These bytes should be provided by the software at the end of the data buffers on its own dedicated buffer. These 24 bytes are indicated in the last data descriptor but must not be included in the "TSO Total Length (TLEN)" programmed by the FCoE context descriptor. The software should set the EOP flag on the last descriptor of the ETSO that includes the extra 24 bytes of the RSP payload.

9.2.4 Dynamic parameters for TSO and ETSO

The dynamic parameters in the transmitted packets of TSO and ETSO are described in Table 9-2 below.

Table 9-2. Dynamic Fields in TSO and ETSO

Parameter	First packet	Middle packet	Last data packet	RSP packet
	TSO / ETSO	TSO / ETSO	TSO / ETSO	ETSO
SOF	Theader (1)	SOFn2 or SOFn3 (2)	SOFn2 or SOFn3 (2)	SOFi2 or SOFi3 (2)
EOF	EOFn	EOFn	Set by the EOFT flag (1)	EOFt
R_CTL	Theader	Theader	Theader	0x07
Relative_offset	Theader	Theader	Theader	0
End_Sequence	0	0	Theader (1)	1
Sequence_Initiative	0	0	Theader (1)	1
Last_Sequence	0	0	Theader (1)	1
Continue_Sequence_Condition	0	0	Theader (1)	0
Abort_Sequence_Condition	Theader	Theader	Theader	0
Fill_Bytes	0	0	Theader (1)	0
Parameter Field if RELOFF flag in the Context descriptor is cleared	Theader	Theader	Theader	0x0
Parameter Field if RELOFF flag in the Context descriptor is set	Equals to the PARAM field in the THeader plus the total transmitted bytes excluding this packet			0x0
Sequence_ID	Theader	Theader	Theader	Theader + 1
Sequence_Count if CLRSEQ flag in the Context descriptor is cleared	Theader (1)	Increment by 1 for each packet	Increment by 1 for each packet	Increment by 1 for each packet
Sequence_Count if CLRSEQ flag in the Context descriptor is set	Theader (1)	Increment by 1 for each packet	Increment by 1 for each packet	0x0
All other header parameters	Theader			
FC (data)	Taken from the data buffers			

Note 1: Setting of the flag in case of TSO/ETSO composed of a single data packet.
Note 2: Selection between SOFx2 and SOFx3 (x='i' or 'n') is made according to the OPCODE field in the Context Descriptor

9.3 FCoE Receive Operation

XL710 can offload the following tasks from the CPU while processing FCoE receive traffic: FC CRC check and Direct Data placement (DDP). These offload are described in the following sections:

- Stateless (per packet) FCoE receive offload (see Section 9.3.1)
- Direct Data Placement offload (see Section 9.3.2)

The diagram below in Figure 9-6 describes the received FCoE frame processing flow with and without DDP. In the descriptive text, any packet that fails a specific step, is not candidate for the proceeding steps.

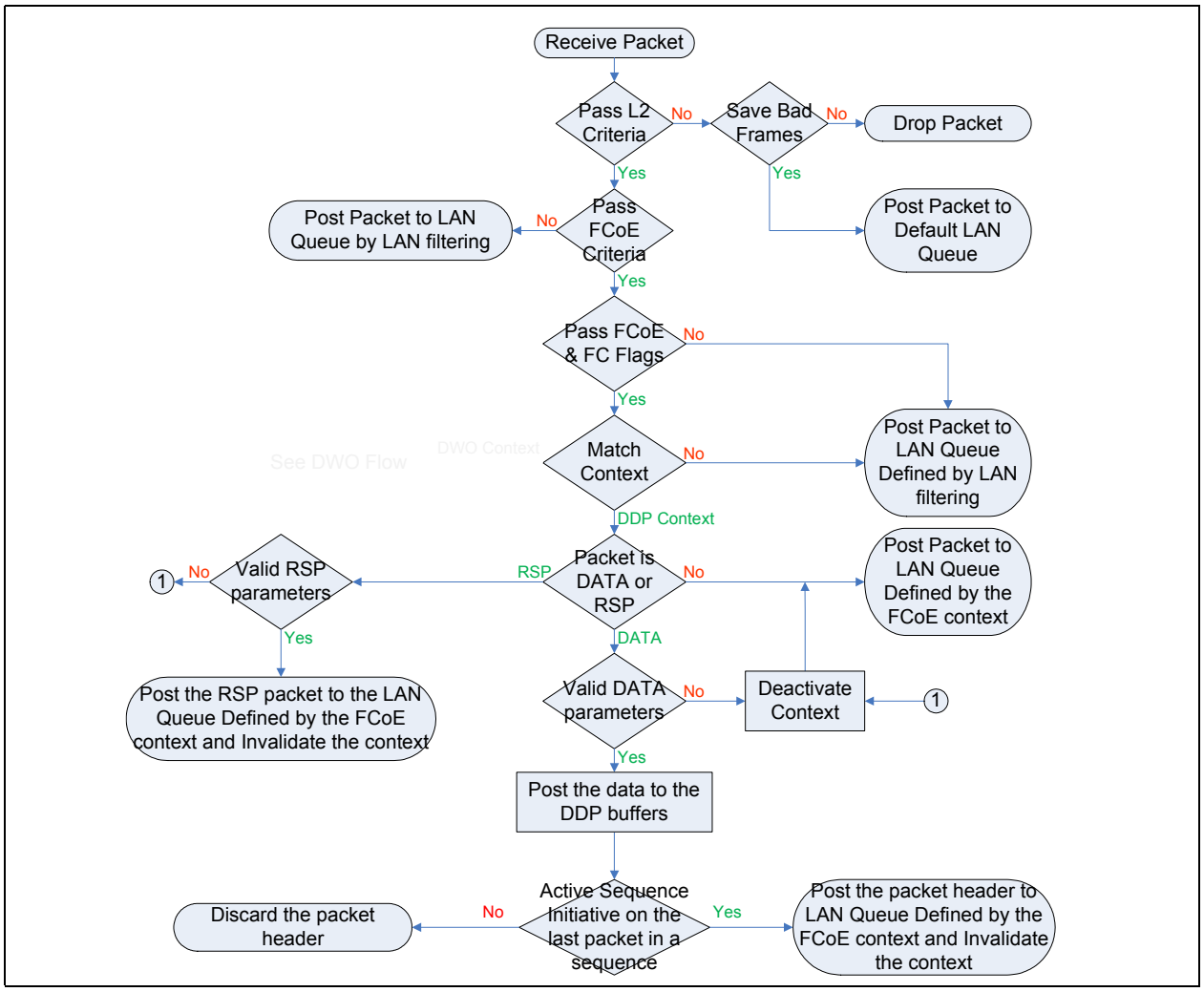


Figure 9-6. Receive Processing and DDP validity Flow Diagram

- Check the following stateless criteria as detailed in Section 9.3.1. This section describes also how packets that fails these criteria are handled.
  - Pass Ethernet (L2) processing
  - Identified as FCoE packet
  - Pass FC processing criteria
- Check for FCoE context match as described in Section 9.3.2.4.1. This section describes also how packets that fails these criteria are handled.
- Candidate for DDP processing can be either Data or RSP or RDY packets. Packets that are not identified as Data or RSP or RDY, bypass the DDP filters and are posted to LAN queue defined by the LAN filters .
  - RSP packet identification is described in Section 9.3.2.4.2. If the packet is identified as viable RSP packet then the packet is posted to the LAN queue defined by the FCoE context with good context match indication and also auto-invalidate the context.

— Data packet identification is described in the Section 9.3.2.4.3. If the packet is identified as a viable Data packet for DDP it is processed by the DDP context as described in Section 9.3.2.1. Otherwise (exceptional case), it is posted to the LAN queue defined by the FCoE context with matched exception error indication and also deactivate the context. In this case, further matched packets will be directed to the LAN queue defined by the context.

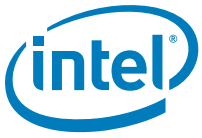
9.3.1 Stateless Receive FCoE Processing

Received FCoE frame are subjected to the following stateless processing. Packets that meet all criteria below are candidates for DDP processing. Packets that do not pass all the criteria below are handled as described below. Packets that are posted to the receive LAN queue includes the complete packet starting with the L2 header and up to including the last DWord that contains the EOF flag. The Ethernet CRC is optionally posted as well if not gated by the CRCStrip flag in the receive LAN queue context.

- Ethernet L2 packet criteria. Packets that fail these criteria are discarded in nominal operation. If bad frame reception is enabled, the packets are posted to the default LAN queue of the virtual port.
  — The packet does not pass the Ethernet length criteria or it has CRC errors. Bad frame reception is enabled by the SBP flag in the PRT_SBPVSI register.
- FCoE packet Identification is done according to the EtherType value associates with FCoE. Packets that are not identified as FCoE are posted to LAN queues according to LAN queue filters of the virtual port.
- Fibre Channel packet processing. Packets that fail this criteria are posted to LAN queues defined by the Hash filter for FCoE packets defined without checking for potential matched FCoE context.
  — The FCoE version in the received frame is equal to zero .
  — The packet carries FC class 2 or class 3 (magnetic tapes storage and SCSI over FC). XL710 identifies these classes by the SOF flag in the FCoE header equals to SOFi2 or SOFn2 or SOFi3 or SOFn3. Note that the SOF flag is posted to host buffers only for packets received to the LAN queues (no DDP).
  — The EOF field is found to be one of the following: EOFn, EOFt, EOFa or EOFni .
  — The received EOF field matches the state of the F_CTL->"End_Sequence" flag: Last packet in a sequence is indicated by active "End_Sequence" flag and EOF equals to EOFt. Any other packets in a sequence are indicated by cleared "End_Sequence" flag and EOF equals to EOFn.
  — Active "Sequence_Initiative" is permitted only in the last packet in a sequence.
  — The following obsolete flags in the F_CTL are found cleared: bit 9 (was "Retransmitted Sequence"); bit 8 and bits 7-6 (were "Continue Sequence Condition")
  — The packet has no FC CRC error. FC CRC processing is described in Section 9.3.1.1.
  — FC Extended headers: The packet may contain only VFT extended header or no extended headers. No action is done on the VFT header other than skipping it and identifying the basic FC header afterward.
  — FC Option Headers: The packet does not carry any optional headers.

9.3.1.1 FC CRC Integrity Check

XL710 checks the FC CRC integrity for all FCoE received packets and also checks that the packets are composed of whole number of 4 bytes. It reconstructs the FC CRC using the CRC polynomial described in Section 9.2.1.5. The FC CRC covers the whole FC frame starting after the FCoE header up to the received FC CRC bytes. The reconstructed FC CRC is compares against the received FC CRC. The frame is considered as "good" FC packet if the reconstructed FC CRC matches the received one and the packet



is whole number of 4 bytes. Otherwise the packet is considered as a "bad" one. The FC CRC bytes are posted only to the receive buffers in the LAN receive queues (no DDP). The FC CRC bytes are extracted from the packet before it is posted to the DDP buffers.

9.3.1.2 FC Padding Bytes

Packets directed to the LAN queues are posted with the FC padding bytes if exist. Packets directed to the DDP buffers are always posted without the FC padding bytes.

9.3.1.3 FCoE Receive Filtering

Filtering to VSI is based on MAC address and VLAN filters as any other packets. Filtering to LAN receive queues, for packets that do not match a DDP context, is based on the originator exchange ID OX_ID. FCoE contexts is identified according to a perfect match of the exchange ID (OX_ID or RX_ID depending on F_CTL -> "Exchange_Context"), D_ID, Source MAC address VLAN tag and FC_TYPE.

9.3.2 FCoE Direct Data Placement (DDP)

FCoE receive DDP offload provides two main components that reduce CPU processing overhead:

- Save a data copy by posting the received FC payload directly to the buffers provided up-front by the FCoE stack. These buffers are named in this document as "DDP buffers".
- Minimize further CPU cycles by posting to the software only a limited number of required packets (or packets headers). In nominal operation the initiator's software processes only the FCP RSP packet and the target's software processes only the last data packet on each "burst" ("burst" is the requested data indicated by the FCP RDY packet).

DDP offload is supported for the PFs with direct programming of the DDP context by the owning functions. The following sections detail the FCoE receive DDP functionality as follows:

- FCoE receive Direct Data Placement flow (Section 9.3.2.1)
- FCoE Context composed of Filter context and DMA context (Section 9.3.2.3)
- FCoE Context Management: programming, invalidation... (Section 9.4.1)
- Exceptions Handling (Section 9.4.3)

9.3.2.1 FCoE DDP Flow

This section talks about DDP flow at the Initiator and the Responder. The Figure 9-7 below shows the data structures related to DDP as detailed in the following sections.

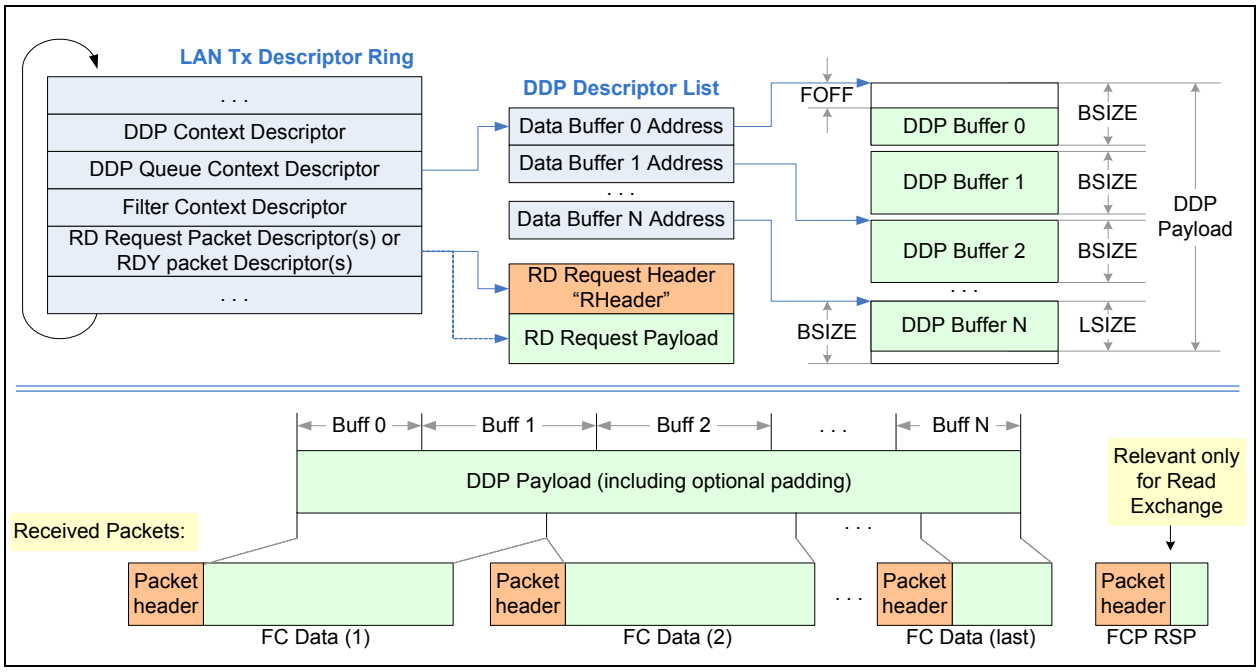
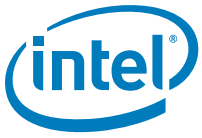


Figure 9-7. DDP Context programming and its Buffer List

9.3.2.1.1 Initiator DDP Flow

Listed below is a description of FC read exchange using the DDP offload in the Initiator. See also Figure 9-2 for an example for FC Class 3 Read Exchange Flow.

- The software prepares the following data structures illustrated in Figure 9-7
  - FC Payload buffers for the DDP - named "DDP Buffers"
  - DDP Descriptor list that indicate the above structures
  - Read request packet
- The software programs the DDP context, indicating the DDP descriptor list and the Filter parameters to the hardware by the LAN Tx context descriptors as illustrated in Figure 9-7. These parameters are detailed in Section 9.3.2.3.
  - The hardware processes the FCoE context descriptors and the read request packet descriptor(s) as atomic operation. Once the hardware fetches the Read request packet buffers the FCoE context is programmed and the transmit queue head pointer is updated to the next descriptor that follows the read request packet.
  - The read request header must be defined in a single buffer while the payload may span on multiple buffers and optionally it could be on the same buffer as the header.
- The hardware repeats the following steps for Data packets that pass successful all criteria described in Section 9.3.2.4.3 till RSP packet is received or till the last data frame of a write exchange or any exception:
  - The FC payload is posted directly to the DDP buffers
  - The Filter context and DDP context parameters are updated according to the packet parameters as described in Section 9.4.2.3.1 and Section 9.4.2.3.2 respectfully.
  - The packet header is used only for the packet processing while it is not posted to the host



- RSP packet Received:
  - Post the ‘RSP’ packet to the LAN queue defined by the FCoE context while the number of received data bytes is reported in the PARAM field in the receive descriptor and the DDP context index in the “MIRR / FCoE” field in the receive descriptor.
  - The hardware Auto-invalidates the FCoE Context enabling further use of the FCoE Context resources and releasing the DDP data buffers by the software
  - Note that even though ‘RSP’ packets are expected only in the initiator the hardware does not check it. Instead, the hardware relates to such packets the same for initiator and responder DDP contexts.
- Exception Errors
  - In case of exception errors, the packets might be posted to the LAN queues and exception errors are reported.
  - In the case of exception errors for packets that match the DDP context, the hardware auto-deactivate the context. Then the software should invalidate the DDP context and then unlock the DDP buffers. Once the context is invalidated the software can reuse the context entry for other exchange.
  - Detailed description of exception errors are given in the following sections and summarized in Section 9.4.3.

9.3.2.1.2 Responder DDP Flow

DDP flow in the target is very similar to the flow in the initiator. Listed below are the differences between the two flows.

- Instead of the read request packet, the target sends the RDY packet.
- RSP packet is not expected to be received in the target. Instead, the DDP flow is completed by reception of the last data packet. The last data packet is the last packet in a sequence (indicated by active F_CTL->End_Sequence flag) with active F_CTL->Sequence_Initiative flag.
  - The packet data is posted to the DDP buffers like all other data packets while the packet header is posted to the LAN receive queue defined by the FCoE context. The receive descriptor of the packet header reports the PARAM field from the DDP context equals to the initial programmed PARAM value plus the total number of received data bytes.
  - The hardware Auto-invalidates the FCoE Context enabling further use of the FCoE Context resources and releasing the DDP data buffers by the software
  - Note that even though a last packet in a sequence with active F_CTL->“Sequence_Initiative” flag are expected only in the responder, the hardware does not check it. Instead, the hardware relates to such packets the same for initiator and responder DDP contexts.

9.3.2.2 DDP Descriptors

All DDP context programming descriptors that are fed to the hardware via the LAN transmit queue are described in the LAN Engine Chapter.

- DDP Context Descriptor (described in section Section)
- DDP Queue Context Descriptor (described in section Section 8.4.2.4.6)
- DDP Filter Context Descriptor (described in section Section 8.4.2.4.7)

On top of the above descriptors there are the DDP buffer descriptors described below in Section 9.3.2.2.1.

9.3.2.2.1 DDP Data Buffers Descriptor

DDP descriptors are 8 byte entities containing only the address of the DDP buffers while its size is defined in the DDP context. The DDP address is assumed to be aligned to the DDP buffer size. Meaning, 512B buffers are aligned to 512B address, 4KB buffers are aligned to 4KB address and so on.

9.3.2.3 FCoE DDP Context Tables

The DDP context is composed of filter context and DMA context. It is programmed by the FCoE context descriptors illustrated in the Figure 9-7 and described in subsections of Section 8.4.2.4. DDP offload is based on a dedicated receive buffer list (queue) per exchange. XL710 supports a Filter table up to 4K filters per function. Note that programming an existing context is prohibited. If the context entry in the table is already there, the software must first invalidate it before reusing it for another exchange.

The Filter context provides the following functionality:

- Identify a match of received packet to a specific exchange
- Check packet validity including expected packet types and in order reception
- Define the matched DDP context index and the LAN queue index. Note that if the indicated LAN queue is not enabled then the DDP functionality of the specific context is not enabled as well.

The DMA context describes the DDP list of buffers (queue) as well as the offload parameters.

9.3.2.4 Packet Validity for DDP Processing

9.3.2.4.1 FCoE Context Match Verification

Packets that pass the "Stateless Receive FCoE Processing" are candidates for DDP processing. Packets that do not match an active FCoE context are posted to LAN queues defined by the Hash filters for FCoE packets. XL710 checks for context match according to global settings described in Section 7.1.9.2.

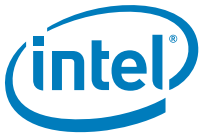
9.3.2.4.2 RSP Packet Verification

RSP Frame is identified by the R_CTL field.

- R_CTL->Information (least significant four bits) equals 0x7 (command status)
- R_CTL->Routing (most significant four bits) equals 0x0 (device data)

Packets identified as RSP packets are subjected to further validity check described below. Packets that do not meet these criteria are posted to the LAN queue defined by the FCoE context with erroneous indication and also auto-deactivate the context. Note that deactivated context directs further matched packets to the LAN queue defined by the context.

- The RSP packet is expected by the context.
  - RSP packets are expected only for FC Initiator defined by active "Initiator" flag in the FCoE context.
  - The previous received frame that matched the context was the last packet in a sequence captured by the "End_Sequence" flag in the FCoE context.



- The received packet composes a whole sequence as expected for control packets (single packet sequence).
  - The SOF flag equals to SOFi2 or SOFi3 depending on FC_Class in the context (defining the first packet in a sequence).
  - The EOF flag equals to EOFt (and F_CTL->"End_Sequence" is set), defining the last packet in a sequence.
  - The received Sequence_ID must be different than the previous Sequence_ID stored in the FCoE context.
  - The Sequence_Count in the received packet could be either the same as the SEQ_CNT in the FCoE context or zero (in order reception check for a first packet in a sequence).
- Additional flags in the Frame Control (F_CTL) match the following values
  - "Exchange_Context" = '1' (packet is received from the responder)
  - "Sequence_Context" = '0'.
  - The "Relative_offset present" = is expected to be cleared (indicating that the Parameter field in the received packet is not used as a Relative_offset).
  - The "First_Sequence" flag is inactive (never expected by active FCoE context)

9.3.2.4.3 Data Packet Verification

FC Data Frame is identified by the R_CTL field.

- R_CTL->Information (least significant four bits) equals 0x1 (solicited data)
- R_CTL->Routing (most significant four bits) equals 0x0 (device data)

Packets identified as Data packets are subjected to further validity check described below. Packets that do not meet these criteria are posted to the LAN queue defined by the FCoE context with erroneous indication and also auto-deactivates the context. Note that deactivated context directs further matched packets to the LAN queue defined by the context.

- EOF and SOF fields
  - The EOF flag equals to either EOFn or EOFt
  - If the previous received packet was the last frame in a sequence the "End_Sequence" flag in the Filter context is set. Hence, it is expected that the current received frame is the first of a sequence. In this case, the received SOF flag should be equal to SOFi2 or SOFi3depending on FC_Class in the context .
  - For non-first packet in a sequence the SOF flag should be equal to SOFn2 or SOFn3 depending on FC_Class in the context .
- In order reception check of the Sequence_ID
  - The Sequence_ID can be any value on the first packet that match the DDP context
  - The Sequence_ID of a first packet in a sequence is expected to be different than the SEQ_ID in the DDP context
  - On any other packet, the Sequence_ID is expected to match the SEQ_ID in the DDP context
- In order reception check of the Sequence_Count
  - The Sequence_Count of a first packet in a sequence, can be either Zero or match the SEQ_CNT in the DDP context
  - On any other packet, the Sequence_Count is expected to match the SEQ_CNT in the DDP context
- Additional flags in the received Frame Control (F_CTL) field
  - "Exchange_Context" flag is expected to be active (1) for packets received at Initiator context and cleared (0) for packets received at Responder context.

  - — "Sequence_Context" = '0'.
  - — If the "Relative_offset present" = is set, the received Parameter field must match the PARAM field in the Filter context.
  - — The "First_Sequence" flag is inactive (never expected by active Filter context)
- Last data packet indicated by active F_CTL->"Sequence_Initiative"
  - — In this case the packet header is posted to the LAN queue and the context is invalidated.

9.3.2.4.4 Other Packets (None Data and None RSP)

Other packets then RSP and Data might be received to an active DDP context. Most likely it could be control packets. Such packets are checked by the DDP context (as described below) for the sake of routing to the LAN queue defined in the DDP context. Failing any of the criteria below is handled as DDP exception that de-activate the context.

- If the previous received packet to this DDP context was the last packet in a sequence, it is expected that the current packet is a first one of a sequence.
- In order reception: The Sequence_ID and Sequence_Count should meet expected values (same criteria as described for the Data Packet)
- Additional flags in the received Frame Control (F_CTL) field
  - — "Exchange_Context" flag is expected to be active for Initiator context and cleared for Responder context.
  - — The "First_Sequence" flag is inactive (never expected by active Filter context)
  - — In read exchange offload in the initiator (indicate by ENODE parameter in the DDP Filter Context Descriptor equals to 0b) the "Abort_Sequence_Condition" field equals 00b (Continue sequence)

9.3.2.4.5 Packets and Headers Indication in the Legacy Receive Queue

As described in the above sections, FCoE packets may be indicated in the LAN receive queues. Listed below are unique status indications for these FCoE packets.

PTYPE — FCoE packets are identified by their Ethernet type.

ARAM — Reflects the value of the PARAM field in the DDP context.

FLTSTAT — FCoE DDP context indication.

FCERR — FCoE Error indication. .

9.3.3 Steering Tag and Processing Hint Support for DDP Traffic (TPH)

See Section 3.1.2.6.2 for information on TLP processing hint support.

The socket ID in the steering tag is inherited from the CPUID parameter in the Tx LAN queue context that is used for the DDP context programming. Note that the CPUID parameter in the DDP context remains static equals to the most updated value of the CPUID in the Tx LAN queue context. The Table 9-3 below describes the other Steering parameters and its enablement for DMA type of DDP.

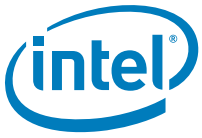


Table 9-3. Steering tag and Processing hint programming by DDP

Traffic Access	TPH enablement	PH value
DDP descriptors fetch	TPHRDESC in the DDP Queue Context Descriptor	Desc_PH in GLTPH_CTRL register
Write data to the DDP buffers	TPHDATA in the DDP Queue Context Descriptor	DATA_PH in GLTPH_CTRL register

9.4 Common Transmit and Receive Logic

9.4.1 DDP Context Management

This section describes the DDP contexts management. Note that TSO context is part of the LAN Tx queue context from which the TSO is transmitted. XL710 supports a table for DDP containing up to 4K contexts per PF. The Context table is stored in a "private" memory in the PF space. DDP context are fetched to an internal cache as required. The DDP contexts are enabled per VSI by the FCOE_ENA flag in the VSIQF_CTL registers. The number of FCoE filter's buckets as well as the number of FCoE DMA contexts (DDP) is set for the PF by the FCHSIZE; FCDSIZE parameters in the PFQF_CTL_0 register.

9.4.1.1 FCoE Context Programming

Context programming sequence is described for TSO in Section 9.2.2 and DDP in Section 9.3.2.1. The context descriptors used for programming are described in Section 8.4.2.4. During programming time the software defines the DDP context index to be used. DDP offloads have also matched filter contexts associated with the same exchange. The filter index is chosen internally by the hardware as described in Section 9.3.2.4.1. The filter context holds the index of the DDP context that matches the specific exchange as well as the LAN receive queue that is used for the exchange. See Figure 9-8 below that describes these index pointers. As indicated above, the index of the DDP contexts is defined by software. Therefore, it is the software responsibility to keep track of used contexts and avoid stepping over active context.

Following a successful context programming the hardware increments the PFFCOE_DDPCNT

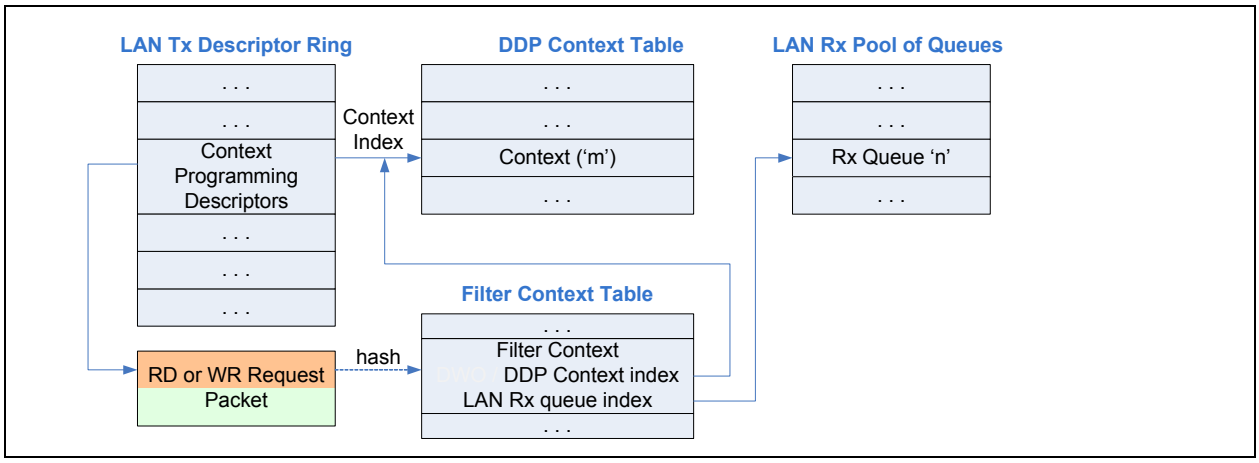


Figure 9-8. FCoE context table mapping

9.4.1.2 FCoE Context Invalidation

In some erroneous cases software might need to invalidate a context before the exchange is completed (for example a time out event). The software can invalidate an FCoE context from a LAN transmit queue by context invalidation descriptor (FCoE Transmit context descriptor with OPCODE field equals to "DDP Context invalidation"). The FCoE transmit context descriptor as well as the resulted receive descriptor are detailed in the "FCoE Context Invalidation" paragraph in Section 8.4.2.4. The context invalidation structures are illustrated in the Figure 9-9 below. The software may release the DDP buffers only after the context invalidation status descriptor is reported by the hardware.

Following a successful context invalidation the hardware decrements the PFFCOE_DDPCNT

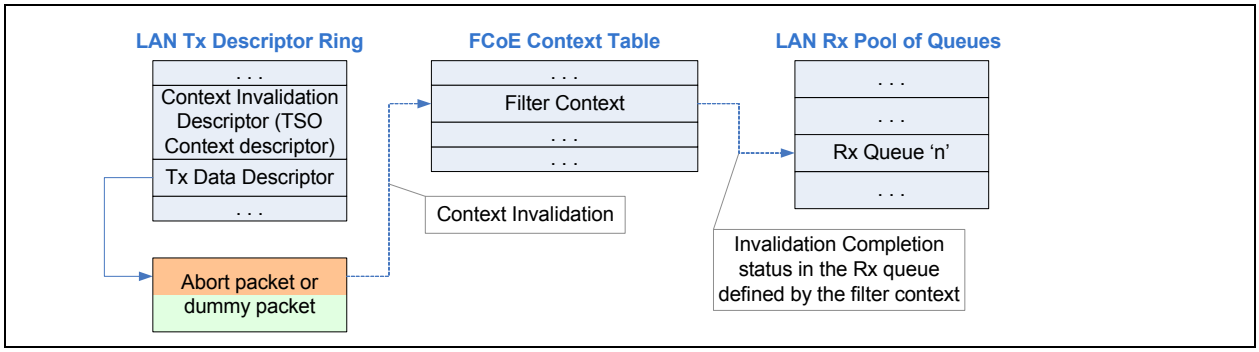
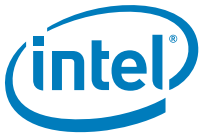


Figure 9-9. Context Invalidation Diagram

9.4.2 FCoE Initialization Flow

This section describe the required initialization flow done by the PF software enabling FCoE offload.



- Follow the steps below only if FCoE offload is enabled for the device by the FCOEN flag in the GLGEN_STAT register. If it is not enabled, the software should not attempt to activate any of the FCoE offloads.
- Define the number of FCoE contexts and the number of hash buckets for the PF by setting the FCDSIZE and FCHSIZE parameters in the PFQF_CTL_0 register. Note that the software is expected to set all the fields in this register in a single cycle as part of the driver initialization flow since setting this register also initializes internal DDP.
- Allocate pages in the FPM for the FCoE DMA and filter contexts and indicate them to the hardware by the FPM Object Registers described in Section 7.9.2. The PF software is expected to initialize (set to zero) the whole space in the FPM assigned for the FCoE filter table for the function as part of the driver initialization flow. The initialized FPM provides an indication to the hardware that the table is empty.
- FCoE MAC address filter should be defined for the function and FCoE VLAN should be defined for the VSI
- Jumbo packets should be enabled for the LAN port by the "Max Frame Size" parameter in the "Set MAC Config" Admin Command. The global GLFCOE_RCTL should be defined as well indicating the Maximum received FC payload size (loaded from the NVM).
- Assign transmit and receive LAN queues for the FCoE traffic
  - Set the FCENA flag in these queues at queue contexts programming
  - Enable Jumbo packet reception per each queue in the
  - If multiple LAN receive queues are assigned then enable the FCoE hash filter for the function in the PFQF_HENA.
- If DDP offloads are required then enable FCoE for the VSI (setting the FCOE_ENA flag in the VSIQF_CTL register) as part of the "create VSI" procedure
- Define the traffic class and user priority as "No Drop" (enable PFC)
  - Set the matched bits in the TC2PFC field in the following registers: PRTDCB_TLPMC; PRTDCB_TUPMC and PRTDCB_MFLCN.
  - Set the matched bits in the TC2PFC field in the PRTDCB_TC2PFC
  - Note that for any TC that uses the PFC, the PFCTIMER field in the GLDCB_RSPMC should be set. See Section 7.7.1.2.8 for more details on this timer.

9.4.3 Exception Handling

The Table 9-4 below summarizes the groups of FCoE exceptions including its impact and the reported status to the software. Unless specified differently, in all cases of receive exceptions the packet are reported to LAN receive queue based on the exchange ID with FCoE packet type indication. The following subsections provide a detailed list of these exceptions. In most of the exceptions the FCoE context is deactivated (as indicated in the table below). Some rules for exceptions handling:

- Following exceptions that deactivate the DDP context (either the DMA or the filter contexts), the software can unlock the DDP buffers only after invalidating the DDP context. Only then the software could initiate a new exchange with the same exchange ID and re-use the same DDPINDX for a new DDP context.
- Exceptions detected by the filter are reported immediately. Exceptions detected by the DMA engine like ECC might be delayed by several packets. In those cases any packet on the same context will carry this error.
- The errors are reported in the receive descriptor in a priority order. If a packet encounters multiple errors, only the first detected error is reported.

Table 9-4. FCoE Exceptions Table

Exception Group	Section Number	Descriptor Parameters	
		FLTSTAT	FCE (FC Error)
Stateless Receive	Section 9.4.3.1	No Context Match	FCoE protocol Error
Filter Context "Abnormal" Exceptions. The filter context remains active.	Section 9.4.3.2	FCoE context match with no action	Good packet (no error)
Filter Context "Error" Exceptions. The filter context is deactivated (note 1).	Section 9.4.3.3	FCoE context deactivated	FCoE Filter Context Error
DMA Context Exceptions. The DMA context is deactivated (note 1).	Section 9.4.3.4	FCoE context deactivated	FCoE DMA Context Error
DMA Context Warnings. The warning has no impact on the context deactivation.	Section 9.4.3.5	FCoE context match with action (either 01b or 11b)	FCoE DMA Context Warning
Note 1: Following exceptions that deactivate a DDP context, the software must invalidate the context before reusing the same DDPINDX for a new exchange.			

9.4.3.1 Exceptions related to Stateless Receive

Listed below are exceptions related to stateless receive processing:

- Receive packet with unsupported FCoE version. Packets with unsupported FCoE version are not candidates for any FCoE offload (both state-less and state-full). The PTYPE reported in the receive descriptor is unaffected by this criteria.
- Receive packet is not whole number of 4 bytes. Packets that are not whole number of 4 bytes are posted to host only if the SAVBAD flag is set in the GLFCOE_RCTL register.
- Receive Data packet with payload is larger than 2KB
- Receive packet has FC CRC error. Packets with FC CRC error are posted to host only if the SAVBAD flag is set in the GLFCOE_RCTL register.
- The SOF flag does not match FC Class 2 or FC class 3.
- The received F_CTL->"End_Sequence" flag is not set while the EOF tag is EOFt or the F_CTL->"End_Sequence" flag is set while the EOF tag is not EOFt.
- Any of following obsolete flags in the F_CTL is found active: bit 9 (was "Retransmitted Sequence"); bit 8 and bits 7-6 (were "Continue Sequence Condition")
- Packets with extended FC headers other than VFT header and EOF equals to EOFt are reported with a stateless exception. Note that when EOF equals to EOFt the device expects for End_Sequence flag, which is not present in the field vector).
- The packet includes Optional FC header(s).

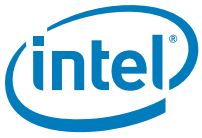
9.4.3.2 FCoE Filter Context "Abnormal" Exceptions

Listed below are "Abnormal" exceptions related to the Filter context:

- Received packet is not RSP nor Data in the case of DDP context

9.4.3.3 FCoE Filter Context "Error" Exceptions

Listed below are "Error" exceptions related to the Filter context:



- The SOF flag in the received packet does not match to the "FC Class" flag in the context.
- The F_CTL field in the received packet differs than the expected value (see Section 9.3.2.4.2 for expected value in RSP packet. see Section 9.3.2.4.3 for expected value in Data packet)
- Out of order reception
  - The first packet received to the context do not have the SOFi2 flag nor SOFi3 flag.
  - Receive packet with SOFi2 tag or SOFi3 tag is received while the previous "good" packet received to this context did not have the EOFt tag (and the F_CTL->"End_Sequence" was set as well).
  - Receive packet with SOF tag is not SOFi2 or SOFi3 while the previous "good" packet received to this context had the EOFt tag (and the F_CTL->"End_Sequence" was set as well).
  - Receive packet with SOFi2 tag or SOFi3 tag do not have a different Sequence_ID than the value in the previous "good" packet received to this context.
  - Receive packet with SOF tag other than SOFi2 or SOFi3 do not match the Sequence_ID in the filter context (as received from the previous "good" packet to this context).
  - The Sequence_Count in the received packet differs than the expected value.

Listed below are "Error" exceptions related to a DDP Filter context:

- Receive RSP packet to a "Target" context is not expected and considered as exception Error.

9.4.3.4 FCoE DMA Context Exceptions

9.4.3.4.1 Receive DMA Exceptions

Listed below are exceptions related to the DMA context:

- Receive Data packet that requires more space than the DDP buffers
- Detected ECC errors in the receive packet buffer

9.4.3.4.2 Transmit DMA Exceptions

Transmit segmentation (or ETSO) is ended when the total byte count of the TSO is exhausted or the transmit buffer with EOP flag is exhausted whichever comes first. During nominal operation it is expected that the total byte count and the transmit buffers exhaust together. Any mismatch is an exception detailed below.

- The transmit buffers are exhausted before the total byte count. In this case the exhausted transmit buffer might abort the whole transmit queue.

9.4.3.5 FCoE DMA Context Warnings

9.4.3.5.1 Receive DMA Warnings

Listed below are warnings related to receive DMA context:

- This is a place holder for DMA warnings.

9.4.3.5.2 Transmit DMA Warnings

This subsection is intentionally blank (reserved as a space holder for future use).

9.4.3.6 Other Exceptions Handling

The Table 9-5 below summarizes additional exceptions that may be related to transmission, context programming and packet reception as follow.

Table 9-5. Other Exceptions Table

Exception Num	Exception	Response
1	Setting the LAN queue index in the DDP context to a non valid queue	Context programming is accepted. At packet reception the packet is dropped and count by the GLVREPC counter of the VSI
2	Setting the DDP context index in the filter context to an existing DDP context	Programming is accepted. Received packets are processed by a shared DMA context till its buffers are exhausted. The DDP buffers in this case contain mixed data from multiple exchanges.
3	Setting the DDP context index in the DMA context descriptor to an existing context	Programming is accepted stepping over the previous context
4	Setting the DDP context index in the filter and the DMA contexts to different values	The index in the DDP context descriptor is used to program the context. The index in the filter context descriptor is used for the filtering purpose. The programming is accepted. Received packets will be directed to a DDP context by the index in the filter context.
5	Setting the DDP context outside the range of the function	Programming is rejected (1)
6	Not enough space in the data buffers	Packet is directed to the LAN queue
7	Last buffer size > data buffer size (LSIZE > BSIZE)	Programming is accepted. During reception it is handled the same as exception #6 if there is not enough space for the received packet.
8	The offset in the first buffer > data buffer size (FOFF > BSIZE)	Programming is accepted. During reception it is handled the same as exception #6. The packets are directed to the LAN queue.
9	In a DDP with a single buffer the offset in the first buffer > Last buffer size (FOFF > LSIZE)	Programming is accepted. During reception it is handled the same as exception #6. The packets are directed to the LAN queue.
10	Program an FCoE context to an existing FCoE filter	Programming is rejected (1)
11	Program a context even though the FCOE_ENA flag in the VSIQF_CTL register is inactive	Programming is rejected (1)
Note 1: In all cases that a context programming is rejected the packet used for the context programming is transmitted OK.		

10.0 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 XL710 in each platform configuration.

10.1 Pass-through (PT) functionality

PT is the term used when referring to the process of sending and receiving Ethernet traffic over the sideband interface. The XL710 has the ability to route Ethernet traffic to the host operating system as well as the ability to send Ethernet traffic over the sideband interface to an external MC. See Figure 10-1.

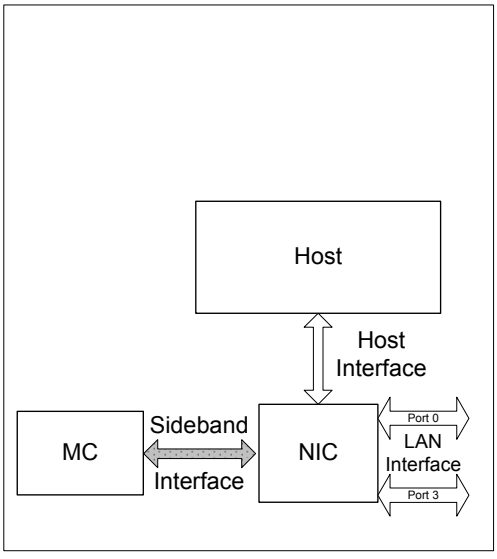


Figure 10-1. Sideband interface

The sideband interface provides a mechanism by which the XL710 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 XL710 supports three sideband interfaces:

- SMBus (legacy or as part of MCTP)
- NC-SI
- PCIe (together with MCTP) - when the system is up.

Note: The usable bandwidth for either direction is up to 1 Mb/s when using SMBus and 100 Mb/s for the NC-SI interface. When working over PCIe, the bandwidth is limited only by the PCIe bandwidth and the the XL710 processing capabilities and can sustain any network bandwidth. The XL710 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 S-tag, or VLAN tags. For traffic from the MC to the LAN the maximum supported packet size is 1536 bytes including all tags.

Note: In MCTP mode, the PCIe and SMBus interface can receive MCTP commands in parallel. For example, the MCTP enumeration process can be done both over SMBus and over PCIe. However, only one of the interfaces can receive NC-SI commands or pass-through traffic.

10.1.1 Supported topologies

The XL710 can support up to two channels to management controllers in some topologies. The following connections are available:

- Connection via legacy SMBus (See Section 10.5).
- Connection via NC-SI over RBT (RMII) (See Section 10.6)
- Connection via NC-SI over MCTP. This connection can be over SMBus, PCIe or both. This connection can be used either for pass through or for control only (See Section 10.7)
- Two connections — A connection via NC-SI over RBT for pass-through traffic and a connection via MCTP used only for control traffic (See Section 10.7)

The channel used is for pass through is defined in the *Redirection Sideband Interface* field in the *Common Manageability Parameters* NVM word (see Section 7.2.31.2) and is common to all the ports in the device.

10.1.2 PT packet routing

When an Ethernet packet reaches the XL710, it is examined and compared to a number of configurable filters. These filters are configurable by the MC and include, but not limited to, filtering on:

- MAC address
- IP address
- UDP/IP ports
- VLAN tags
- Ethertype

If the incoming packet matches any of the configured filters, it is passed to the MC. Otherwise, it is not passed.

The packet filtering process is described in Section 10.3.

10.2 Components of the sideband interface

There are two components to a sideband interface:

- Physical layer
- Logical layer

10.2.1 Physical layer

This is the electrical connection between the XL710 and the MC.

10.2.1.1 SMBus

The SMBus physical layer is defined by the SMBus specification. The interface is made up of two connections: data and clock. There is also an optional third connection: the alert line. This line is used by the XL710 to notify the MC that there is data available for reading in legacy SMBus mode. Refer to the SMBus specification for details.

The SMBus can run at three speeds: 100 KHz (standard SMBus), or 400 KHz (I^2C fast mode) or 1 MHz (I^2C fast mode plus).The speed used is selected by the *SMBus Connection Speed* field in the SMBus Notification Timeout and Flags NVM word.

10.2.1.1.1 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 and legacy SMBus transactions, a PEC is not expected.

The following table lists the behavior of the XL710 in each PEC configured mode for transactions directly handled by hardware after receiving packets with or without PEC.

Table 10-1. SMBus PEC modes[1]

		Target PEC Mode	
SMBus transaction (relative to the XL710)	**XL710 PEC Mode**	**PEC Enabled**	**PEC Disabled**
Master Write[2]	Enabled	(A) Target ACKs the PEC byte	(A) Target NACKs the PEC byte
Master Write[2]	Disabled	(A) Target receives stop before expected PEC byte	(A) PEC byte is not expected
Slave Write[3]	Enabled	(A) Target ACKs last data byte; PEC byte is NACKed	(A) Target NACKs last data byte; No PEC byte is written by the slave

Table 10-1. SMBus PEC modes[1]

		Target PEC Mode	
SMBus transaction (relative to the XL710)	**XL710 PEC Mode**	**PEC Enabled**	**PEC Disabled**
Slave Write[3]	Disabled	(A) Target ACKs last data byte; PEC byte is 0xFF	(A) Target NACKs last data byte and generates stop afterwards
Slave Read[4]	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
Slave Read[4]	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 Legacy SMBus writes commands (direct receive) and in MCTP over SMBus (transmitted transactions.
3. Used in Legacy SMBus Read commands.
4. Used in Legacy SMBus mode (alert/async-notify) and 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 XL710.

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 be set for MCTP mode and should be cleared in legacy SMBus mode.

10.2.1.2 NC-SI

The XL710 uses the DMTF standard sideband interface. 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 physical layer of NC-SI is very similar to the RMII interface, although not an exact duplicate. Refer to the NC-SI specification for details of the differences.

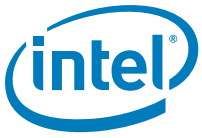
10.2.1.3 PCIe Vendor Defined Messages (VDMs)

The XL710 uses VDMs over PCIe defined in the DMTF MCTP specification to convey pass-through traffic or NC-SI control traffic. See Section 3.1 for details of the PCIe interface.

10.2.2 Logical layer

10.2.2.1 Legacy SMBus

The protocol layer for SMBus consists of commands the MC issues to configure filtering for XL710 management traffic and the reading and writing of Ethernet frames over the SMBus interface. There is no industry standard protocol for sideband traffic over SMBus. The protocol layer for SMBus on the XL710 is Intel proprietary. The legacy SMBus protocol is described in Section 10.5.



10.2.2.2 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 XL710 are discussed in Section 10.6.4. For information on base NC-SI commands, see the NC-SI specification.

NC-SI traffic can run on top of three different Physical layers:

1. NC-SI physical layer as described in Section 10.2.1.2.
2. MCTP over PCIe VDM.This protocol enables control and pass-through traffic over PCIe of a NIC or a LOM device. The NC-SI over MCTP protocol is slightly different than the standard NC-SI as it includes additional NC-SI commands. This mode is usually paired with an MCTP over SMBus, where this mode is used in S0 states and the SMBus interface is used in Sx state.The MCTP protocol and the differences from standard NC-SI is described in Section 10.7.
3. MCTP over SMBus. As previously described, this layer is paired with the MCTP over PCIe to support Sx modes.

The XL710 exposes one NC-SI package with four channels, one per port. The XL710 implements a type C NC-SI interface (single package, common bus buffers and shared RX queue) as described in section 5.2 of the NC-SI specification.

10.2.2.2.1 Package ID setting

The package ID can be set either from the NVM *Package ID* field in the NC-SI Configuration 1 NVM word (Section 7.2.34.4) 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 7.2.34.5). Note that when the package ID is set from the SDP pins, the used SDP should be set as an input in the relevant GLGEN_GPIO_CTL register. The SDP to use is defined in the *PackageID SDP* field in the NC-SI Configuration 2 NVM word (Section 7.2.34.5).

10.2.2.2.2 Channel ID mapping

The mapping of the channels to physical ports is according to the NC-SI Channel to Port Mapping NVM word (Section 7.2.34.14) if the NC-SI Channel to Port Mapping Table *Valid* bit is set. If this bit is not set, the following algorithm should be used:

```
Channel_ID = 0

NC-SI_channel[3 :0] = -1 // ports not associated to channels yet.

For func = 0 to 15 { // 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 ) {

        // Port is valid and port is not already associated to a channel

        NC-SI_channel[Port_ID] = Channel_ID; // assign channel

        Channel_ID++; // go to next channel

        }
```

```
}
```

This algorithm maps channel numbers that match the order of the 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.

10.3 Packet filtering

Since both the host operating system and an MC use the XL710 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.

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 etc).

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.

10.3.1 Manageability receive filtering

This section describes the manageability receive packet filtering flow. Packet reception by the XL710 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:

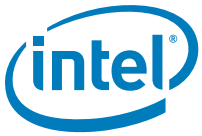


Table 10-2. Filters accessible to MC

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]	Four unicast MAC manageability addresses	LAN_PWR_GOOD
VLAN Filters [7:0]	Eight VLAN tag values	LAN_PWR_GOOD
UDP/TCP Port Filters [15:0]	16 destination port values	LAN_PWR_GOOD
Flexible 128 bytes TCO Filter	Length and values for one flex TCO filter	LAN_PWR_GOOD
IPv4 and IPv6 Address Filters [3:0]	IP address for manageability filtering	LAN_PWR_GOOD
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 both the NC-SI and legacy SMBus interfaces.In NC-SI modes, part of the filters are programmed via standard NC-SI commands and part of the filter are programmed via the Intel OEM commands described in Section 10.6.4. In legacy SMBus, the filtering is programmed either from NVM or via the commands described in Section 10.5.11.1.

All filters are reset only on internal power on reset. Register filters that enable filters or functionality are also reset by firmware reset in NC-SI mode. These registers can be loaded from the NVM following a reset in SMBus mode. See Section 7.2.32 for description 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. 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 steps 2 and 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, etc.) are not forwarded to the MC.
- Packets longer than 2 KB are filtered out.

The following sections describe the manageability filtering, followed by the final filtering rules.

The filtering rules are created by programming the decision filters as described in Section 10.3.4.

10.3.2 L2 filters

10.3.2.1 MAC and VLAN filters

The manageability MAC filters allow a comparison of the destination MAC address to one of 4 filters defined in the *PRT_MNG_MMAH* and *PRT_MNG_MMAL* registers.

The VLAN filters allow a comparison of the 12 bit inner VLAN tag to one of 8 filters defined in the *PRT_MNG_MAVTV* registers.

10.3.2.2 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 4 filters); the result is incorporated into decision filters.

Each manageability EtherType filter can be configured as pass (positive) or reject (negative) using a polarity bit. In order 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 XL710 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 XL710. 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 shouldn't configured with IPv4 or IPv6 Ethertype values.

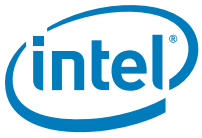
10.3.3 L3/L4 filtering

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.

10.3.3.1 ARP filtering

ARP filtering — The XL710 supports filtering of ARP request packets (initiated externally) and ARP responses (to requests initiated by the MC).

In legacy SMBus mode, the ARP filters can be used as part of the ARP offload described in Section 10.5.4. ARP offload is not specifically available when using NC-SI. However, the general filtering mechanism is used to filter incoming ARP traffic as requested using the Enable Broadcast Filtering NC-SI command.



In order 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, then ARP packets matching the IP address pass the filter. If these packets should be dropped, then an OR Ethertype filter with a a value of 0x0800 (IPv4) should be added.

See Section 7.3.2.6 for the format of ARP packets.

10.3.3.2 Neighbor discovery filtering and MLD

The XL710 supports filtering of the following ICMPv6 packets.

Neighbor discovery packets:

1. 0x86 (134d) - Router Advertisement.
2. 0x87 (135d) - Neighbor Solicitation.
3. 0x88 (136d) - Neighbor Advertisement.
4. 0x89 (137d) - Redirect.

MLD packets:

1. 0x82 (130d) - MLD Query
2. 0x83 (131d) - MLDv1 Report
3. 0x84 (132d) - MLD Done
4. 0x8F (143d) - MLDv2 Report

The neighbor discovery packets has dedicated enables for each type in the decision filters. For MLD, a single enable controls the forwarding of all the MLD packets. This means that either all the MLD packets types are selected for reception or none of them.

See Section 7.3.2.5 for the format of ICMPv6 packets.

10.3.3.3 RMCP filtering

The XL710 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 *PRT_MNG_MSFM.PORT_26F/298_UDP/TCP* fields.

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.

10.3.3.4 ICMP filtering

The XL710 supports filtering by ICMPv4. This filter matches if the IP *Protocol* field equals to 1b.

See Section 7.3.2.4 for the format of ICMP packets.

10.3.3.5 Flexible port filtering

The XL710 implements 16 flex destination port filters. The XL710 directs packets whose L4 destination 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.

10.3.3.6 IP address filtering

The XL710 supports filtering by destination IP address using IPv4 and IPv6 address filters. These are dedicated to manageability. The XL710 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 LSBits should be compared to the 24 LSBits 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.

10.3.3.7 Checksum filtering

The XL710 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.

Enabling the XSUM filter when using the SMBus interface is accomplished by setting the *Enable XSUM Filtering to Manageability* bit. This is done using the Update Management Receive Filter Parameters command. See Section 10.5.11.1.6.

To enable the XSUM filtering when using NC-SI, use the Enable Checksum Offloading command. See Section 10.6.4.13.

10.3.4 Flexible 128-byte filter

The XL710 provides one flex TCO filter. This filter looks for a pattern match within the first 128 bytes of the packet.

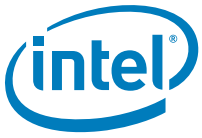
From the first 128 bytes, some of the fields are skipped for this comparison. These are field that are not exposed to the MC. The tags skipped are:

- S-tag

The flex filter programming should ignore the presence of these fields.

Note: The flex filter comparison should be disabled in the MDEF registers while the flex filter is being updated.

Note: The flexible filter is not applied to transmit packets and a transmit packet is considered as if it didn’t pass the filter.



10.3.4.1 Flexible filter structure

The filter is composed of the following fields:

1. Flexible filter length — This field indicates the number of bytes in the packet header that should be inspected. The field also indicates the minimal length of packets inspected by the filter. Packet below that length is not inspected. Valid values for this field are: 8*n, where n=1…16.
2. Data — This is a set of up to 128 bytes comprised of values that header bytes of packets are tested against.
3. Mask — This is a set of 128 bits corresponding to the 128 data bytes that indicate for each corresponding byte if is tested against its corresponding byte. The general filter is 128 bytes that the MC configures; all of these bytes might not be needed or used for the filtering, so the mask is used to indicate which of the 128 bytes are used for the filter.

Each filter tests the first 128 bytes (or less) of a packet, where not all bytes must necessarily be tested.

10.3.4.2 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 need to 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.

10.3.4.2.1 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 pointed to by global offset 0x08 of the manageability module header section.

10.3.4.2.1.1 Section header — offset 0x0

Bits	Name	Default	Description	Reserved
15:0	Block Length	0xC	Section length in words (including CRC word and length word).	

10.3.4.2.1.2 Flexible filter length and control — offset 0x01

Bits	Name	Default	Description	Reserved
15:8	Flexible Filter Length (bytes)			
7:5	Reserved		Reserved.	
4	Last Filter			
3	Apply Filter to LAN 3			
2	Apply Filter to LAN 2			
1	Apply Filter to LAN 1			
0	Apply Filter to LAN 0			

10.3.4.2.1.3 Flexible filter enable mask — offset 0x02 – 0x09

Bits	Name	Default	Description	Reserved
15:0	Flexible Filter Enable Mask			

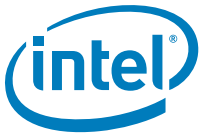
10.3.4.2.1.4 Flexible filter data — offset 0x0A – 0x49

Bits	Name	Default	Description	Reserved
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.

10.3.4.2.1.5 Section footer — offset block length

Bits	Name	Default	Description	Reserved
15:8	CRC 8		CRC8 of the previous section.	
7:0	Reserved		Reserved.	



10.3.5 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.

10.3.5.1 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 10-2. 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), then 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, then 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, then at least one of the enabled OR filters must match for the decision filter to match.

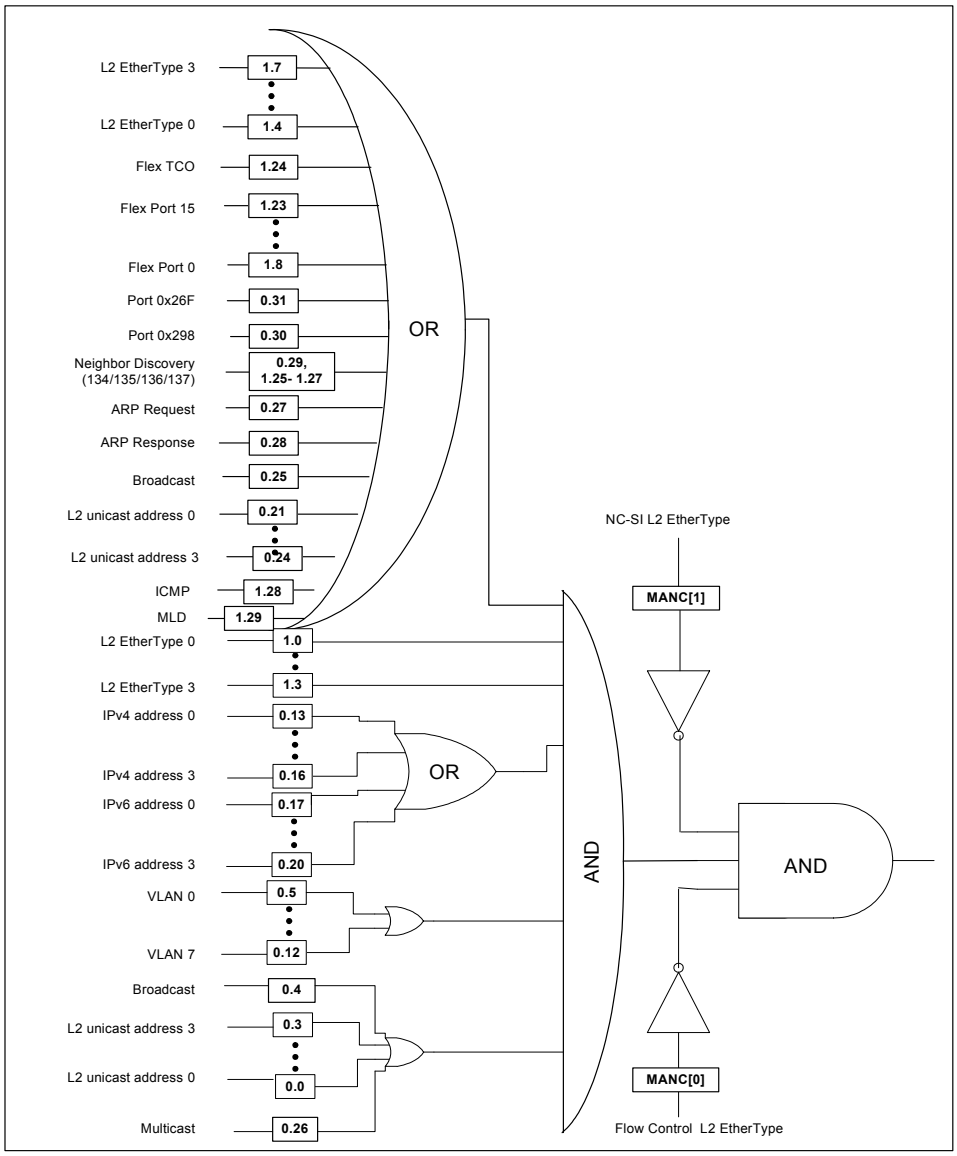
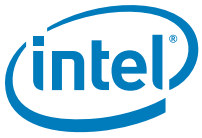


Figure 10-2. Manageability decision filters

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 11.2.2.11.5 and Section 11.2.2.11.4. 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.

10.3.5.2 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 10-3 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 10-3. Exclusive traffic behavior

	Filter Match		Filter Doesn't Match
traffic Source	**PRT_MNG_MNGONLY = 0**	**PRT_MNG_MNGONLY = 1**	**N/A**
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 10.3.5.1) can be used as filters to pass traffic to the host.

Table 10-4. 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.

When using the SMBus interface, the MC enables these filters by issuing the Update Management Receive Filter Parameters command (see Section 10.5.11.1.6) with the parameter of 0x0F.

The *PRT_MNG_MNGONLY* is also configurable when using NC-SI using the Set Intel Filters — Manageability Only Command (see Section 10.6.4.5.3).

All manageability filters are controlled by the MC only and not by the LAN software device driver.

10.3.5.3 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— In order 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 relates to the inner VLAN.

Table 10-5 lists the relationship between the previously mentioned bits and the forwarding decisions:

Table 10-5. 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 a host (and optionally to LAN).	Packet is sent to the host (and optionally to LAN).	Packet is sent to the host (and optionally to LAN).	Packet is sent to the host (and optionally to LAN).

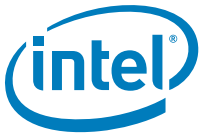


Table 10-5. PRT_MNG_MANC bits impact

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 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 LAN).	Packet is sent to the host (and optionally to LAN).	Packet is sent to the LAN.	Packet is sent to the host (and optionally to 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).

10.3.6 Filtering programming interfaces

The XL710 provides multiple options to program the forwarding filters, depending on the interface used and the level of flexibility needed. The Table 10-6 lists the different options and points to the description of the relevant commands.

Table 10-6. 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 10.6.2.1. 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 10.3.6.1 and in Section 10.6.4.14 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 10.6.4.14.12.1)
	Flexible	This interface described in most of the sub-sections of Section 10.6.4. 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.
SMBus	Abstract	The Set Common filter command (Section 10.5.11.1.7) can be used to set the most common filters. When using this commands the flexible filtering interface should not be used. When sending this command, all previous filtering requests are cleared.
	Flexible	The Update MNG RCV Filter Parameters (Section 10.5.11.1.6) can be used to define the exact filtering rules to be applied.

10.3.6.1 Shared MAC and shared IP support

The XL710 operates in systems where the same MAC and IP are shared between a platform's host operating system and its MC. In order to support such systems, the XL710 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.

10.3.6.1.1 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's possible to bind multiple IP addresses to a single MAC address, the XL710 needs to know the IP address shared by an MC in order to deliver packets to it. An MC uses the Set IP Address command to communicate its IP address to the XL710. The Set IP Address command is defined in Section 10.6.4.14.1.

In order to notify the XL710 that the MC intends to use a shared MAC, the Set Shared Mode command (Section 10.6.4.14.12.1) 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 10.6.4.14).

10.3.6.1.1.1 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 is required. For example, port-based filtering would be used to route WS-management packets to an MC.

The XL710 needs to know the ports owned by an MC in order to deliver packets to it. An MC uses the Set Port command to communicate its ports to the XL710. The Set Port command is defined in Section 10.6.4.14.3. The XL710 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 10.6.4.14.10 for more details).

10.3.6.1.1.2 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.

10.3.6.1.1.3 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 XL710 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.

10.3.7 Possible configurations

This section describes ways of using management filters. Actual usage might vary.

10.3.7.1 Dedicated MAC packet filtering

- Select one of the eight rules for dedicated MAC filtering.
- 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.
- 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.

10.3.7.2 Broadcast packet filtering

- Select one of the eight rules for broadcast filtering.
- Set bit 25 in the MDEF register of the decision rule to enforce broadcast filtering.
- Set other bits to qualify which broadcast packets are allowed to pass through. For example:
  - Set bit 5 in 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.

10.3.7.3 VLAN packet filtering

- Select one of the eight rules for VLAN filtering.
- Set bit 5 to 12 in the MDEF register to qualify with the relevant manageability VLANs.
- Set other bits to qualify which VLAN packets are allowed to pass through. For example:
  - Set any L3/L4 bits (bits 27 to 31 in the MDEF register and bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

10.3.7.4 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 XL710 supports IPv6 neighbor discovery protocol. Since the protocol relies on multicast packets, the XL710 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.

10.3.7.5 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:

- Select one of the eight rules.
- Set a manageability dedicated MAC filter to the host MAC address and set the matching bit (0-3) in the MDEF register.
- If VLAN is used for management, load one or more management VLAN filters and set the matching bit (5- 12) in the MDEF register.

ARP filter/neighbor discovery filter is enabled when 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.

10.3.8 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.

10.4 OS-to-BMC traffic

10.4.1 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 XL710 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 XL710 examines these packets and it 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 XL710 checks if it should be forwarded to the network, the host, or both. Figure 10-3 describes the flow for OS-to-BMC traffic for the NC-SI over RBT case. OS2BMC is also available when operating over MCTP. It is not available in legacy SMBus mode.

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.

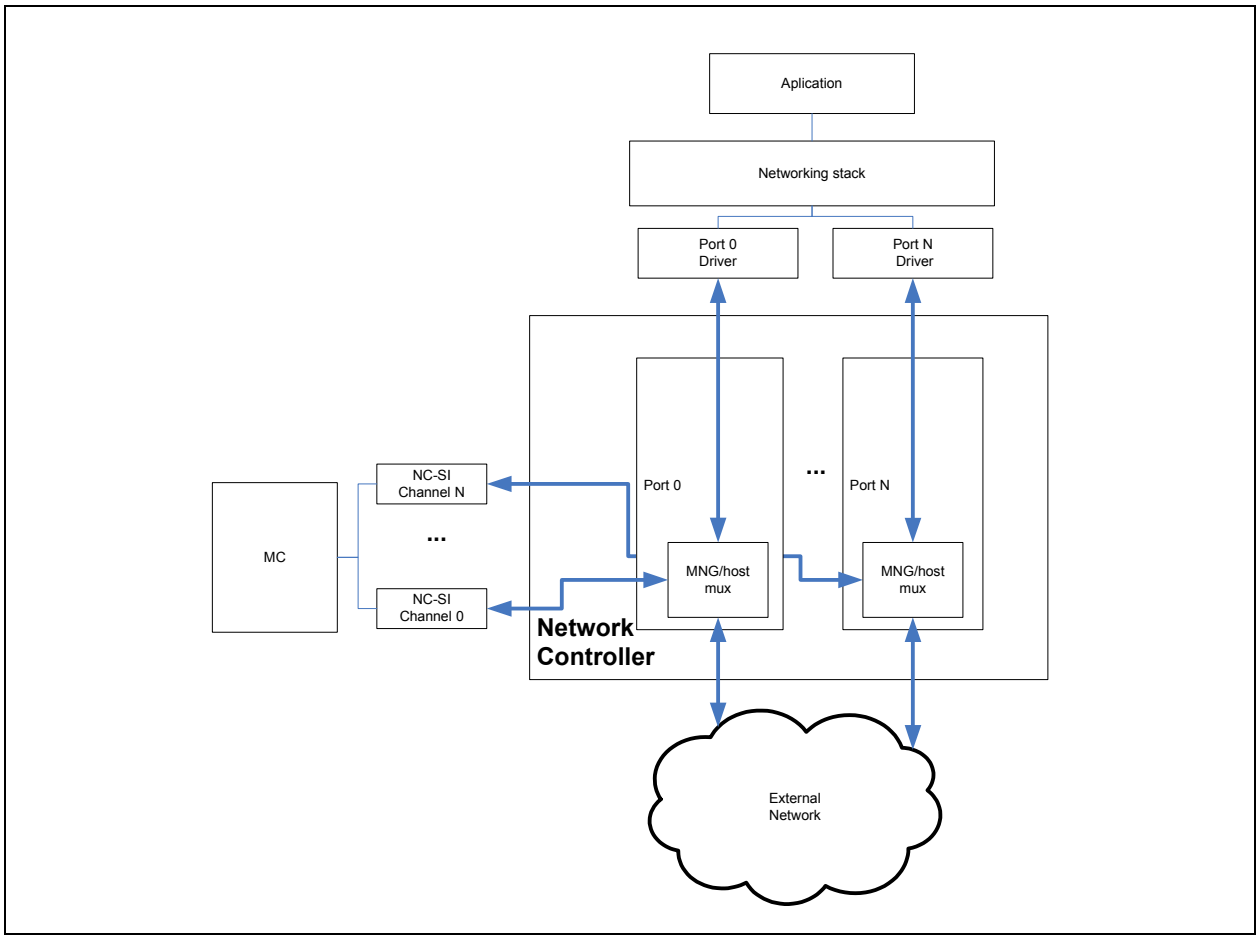


Figure 10-3. OS-to-BMC flow

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).

10.4.2 Filtering

10.4.2.1 OS-to-BMC filtering

The flow used to filter packets from the MC to the host is described in Section 7.4.4.7.2.

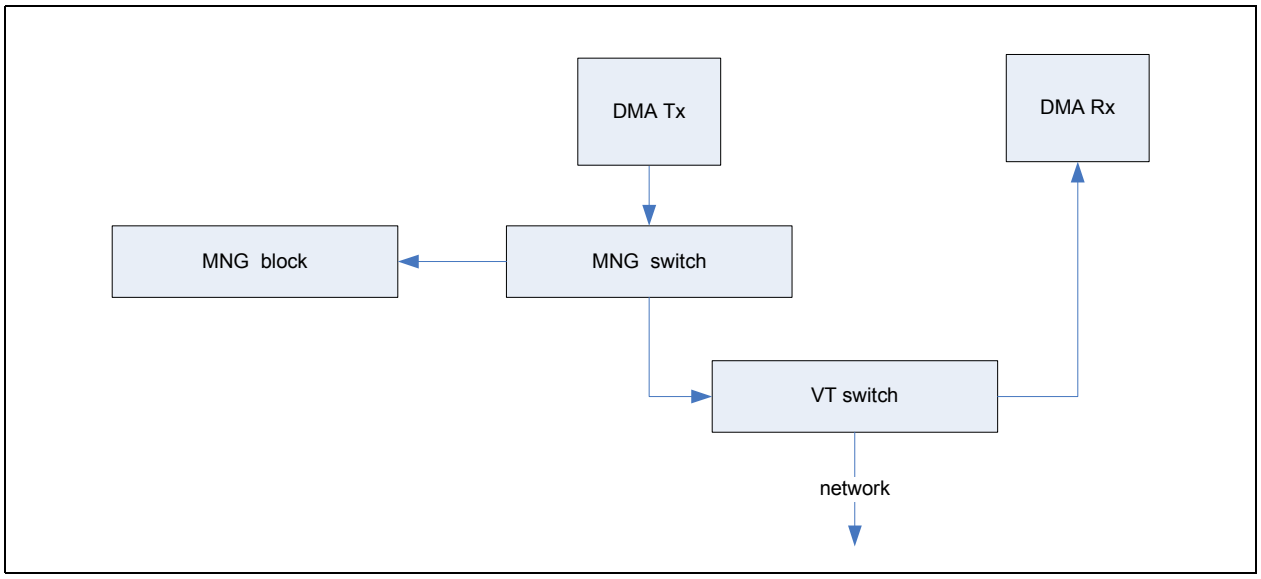


Figure 10-4. OS-to-BMC and VM-to-VM filtering

10.4.2.2 BMC-to-OS filtering

The flow used to filter packets from the host to the BMC is described in Section 7.4.4.7.3.

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.

10.4.3 Blocking network-to-BMC flow

In some systems the MC might have its own private connection to the network and might use a XL710 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.

10.4.4 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.

10.4.5 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, then 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, then they are counted both as received packets and as packet transmitted to the network.

See Section 7.11.5 for details of the statistics hierarchy.

10.4.6 OS-to-BMC enablement

The XL710 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, then the OS-to-BMC communication is stopped.

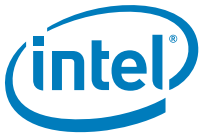
In order to enable OS-to-BMC either:

- 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.

10.5 SMBus PT interface

SMBus is the system management bus defined by Intel. It is used in personal computers and servers for low-speed system management communications. This section describes how the SMBus interface operates in legacy PT mode.



10.5.1 General

The SMBus sideband interface includes standard SMBus commands used for assigning a slave address and gathering device information as well as Intel proprietary commands used specifically for the pass-through interface.

10.5.2 PT capabilities

This section details manageability capabilities the XL710 provides while in SMBus mode. PT traffic is carried by the sideband interface as described in Section 10.1.

These services are not available in NC-SI mode.

When operating in SMBus mode, in addition to exposing a communication channel to the LAN for the MC, the XL710 provides the following manageability services to the MC:

- ARP handling — The XL710 can be programmed to auto-ARP replying for ARP request packets to reduce the traffic over the MC interconnect.
- Default configuration of filters by NVM — When working in SMBus mode, the default values of the manageability receive filters can be set according to the PT LAN and flex TCO NVM structures.
- Padding of short packets. Packets smaller than 60 bytes but larger than 14 bytes are padded to a legal Ethernet packet.
- CRC calculation — The XL710 adds an Ethernet CRC on all sent packets.

10.5.3 Port-to-SMBus mapping

The XL710 is identified on the SMBus manageability link as four different devices (for example, via four different SMBus addresses on which each device is connected to a different LAN port). There is no logical connection between the four devices.

The fail-over between the LAN ports is done by the MC (by sending/receiving packets through different devices). The status report to the MC, ARP handling, DHCP, and other pass-through functionality are unique for each port and configured by the MC.

10.5.4 Automatic Ethernet ARP operation

The XL710 can offload the Ethernet Address Resolution Protocol (ARP) for the MC in order to reduce the bandwidth required on the SMBus link.

Automatic Ethernet ARP parameters are loaded from the NVM when the XL710 is powered up or configured through the sideband management interface. The following parameters should be configured in order to enable ARP operation:

- ARP auto-reply enabled
- ARP IP address (to filter ARP packets)
- ARP MAC addresses (for ARP responses)

These are all configurable over the sideband interface using the advanced version of the Receive Enable command.

When an ARP request packet is received and ARP auto-reply is enabled, the XL710 checks the targeted IP address (after the packet has passed L2 checks and ARP checks). If the targeted IP matches the IP configuration for the XL710, it replies with an ARP response.

The XL710 responds to ARP request targeted to the ARP IP address with the configured ARP MAC address. In case that there is no match, the XL710 silently discards the packets. If the XL710 is not configured to do an auto-ARP response, it can be configured to forward the ARP packets to the MC, which can respond to ARP requests.

When the external MC uses the same IP and MAC address of the operating system, the ARP operation should be coordinated with the host operating system.

Note: If sharing the MAC and IP with the host operating system is possible, the XL710 provides the ability to read the system MAC address, enabling the MC to share the MAC address. However, there is no mechanism provided by the XL710 to read the IP address. The host operating system (or an agent within) and the MC must coordinate the sharing of IP addresses.

10.5.5 SMBus transactions

This section gives a brief overview of the SMBus protocol. Following is an example for a format of a typical SMBus transaction.

Table 10-7. 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 10-8 and are fully defined in the SMBus specification.

Table 10-8. Shorthand field names

Field Name	Definition
S	SMBus START Symbol.
P	SMBus STOP Symbol.
PEC	Packet Error Code.
A	ACK (Acknowledge).

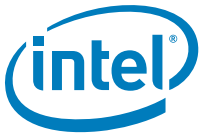


Table 10-8. Shorthand field names

Field Name	Definition
N	NACK (Not Acknowledge).
Rd	Read Operation (Read Value = 1b).
Wr	Write Operation (Write Value = 0b).

10.5.5.1 SMBus addressing

The SMBus is presented as up to four SMBus devices on the SMBus (four addresses). All PT functionality is duplicated on the SMBus address, where each SMBus address is connected to a different LAN port. Note that it is not permitted to configure multiple ports to the same SMBus address. When a LAN function is disabled, the corresponding SMBus address is not presented to the MC.

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 XL710's power up and resets.

10.5.5.2 SMBus ARP functionality

The XL710 supports the SMBus ARP protocol as defined in the SMBus 2.0 specification. The XL710 is a persistent slave address device so its SMBus address is valid after power-up and loaded from the NVM. The XL710 supports all SMBus ARP commands defined in the SMBus specification both general and directed.

SMBus ARP capability can be disabled through the NVM.

10.5.5.3 SMBus ARP flow

SMBus ARP flow is based on the status of two flags:

- Address Valid (AV): This flag is set when the XL710 has a valid SMBus address.
- Address Resolved (AR): This flag is set when the XL710 SMBus address is resolved (SMBus address was assigned by the SMBus ARP process).

These flags are internal XL710 flags and are not exposed to external SMBus devices.

Since the XL710 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 XL710 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 XL710 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 XL710 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 XL710 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 XL710 SMBus address by issuing an Assign Address command. The XL710 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 XL710 does not respond to the Get UDID General command. Note that all other commands are processed even if the AR flag is set. The XL710 stores the SMBus address that was assigned in the SMBus ARP process in the NVM, so at the next power up, it returns to its assigned SMBus address.

Figure 10-5 shows XL710 SMBus ARP flow.

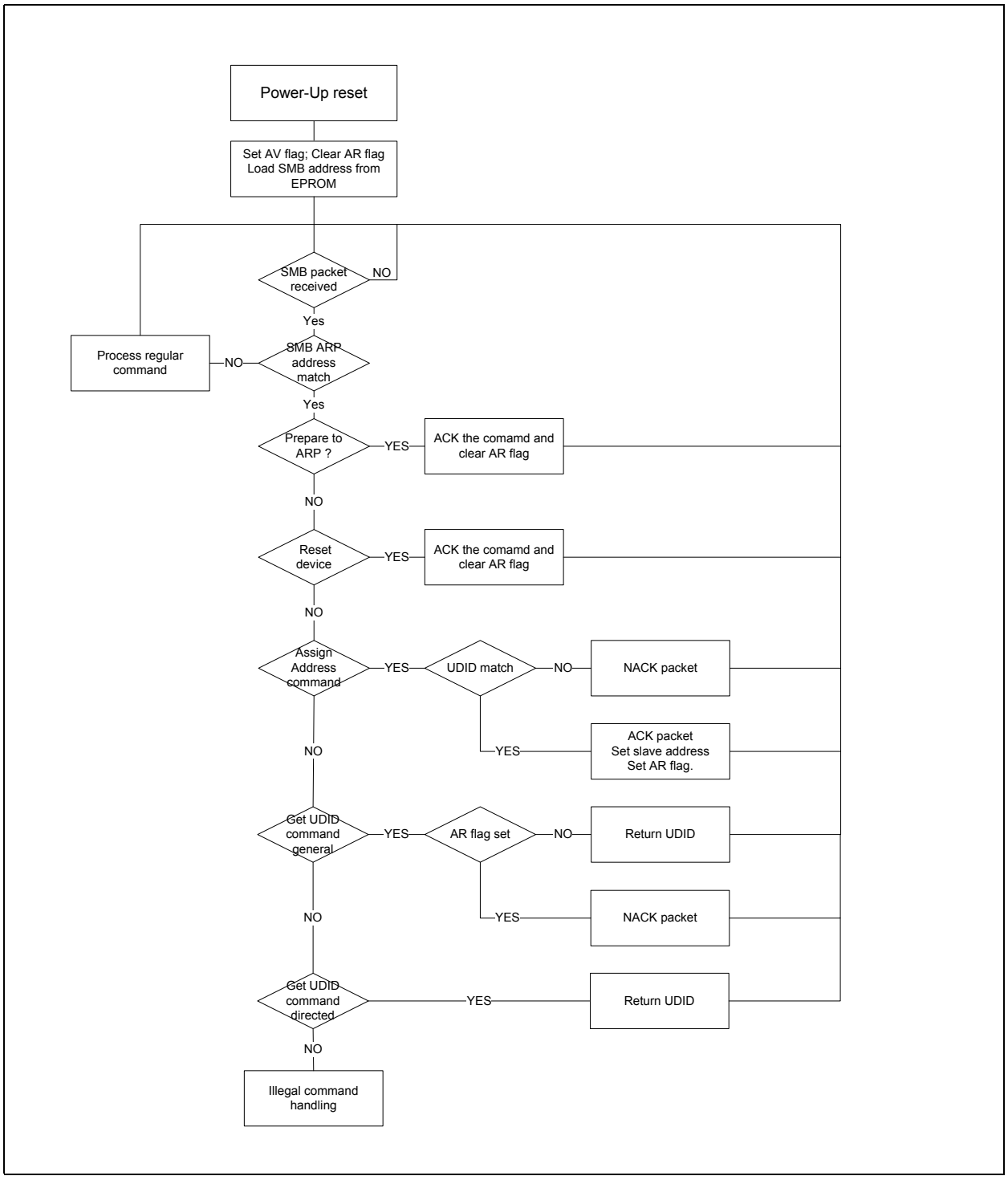


Figure 10-5. SMBus ARP flow

10.5.5.4 SMBus ARP UDID content

The UDID provides a mechanism to isolate each device for the purpose of address assignment. Each device has a unique identifier. The 128-bit number is comprised of the following fields:

Table 10-9. 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	0x154B	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: 0x154B.
Interface:	Identifies the protocol layer interfaces supported over the SMBus connection by the device. Bits 3:0 = 0x4 indicates SMBus Version 2.0. Bit 5 (ASF bit) = 1 in MCTP mode.
Subsystem Fields:	These fields are not supported and return zeros.

Device Capabilities: Dynamic and Persistent Address, *PEC Support* bit:

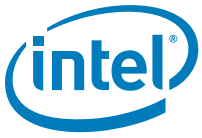
7	6	5	4	3	2	1	0
Address Type		Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)	PEC Supported
0b	1b	0b	0b	0b	0b	0b	0/1b[1]
MSB							LSB

1. The value is set according to the SMBus transaction PEC bit in the NVM.

Version/Revision: UDID Version 1, Silicon Revision:

7	6	5	4	3	2	1	0
Reserved (0)	Reserved (0)	UDID Version			Silicon Revision ID		
0b	0b	001b			See the following table		
MSB							LSB

Silicon Revision ID:



Silicon Version	Revision ID
A0	000b
B0	001b
B1	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 XL710 there are four 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

10.5.5.5 SMBus ARP and multi-port

The XL710 responds as four SMBus devices having four sets of AR/AV flags (one for each port). The XL710 responds four time to the SMBus ARP master, once each for each port. All SMBus addresses are taken from the SMBus ARP address word of the NVM.

Note that the Unique Device Identifier (UDID) is different for the four ports in the version ID field, which represents the MAC address and is different for the four ports. The XL710 first respond as port 0, and only when an address is assigned, then start responding as port 1,2 and 3 to the Get UDID command.

10.5.5.6 Concurrent SMBus transactions

The SMBus interface is single threaded. Thus, concurrent SMBus transactions are not permitted. Once a transaction starts, it must complete before an additional transaction is initiated.

A transaction is defined as:

- All the SMBus commands used to receive a packet.
- All the SMBus commands used to send a packet.
- The read and write SMBus commands used as part of read parameters described in Section 10.5.11.2.
- The single write SMBus commands described in Section 10.5.11.1.

10.5.6 SMBus notification methods

The XL710 supports three methods of notifying the MC that it has information that needs to be read by the MC:

- SMBus alert — Refer to Section 10.5.6.1.

- Asynchronous notify — Refer to Section 10.5.6.2.
- Direct receive — refer to Section 10.5.6.3.

The notification method used by the XL710 can be configured from the SMBus using the Receive Enable command (Section 10.5.11.1.3). The default method is set by the NVM in the *Notification Method* field in LAN Receive Enable 1 (Section 7.2.32.9).

Note: The SMBus notification method used must be the same for all ports.

The following events cause the XL710 to send a notification event to the MC:

- Receiving a LAN packet that is designated to the MC.
- Firmware was reset and requires re-initialization.
- Receiving a Request Status command from the MC initiates a status response.
- The XL710 is configured to notify the MC upon status changes (by setting the EN_STA bit in the Receive Enable command) and one of the following events happen:
  — TCO Command aborted
  — Link status changed
  — Power state change

There can be cases where the MC is hung and not responding to the SMBus notification. The XL710 has a time-out value (defined in the NVM) to avoid hanging while waiting for the notification response. If the MC does not respond until the time out expires, the notification is de-asserted and all pending data is silently discarded.

Note that the SMBus notification time-out value can only be set in the NVM. The MC cannot modify this value.

10.5.6.1 SMBus alert and alert response method

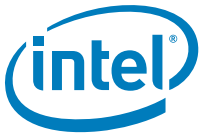
The SMBus Alert# (SMBALERT_N) signal is an additional SMBus signal that acts as an asynchronous interrupt signal to an external SMBus master. The XL710 asserts this signal each time it has a message that it needs the MC to read and if the chosen notification method is the SMBus alert method. Note that the SMBus alert method is an open-drain signal which means that other devices besides the XL710 can be connected on the same alert pin. As a result, the MC needs a mechanism to distinguish between the alert sources.

The MC can respond to the alert by issuing an ARA Cycle command to detect the alert source device. The XL710 responds to the ARA cycle with its own SMBus slave address (if it was the SMBus alert source) and de-asserts the alert when the ARA cycle is completes. Following the ARA cycle, the MC issues a read command to retrieve the XL710 message.

Some MCs do not implement the ARA cycle transaction. These MCs respond to an alert by issuing a Read command to the XL710 (0xC0/0xD0 or 0xDE). The XL710 always responds to a Read command, even if it is not the source of the notification. The default response is a status transaction. If the XL710 is the source of the SMBus Alert, it replies the read transaction and then de-asserts the alert after the command byte of the read transaction.

Note: In SMBus Alert mode, the SMBALERT_N pin is used for notification. Each port generate alerts on events that are independent of each other.

Note: If two ports have events to notify, the second alert is asserted only after the first event is handled.



The ARA cycle is an SMBus receive byte transaction to SMBus Address 0001b - 100b. Note that the ARA transaction does not support PEC. The ARA transaction format is as follows:

1	7	1	1	8	1	1	1
S	Alert Response Address	Rd	A	Slave Device Address		A	P
	0001 100	1	0	Manageability Slave SMBus Address	0	1	

Note: If the MC does not react to the alert in the delay defined by the *SMBus Notification Timeout* NVM field (Section 7.2.34.3), the ALERT pin is de-asserted and the Rx packet indication in the status word is cleared (and packet is dropped).

10.5.6.2 Asynchronous notify method

When configured using the asynchronous notify method, the XL710 acts as a SMBus master and notifies the BMC of one of the events listed in Section 10.5.6 by issuing a modified form of the write word transaction. The asynchronous notify transaction SMBus address and data payload is configured using the Receive Enable command (Section 10.5.11.1.3) or using the NVM defaults. Note that the asynchronous notify is not protected by a PEC byte.

1	7	1	1	7	1	1	8	1	8	1	1
S	Target Address	Wr	A	Sending Device Address		A	Data Byte Low	A	Data Byte High	A	P
	MC Slave Address	0	0	MNG Slave SMBus Address	0	0	Interface	0	Alert Value	0	

The target address and data byte low/high are taken from the Receive Enable command or NVM configuration (Section 7.2.32.9 and Section 7.2.32.10).

If the MC does not read the status in the delay defined by the *SMBus Notification Timeout* NVM field (Section 7.2.34.3), the Rx packet indication in the status word is cleared (and packet is dropped).

10.5.6.3 Direct receive method

If configured, the XL710 has the capability to send a message it needs to transfer to the external MC as a master over the SMBus instead of alerting the MC and waiting for it to read the message.

The message format follows. Note that the command that is used is the same command that is used by the external MC in the Block Read command. The opcode that the XL710 puts in the data is also the same as it put in the Block Read command of the same functionality. The rules for the *F* and *L* flags (bits) are also the same as in the Block Read command.

1	7	1	1	1	1	6	1	
S	Target Address	Wr	A	F	L	Command	A	...

1	7	1	1	1	1	6	1	
	BMC Slave Address	0	0	First Flag	Last Flag	Receive TCO Command 01 0000b	0	

8	1	8	1		1	8	1	1
Byte Count	A	Data Byte 1	A	...	A	Data Byte N	A	P
N	0		0		0		0	

10.5.7 Receive PT flow

The XL710 is used as a channel for receiving packets from the network link and passing them to the external MC. The MC configures the XL710 to pass these specific packets to the MC. Once a full packet is received from the link and identified as a manageability packet that should be transferred to the MC, the XL710 starts the receive TCO flow to the MC.

The XL710 uses the SMBus notification method to notify the MC that it has data to deliver. Since the packet size might be larger than the maximum SMBus fragment size, the packet is divided into fragments, where the XL710 uses the maximum fragment size allowed in each fragment (configured via the NVM). The last fragment of the packet transfer is always the status of the packet. As a result, the packet is transferred in at least two fragments. The data of the packet is transferred as part of the receive TCO LAN packet transaction.

When SMBus alert is selected as the MC notification method, the XL710 notifies the MC on each fragment of a multi-fragment packet. When asynchronous notify is selected as the MC notification method, the XL710 notifies the MC only on the first fragment of a received packet. It is the MC's responsibility to read the full packet including all the fragments.

Any timeout on the SMBus notification results in discarding the entire packet. Any NACK by the MC causes the fragment to be re-transmitted to the MC on the next Receive Packet command.

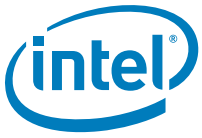
The maximum size of the received packet is limited by the XL710 to 1536 bytes. Packets larger than 1536 bytes are silently discarded. Any packet smaller than 1536 bytes is processed.

10.5.8 Transmit PT flow

The XL710 is used as the channel for transmitting packets from the external MC to the network link. The network packet is transferred from the MC over the SMBus and then, when fully received by the XL710, is transmitted over the network link.

Each SMBus address is connected to a different LAN port. When a packet is received during a SMBus transaction using SMBus address #0, it is transmitted to the network using LAN port #0; it is transmitted through LAN port #1 if received on SMBus address #1, etc.

The XL710 supports packets up to an Ethernet packet length of 1536 bytes. Since SMBus transactions can only be up to 240 bytes in length, packets might need to be transferred over the SMBus in more than one fragment. This is achieved using the *F* and *L* bits in the command number of the transmit TCO packet Block Write command. When the *F* bit is set, it is the first fragment of the packet. When the *L* bit is set, it is the last fragment of the packet. When both bits are set, the entire packet is in one fragment.



The packet is sent over the network link only after all its fragments are received correctly over the SMBus. The maximum SMBus fragment size is defined within the NVM and cannot be changed by the BMC.

The minimum packet length defined by the 802.3 spec is 64 bytes. The XL710 pads packets that are less than 64 bytes to meet the specification requirements (there is no need for the external MC to pad packets less than 64 bytes). If the packet sent by the MC is larger than 1536 bytes, the XL710 silently discards the packet. The minimal packet size that the XL710 can handle is 14 bytes.

The XL710 calculates the L2 CRC on the transmitted packet and adds its four bytes at the end of the packet. Any other packet field (such as XSUM or VLAN) must be calculated and inserted by the MC (the XL710 does not change any field in the transmitted packet, other than adding padding and CRC bytes).

If the network link is down when the XL710 has received the last fragment of the packet from the MC, it silently discards the packet. Note that any link down event during the transfer of any packet over the SMBus does not stop the operation since the XL710 waits for the last fragment to end to see whether the network link is up again.

10.5.8.1 Transmit errors in sequence handling

Once a packet is transferred over the SMBus from the MC to the XL710, the *F* and *L* flags should follow specific rules. The *F* flag defines the first fragment of the packet; the *L* flag that the transaction contains the last fragment of the packet. Table 10-10 lists the different flag options in transmit packet transactions.

Table 10-10. Flag options during transmit packet transactions

Previous	Current	Action/Notes
Last	First	Accept both.
Last	Not First	Error for the current transaction. Current transaction is discarded and an abort status is asserted.
Not Last	First	Error in previous transaction. Previous transaction (until previous First) is discarded. Current packet is processed. No abort status is asserted.
Not Last	Not First	Process the current transaction.

Note: Since every other Block Write command in TCO protocol has both *F* and *L* flags set, they cause flushing any pending transmit fragments that were previously received. When running the TCO transmit flow, no other Block Write transactions are allowed in between the fragments.

10.5.8.2 TCO command aborted flow

The XL710 indicates to the MC an error or an abort condition by setting the *TCO Abort* bit in the general status. The XL710 might also be configured to send a notification to the MC (see Section 10.5.11.1.3.3).

Following is a list of possible error and abort conditions:

- Any error in the SMBus protocol (NACK, SMBus timeouts, etc.).
- If the MC does not respond until the notification timeout (programmed in the NVM) expires.

- Any error in compatibility between required protocols to specific functionality (for example, Rx Enable command with a byte count not equal to 1/14, as defined in the command specification).
- If the XL710 does not have space to store the transmitted packet from the MC (in its internal buffer space) before sending it to the link, the packet is discarded and the external MC is notified via the *Abort* bit.
- Error in the *F/L* bit sequence during multi-fragment transactions.
- An internal reset to the XL710's firmware.

10.5.9 SMBus link state control

While in SMBus mode, the default setting of the link is defined by the *EMP_LINK_ON* bit in Common Firmware Parameters 2 NVM word.

When a channel is enabled through NVM setting or through the *RCV_EN* option of the Receive Enable command, the link is established (if not already required for other purposes).

If the channel is disabled by clearing of the *RCV_EN* option, then the link might move back to the default defined by the *EMP_LINK_ON* if not needed for other purposes.

Note: Before transitioning to D3 it is the responsibility of the software device driver to request the PHY to be active for wake-up activities.

10.5.10 SMBus ARP transactions

All SMBus ARP transactions include the PEC byte.

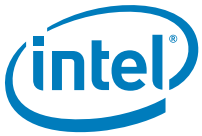
10.5.10.1 Prepare to ARP

This command clears the *Address Resolved* flag (set to false). It does not affect the status or validity of the dynamic SMBus address and is used to inform all devices that the ARP master is starting the ARP process:

1	7	1	1	8	1	8	1	1
S	Slave Address	Wr	A	Command	A	PEC	A	P
	1100 001	0	0	0000 0001	0	[Data Dependent Value]	0	

10.5.10.2 Reset device (general)

This command clears the *Address Resolved* flag (set to false). It does not affect the status or validity of the dynamic SMBus address.



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 Value]	0	

10.5.10.3 Reset device (directed)

The Command field is NACKed if bits 7:1 do not match the current SMBus address. This command clears the *Address Resolved* flag (set to false) and does not affect the status or validity of the dynamic SMBus address.

1	7	1	1	8	1	8	1	1
S	Slave Address	Wr	A	Command	A	PEC	A	P
	1100 001	0	0	Targeted Slave Address | 0	0	[Data Dependent Value]	0	

10.5.10.4 Assign address

This command assigns SMBus address. The address and command bytes are always acknowledged.

The transaction is aborted (NACKed) immediately if any of the UDID bytes is different from XL710 UDID bytes. If successful, the manageability system internally updates the SMBus address. This command also sets the *Address Resolved* flag (set to true).

1	7	1	1	8	1	8	1	
S	Slave Address	Wr	A	Command	A	Byte Count	A	· · ·
	1100 001	0	0	0000 0100	0	0001 0001	0	

8	1	8	1	8	1	8	1	
Data 1	A	Data 2	A	Data 3	A	Data 4	A	· · ·
UDID Byte 15 (MSB)	0	UDID Byte 14	0	UDID Byte 13	0	UDID Byte 12	0	

8	1	8	1	8	1	8	1	
Data 5	A	Data 6	A	Data 7	A	Data 8	A	· · ·
UDID Byte 11	0	UDID Byte 10	0	UDID Byte 9	0	UDID Byte 8	0	

8	1	8	1	8	1	
Data 9	A	Data 10	A	Data 11	A	· · ·
UDID Byte 7	0	UDID Byte 6	0	UDID Byte 5	0	

8	1	8	1	8	1	8	1	
Data 12	A	Data 13	A	Data 14	A	Data 15	A	· · ·
UDID Byte 4	0	UDID Byte 3	0	UDID Byte 2	0	UDID Byte 1	0	

8	1	8	1	8	1	1
Data 16	A	Data 17	A	PEC	A	P
UDID Byte 0 (LSB)	0	Assigned Address	0	[Data Dependent Value]	0	

10.5.10.5 Get UDID (general and directed)

The general get UDID SMBus transaction supports a constant command value of 0x03 and, if directed, supports a Dynamic command value equal to the dynamic SMBus address.

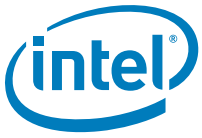
If the SMBus address has been resolved (*Address Resolved* flag set to true), the manageability system does not acknowledge (NACK) this transaction. If it's a General command, the manageability system always acknowledges (ACKs) as a directed transaction.

This command does not affect the status or validity of the dynamic SMBus address or the *Address Resolved* flag.

S	Slave Address	Wr	A	Command	A	S	• • •
	1100 001	0	0	See Below	0		

7	1	1	8	1	
Slave Address	Rd	A	Byte Count	A	· · ·
1100 001	1	0	0001 0001	0	

8	1	8	1	8	1	8	1	
Data 1	A	Data 2	A	Data 3	A	Data 4	A	· · ·



8	1	8	1	8	1	8	1	
UDID Byte 15 (MSB)	0	UDID Byte 14	0	UDID Byte 13	0	UDID Byte 12	0	

8	1	8	1	8	1	8	1	
Data 5	A	Data 6	A	Data 7	A	Data 8	A	· · ·
UDID Byte 11	0	UDID Byte 10	0	UDID Byte 9	0	UDID Byte 8	0	

8	1	8	1	8	1	
Data 9	A	Data 10	A	Data 11	A	· · ·
UDID Byte 7	0	UDID Byte 6	0	UDID Byte 5	0	

8	1	8	1	8	1	8	1	
Data 12	A	Data 13	A	Data 14	A	Data 15	A	· · ·
UDID Byte 4	0	UDID Byte 3	0	UDID Byte 2	0	UDID Byte 1	0	

8	1	8	1	8	1	1
Data 16	A	Data 17	A	PEC	~Ã	P
UDID Byte 0 (LSB)	0	Device Slave Address	0	[Data Dependent Value]	1	

The Get UDID command depends on whether or not this is a Directed or General command.

The General Get UDID SMBus transaction supports a constant command value of 0x03.

The Directed Get UDID SMBus transaction supports a Dynamic command value equal to the dynamic SMBus address with the LSB bit set.

Note: Bit 0 (LSB) of Data byte 17 is always 1b.

10.5.11 SMBus PT Transactions

This section details commands (both read and write) that the XL710 SMBus interface supports for PT.

10.5.11.1 Write SMBus Transactions

This section details the commands that the MC can send to the XL710 over the SMBus interface. The SMBus write transactions table lists the different SMBus write transactions supported by the XL710.

TCO Command	Transaction	Command	Fragmentation	Section
Transmit Packet	Block Write	First: 0x84 Middle: 0x04 Last: 0x44	Multiple	10.5.11.1.1
Transmit Packet	Block Write	Single: 0xC4	Single	10.5.11.1.1
Request Status	Block Write	Single: 0xDD	Single	10.5.11.1.2
Receive Enable	Block Write	Single: 0xCA	Single	10.5.11.1.3
Force TCO	Block Write	Single: 0xCF	Single	10.5.11.1.4
Management Control	Block Write	Single: 0xC1	Single	10.5.11.1.5
Update MNG RCV Filter Parameters	Block Write	Single: 0xCC	Single	10.5.11.1.6
Set Common Filters	Block Write	Single: 0xC2	Single	10.5.11.1.7
Clear All Filters	Block Write	Single: 0xC3	Single	10.5.11.1.8

10.5.11.1.1 Transmit packet command

The Transmit Packet command behavior is detailed in Section 10.5.8. The Transmit Packet fragments have the following format.

The payload length is limited to the maximum payload length set in the NVM. If the overall packet length is bigger than 1536 bytes, the packet is silently discarded.

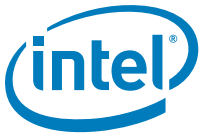
Function	Command	Byte Count	Data 1	...	Data N
Transmit first fragment	0x84	N	Packet data MSB	...	Packet data LSB
Transmit middle fragment	0x04				
Transmit last fragment	0x44				
Transmit single fragment	0xC4				

10.5.11.1.2 Request status command

An external MC can initiate a request to read XL710 manageability status by sending a Request Status command. When received, the XL710 initiates a notification to an external MC when status is ready. After this, the external controller is able to read the status, by issuing a Read Status command (see Section 10.5.11.2.2).

The format is as follows:

Function	Command	Byte Count	Data 1
Request Status	0xDD	1	0



10.5.11.1.3 Receive enable command

The Receive Enable command is a single fragment command used to configure the XL710. This command has two formats: short, 1-byte legacy format (providing backward compatibility with previous components) and long, 14-byte advanced format (allowing greater configuration capabilities). The Receive Enable command format is as follows:

Function	CMD	Byte Count	Data 1	Data 2	...	Data 7	Data 8	...	Data 11	Data 12	Data 13	Data 14
Legacy Receive Enable	0xCA	1	Receive Control Byte	-	...	-	-	...	-	-	-	-
Advanced Receive Enable		14 (0x0E)		MAC Addr MSB		MAC Addr LSB	IP Addr MSB		IP Addr LSB	BMC SMBus Addr	I/F Data Byte	Alert Value Byte

Field	Bit(s)	Description
RCV_EN	0	Receive TCO Enable. 0b = Disable receive TCO packets. 1b = Enable Receive TCO packets. Setting this bit enables all manageability receive filtering operations. Enabling specific filters is done via the NVM or through special configuration commands. **Note:** When the *RCV_EN* bit is cleared, all receive TCO functionality is disabled, not just the packets that are directed to the MC (also auto ARP packets).
RCV_ALL	1	Receive All Enable. 0b = Disable receiving all packets. 1b = Enable receiving all packets. Forwards all packets received over the wire that passed L2 filtering to the external MC. This flag has no effect if bit 0 (Enable TCO packets) is disabled.
EN_STA	2	Enable Status Reporting. 0b = Disable status reporting. 1b = Enable status reporting.
EN_ARP_RES	3	Enable ARP Response. 0b = Disable the XL710 ARP response. The XL710 treats ARP packets as any other packet, for example, packet is forwarded to the MC if it passed other (non-ARP) filtering. 1b = Enable the XL710 ARP response. The XL710 automatically responds to all received ARP requests that match the IP address programmed by the MC. The MC IP address is provided as part of the Receive Enable message (bytes 8:11). If a short version of the command is used, the XL710 uses IP address configured in the most recent long version of the command in which the EN_ARP_RES bit was set. If no such previous long command exists, then the XL710 uses the IP address configured in the NVM as ARP Response IPv4 address in the PT LAN configuration structure. If the *CBDM* bit is set, the XL710 uses the BMC dedicated MAC address in ARP response packets. If the *CBDM* bit is not set, the MC uses the host MAC address. When the enable ARP response feature is activated, the XL710 uses the following registers to filter in ARP requests. MC should not modify these registers: • Manageability Decision Filter – MDEF7 (and corresponding bit 7 in Management Only traffic register – *MNGONLY*). • Fourth IPv4 Filter.

Field	Bit(s)	Description
NM	5:4	Notification Method. Define the notification method the XL710 uses. 00b = SMBUS alert. 01b = Asynchronous notify. 10b = Direct receive. 11b = Not supported. ***Note:*** Changing the notification method in any port updates the notification method of all ports.
Reserved	6	Reserved. Must be set to 1b.
CBDM	7	Configure the MC Dedicated MAC Address. **Note:** This bit should be 0b when the *RCV_EN* bit (bit 0) is not set. 0b = The XL710 shares the MAC address for MNG traffic with the host MAC address, which is specified in NVM words 0x0-0x2. The MAC filtering is not enforced. See the note at the bottom of this table. 1b = The XL710 uses the MC dedicated MAC address as a filter for incoming receive packets. The MC MAC address is set in bytes 2-7 in this command. If a short version of the command is used, the XL710 uses the MAC address configured in the most recent long version of the command in which the *CBDM* bit was set. When the dedicated MAC address feature is activated, the XL710 uses the following registers to filter in all the traffic addressed to the BMC MAC. BMC can not modify these registers: Manageability Decision Filter – MDEF7 (and corresponding bit 7 in Management Only traffic register – *PRT_MNG_MNGONLY*). Manageability Decision Filter – MDEF6 (and corresponding bit 6 in Management Only traffic register – *MNGONLY*). Manageability MAC Address Low – *PRT_MNG_MMAL[3].* Manageability MAC Address High – *PRT_MNG_MMAH[3].* ***Note:*** When the dedicated MAC address feature is cleared, these registers are not programmed and the BMC may use other filters to enforce MAC filtering using the Update Management Receive Filter Parameters command.

10.5.11.1.3.1 Management MAC address (data bytes 7:2)

Ignored if the *CBDM* bit is not set. This MAC address is used to configure the dedicated MAC address. In addition, it is used in the ARP response packet when the *EN_ARP_RES* bit is set. This MAC address is also used when *CBDM* bit is set in subsequent short versions of this command.

10.5.11.1.3.2 Management IP address (data bytes 11:8)

This IP address is used to filter ARP request packets.

10.5.11.1.3.3 Asynchronous notification SMBus address (data byte 12)

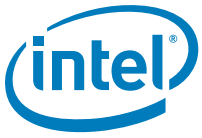
This address is used for the asynchronous notification SMBus transaction and for direct receive. The SMBus address is stored in bit 7:1 of this byte. Bit 0 is always 0b.

10.5.11.1.3.4 Interface data (data byte 13)

Interface data byte used in asynchronous notification.

10.5.11.1.3.5 Alert value data (data byte 14)

Alert value data byte used in asynchronous notification.



10.5.11.1.4 Force TCO command

This command causes the XL710 to perform a TCO reset, TCO isolate, or firmware reset

TCO reset — If force TCO reset is enabled in the NVM (see Section 7.2.31.2), the force TCO reset clears the data path (Rx/Tx) of the XL710 to enable the MC to transmit/receive packets through the XL710 by asserting a global reset. This command should only be used when the MC is unable to transmit receive and suspects that the XL710 is inoperable. The command also causes the LAN software device driver to unload. It is recommended to perform a system restart to resume normal operation.

TCO isolate — if TCO isolate is enabled in the NVM (See Section 7.2.31.3), the TCO Isolate command disables PCIe write operations to the LAN port. If TCO isolate is disabled in NVM, the XL710 does not execute the command but sends a response to the MC with successful completion. Following a TCO isolate, management sets *EMP_TCO_ISOLATE.EMP_TCO_ISOLATE* to 1b for all the PFs associated with the port on which this command is received.

Firmware reset — This command causes re-initialization of all the embedded controller functions and re-load of related NVM words (like a firmware patch code). Applying this command resets the entire device as well as having an effect on TCO reset. A firmware reset is achieved by setting the *GSCR.SET_FWRST* aux bit.

Note: A firmware reset causes a global reset of the entire device (GLOBR).

The XL710 considers the Force TCO Reset command as an indication that the operating system is unavailable. The Force TCO command format is as follows:

Function	Command	Byte Count	Data 1
Force TCO Reset	0xCF	1	TCO Mode

Where TCO mode is:

Field	Bit(s)	Description
DO_TCO_RST	0	Perform 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. ***Note:*** Should be used for debug only.
RESET_MGMT	2	Reset Manageability; re-load manageability NVM words. 0b = Do nothing. 1b = Issue firmware reset to manageability. Setting this bit generates a one-time firmware reset. Following the reset, management related data from NVM is loaded.
Reserved	7:3	Reserved (set to 0x00).

1. TCO isolate host write operation enabled in NVM.

Note: Only one of the fields should be set in a given command. Setting more than one field might yield unexpected results.

10.5.11.1.5 Management control

This command is used to set generic manageability parameters. The parameters are listed in Table 10-11. The command is 0xC1 stating that it is a Management Control command. The first data byte is the parameter number and the data afterwards (length and content) are parameter specific as shown in Management Control Command Parameters/Content.

Note: If the parameter that the MC sets is not supported by the XL710. The XL710 does not NACK the transaction. After the transaction ends, the XL710 discards the data and asserts a transaction abort status.

The Management Control command format is as follows:

Function	Command	Byte Count	Data 1	Data 2	...	Data N
Management Control	0xC1	N	Parameter Number	Parameter Dependent		

Table 10-11. Management Control Command Parameters/Content

Parameter	#	Parameter Data
Keep PHY Link Up	0x00	A single byte parameter: Data 2: Bit 0 = Set to indicate that the PHY link for this port should be kept up throughout system resets. This is useful when the server is reset and the MC needs to keep connectivity for a manageability session. Bit [7:1] = Reserved. 0b = Disabled. 1b = Enabled.

10.5.11.1.6 Update management receive filter parameters

This command is used to set the manageability receive filters parameters. The command is 0xCC. The first data byte is the parameter number and the data that follows (length and content) are parameter specific as listed in management RCV filter parameters.

If the parameter that the MC sets is not supported by the XL710, then the XL710 does not NACK the transaction. After the transaction ends, the XL710 discards the data and asserts a transaction abort status.

The update management RCV receive filter parameters command format is as follows:

Function	Command	Byte Count	Data 1	Data 2	...	Data N
Update Manageability Filter Parameters	0xCC	N	Parameter Number	Parameter Dependent		

Table 10-12 lists the different parameters and their content.

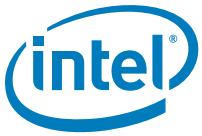


Table 10-12. Management receive filter parameters

Parameter	Number	Parameter Data
Filters Enables	0x1	Defines the generic filters configuration. The structure of this parameter is four bytes as the Manageability Control (*PRT_MNG_MANC*) register. ***Note:*** The general filter enable is in the Receive Enable command that enables receive filtering.
MNGONLY configuration	0xF	This parameter defines which of the packets types identified as manageability packets in the receive path will never be directed to the host memory. Data 2:5 = *PRT_MNG_MNGONLY* register bytes - Data 2 is the MSB.
Flex Filter 0 Enable Mask and Length	0x10	Flex Filter 0 Mask. Data 17:2 = Mask. Bit 0 in data 2 is the first bit of the mask. Data 19:18 = Reserved. Should be set to 00b. Date 20 = Flexible filter length.
Flex Filter 0 Data	0x11	Data 2 — Group of flex filter's bytes: 0x0 = bytes 0-29 0x1 = bytes 30-59 0x2 = bytes 60-89 0x3 = bytes 90-119 0x4 = bytes 120-127 Data 3:32 = Flex filter data bytes. Data 3 is LSB. Group's length is not a mandatory 30 bytes; it might vary according to filter's length and must NOT be padded by zeros.
Decision Filters	0x61	This command is obsolete and should not be used. Please use 0x68 instead.
VLAN Filters	0x62	Three bytes are required to load the VLAN tag filters. Data 2: VLAN filter number. Data 3: MSB of VLAN filter. Data 4: LSB of VLAN filter.
Flex Port Filters	0x63	Three to four bytes are required to load the manageability flex port filters. Data 2 = Flex port filter number. Data 3 = MSB of flex port filter. Data 4 = LSB of flex port filter. Data 5: Bit 0 = Match UDP ports Bit 1 = Match TCP ports Bit 2 = Match destination port (0) or source port (1). If Data 5 is not present, the match is done on TCP and UDP destination ports (legacy behavior).
IPv4 Filters	0x64	Five bytes are required to load the IPv4 address filter. Data 2 = IPv4 address filter number (3:0). Data 3 = LSB of IPv4 address filter. … Data 6 = MSB of IPv4 address filter.
IPv6 Filters	0x65	17 bytes are required to load the IPv6 address filter. Data 2 = IPv6 address filter number (3:0). Data 3 = LSB of IPv6 address filter. … Data 18 = MSB of IPv6 address filter.
MAC Filters	0x66	Seven bytes are required to load the MAC address filters. Data 2 = MAC address filters pair number (3:0). Data 3 = MSB of MAC address. … Data 8 = LSB of MAC address.

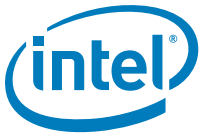
Table 10-12. Management receive filter parameters

Parameter	Number	Parameter Data
EtherType Filters	0x67	Five bytes to load Ethertype Filters (METF). Data 2 = METF filter index (valid values are 0, 1, 2, 3). Data 3 = MSB of METF. ... Data 6 = LSB of METF.
Extended Decision Filter	0x68	Nine bytes to load the extended decision filters (MDEF_EXT & MDEF). Data 2 = MDEF filter index (valid values are 0...6). Data 3 — MSB of MDEF_EXT (DecisionFilter1). .... Data 6 = LSB of MDEF_EXT (DecisionFilter1). Data 7 = MSB of MDEF (DecisionFilter0). .... Data 10 = LSB of MDEF (DecisionFilter0). The command overwrites any previously stored value.
Management Special Filter Modifiers	0x69	Four bytes to load the Management Special Filter Modifiers. Data 2 = MSB of MSFM register. ... Data 5 = LSB of MSFM register.

Table 10-13. Filter enable parameters

Bit	Name	Description
16:0	Reserved	Reserved.
17	RCV_TCO_EN	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 EN_BMC2OS or EN_BMC2NET bits are set. This bit is usually set using the receive enable command (see Section 10.5.11.1.3).
18	KEEP_PHY_LINK_UP	Block PHY reset and power state changes. When this bit is set the PHY reset and power state changes does not get to the PHY.
22:19	Reserved	Reserved.
23	Enable Xsum Filtering to MNG	When this bit is set, only packets that pass the L3 and L4 checksum are send to the manageability block.
24	Reserved	Reserved.
25	FIXED_NET_TYPE	Fixed Net Type. If set, only packets matching the net type defined by the *NET_TYPE* field passes to manageability. Otherwise, both tagged and un-tagged packets can be forwarded to the manageability engine.
26	NET_TYPE	Net Type. 0b = Pass only un-tagged packets. 1b = Pass only VLAN tagged packets. Valid only if *FIXED_NET_TYPE* is set.
31:27	Reserved	Reserved.

10.5.11.1.7 Set common filters command



The Set Common Filters command is a single fragment command capable of configuring the most common filters.

Note: If this command is used, all the other commands that programs forwarding filters should not be used (apart from the Clear All Fitters command). When this command is received, an implied Clear All Filters command is done before the application of this command.

The Set Common Filters command has two possible formats:

IPv4 format:

Function	Command	Byte Count	Data 1	Data 2:4	5:10	Data 11	Data 12	Data 13	Data 14:17
Set Common Filters	0xC2	17	Opcode = 0	Receive Control - see Table 10-14	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv4 Address

IPv6 format:

Function	Command	Byte Count	Data 1	Data 2:4	5:10	Data 11	Data 12	Data 13	Data 14:29
Set Common Filters	0xC2	29	Opcode = 0	Receive Control - see Table 10-14	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv6 Address

Table 10-14. Set common filters receive control bytes

Byte	Bit	Field	Description
1	0	RCV_EN	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 EN_BMC2O or EN_BMC2NET bits are set.
	1	EN_STA	Enable Status Reporting. 0b = Disable status reporting. 1b = Enable status reporting.
	2	Auto ARP	Automatically respond to ARP packets. Ignored in IPv6 mode. If this bit is set, broadcast ARP packets are handled by the XL710 and ARP requests to the IP address set in the command are responded to. ***Notes:*** Mutually exclusive to Configure ARP/ Neighborhood Filter bit. If this bit is set, the IP address must be valid. This bit is ignored if RCV_EN is cleared.
	3	Enable Xsum Filtering to MNG	When this bit is set, only packets that pass the L3 and L4 checksum are send to the manageability block. This bit is ignored if RCV_EN is cleared.
	5:4	Reserved	
	7:6	Notification Method	Notification Method. Define the notification method the XL710 uses. 00b = SMBus alert. 01b = Asynchronous notify. 10b = Direct receive. 11b = Not supported.

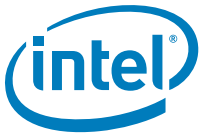


Table 10-14. Set common filters receive control bytes

Byte	Bit	Field	Description
2	8	CBDM	Configure the BMC Dedicated MAC Address. 0b = The XL710 shares the MAC address for manageability traffic with the host MAC address, which is specified in NVM words 0x0-0x2. 1b = The XL710 uses the MC dedicated MAC address as a filter for incoming receive packets. The MC MAC address is set in bytes 5:1 in this command. This bit is ignored if RCV_EN is cleared.
	9	Configure IP Address Filter	Automatically configure an IP address filter. If this bit is set, only packets matching this IP address is forwarded. If the *CBDM* bit is set, only packets matching the MAC and IP address is forwarded. This bit is ignored if RCV_EN is cleared.
	10	Configure RMCP 26Fh Filter	Automatically configure standard IPMI port 0x26F filters. If this bit is set, only packets matching this port is forwarded. If the *CBDM/ Configure IP Address Filter* bits are set, only packets matching the MAC and IP address and this port is forwarded. The other port enable bit (11) might add additional forwarding conditions. This bit is ignored if RCV_EN is cleared
	11	Configure RMCP 298h Filter	Automatically configure standard IPMI port 0x298 filter. If this bit is set, only packets matching this port is forwarded. If the *CBDM/ Configure IP Address Filter* bits are set, only packets matching the MAC and IP address and this port is forwarded. The other port enable bit (10) might add additional forwarding conditions. This bit is ignored if RCV_EN is cleared
	12	Configure ARP/ Neighborhood Filter	Automatically configure filters to enable this traffic to the MC (mutually exclusive to *Auto ARP* bit). If this bit is set, broadcast ARP packets are forwarded to the MC. In IPv4 mode, setting this bit enables forwarding of broadcast ARP requests and responses and unicast ARP responses. If the IP address is set, only a response to this address is forwarded. In IPv6 mode, setting this bit enables forwarding of all types of neighbor discovery and MLD ICMPv6 packet types: • 0x86 (134d) = Router Advertisement. • 0x87 (135d) = Neighbor Solicitation. • 0x88 (136d) = Neighbor Advertisement. • 0x89 (137d) = Redirect. • 0x82 (130d) = MLD Query. • 0x83 (131d) = MLDv1 Report. • 0x84 (132d) = MLD Done. • 0x8F (143d) = MLDv2 Report.
	13	Configure DHCP port 44h Filter	Automatically configure DHCP port 44 filter to the MC. If this bit is set, multicast packets matching this port is forwarded. Otherwise, multicast packets are not forwarded to the MC. This bit is ignored if RCV_EN is cleared or in IPv6 mode.
	15:14	Reserved	Reserved.
3	16	Disable Host ARP	Configure ARP requests and network neighborhood packets not to go to the host. This bit should be cleared during normal operation. Ignored if both bit 12 and bit 2 are cleared or if RCV_EN is cleared.
	17	Disable Host DHCP	Configure DHCP packets (port 0x44) not to go to host. This bit should be cleared in normal operation. Ignored if bit 13 is cleared, RCV_EN is cleared, or in IPv6 mode.
	24:18	Reserved	Reserved.

10.5.11.1.8 Clear all filters command

The Clear all Filters command is a single fragment command capable of clearing all the receive filters currently programmed for manageability traffic.

Function	Command	Byte Count	Data
Clear all Filters	0xC3	1	0x00

10.5.11.2 Read SMBus transactions

This section details the PT read transactions that the MC can send to the XL710 over the SMBus.

SMBus read transactions lists the different SMBus read transactions supported by the XL710. All the read transactions are compatible with SMBus read block protocol format.

Table 10-15. SMBus read transactions

TCO Command	Transaction	Command	Opcode	Fragments	Section
Receive TCO Packet	Block Read	0xD0 or 0xC0	First: 0x90 Middle: 0x10 Last[1]: 0x50	Multiple	10.5.11.2.1
Read Status	Block Read	0xD0 or 0xC0 or 0xDE	Single: 0xDD	Single	10.5.11.2.2
Get System MAC Address	Block Read	0xD4	Single: 0xD4	Single	10.5.11.2.3
Read Management Parameters	Block Read	0xD1	Single: 0xD1	Single	10.5.11.2.4
Read Management RCV Filter Parameters	Block Read	0xCD	Single: 0xCD	Single	10.5.11.2.5
Read Receive Enable Configuration	Block Read	0xDA	Single: 0xDA	Single	10.5.11.2.7.1
Get Controller Information	Block Read	0xD5	Single: 0xD5	Single	10.5.11.2.6
Get Common Filters	Block Read	0xD3	Single: 0xD3	Single	10.5.11.2.7

1. The last fragment of the receive TCO packet is the packet status.

0xC0 or 0xD0 commands are used for more than one payload. If the MC issues these read commands, and the XL710 has no pending data to transfer, it always returns as default opcode 0xDD with the XL710 status and does not NACK the transaction.

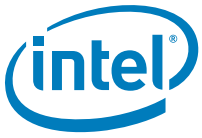
If an SMBus Quick Read command is received, it is handled as a XL710 Request Status command (see Section 10.5.11.1.2 for details).

10.5.11.2.1 Receive TCO LAN packet transaction

The MC uses this command to read packets received on the LAN and its status. When the XL710 has a packet to deliver to the MC, it asserts the SMBus notification for the MC to read the data (or direct receive). Upon receiving notification of the arrival of a LAN receive packet, the MC begins issuing a Receive TCO packet command using the block read protocol.

A packet can be transmitted to the MC in at least two fragments (at least one for the packet data and one for the packet status). As a result, the MC should follow the *F* and *L* bit of the opcode.

The opcode can have these values:



- 0x90 — First fragment
- 0x10 — Middle fragment
- When the opcode is 0x50, this indicates the last fragment of the packet, which contains packet status.

If a notification timeout is defined (in the NVM) and the MC does not finish reading the entire packet within the timeout period, since the packet has arrived, the packet is silently discarded. The time spent in ARA cycle or in reading the packet is not counted by the timeout counter.

Following is the receive TCO packet format and the data format returned from the XL710.

Function	Command
Receive TCO Packet	0xC0 or 0xD0

Function	Byte Count	Data 1 (Opcode)	Data 2	...	Data N
Receive TCO First Fragment	N	0x90	Packet Data Byte	...	Packet Data Byte
Receive TCO Middle Fragment		0x10			
Receive TCO Last Fragment	9 (0x9)	0x50	See Section 10.5.11.2.1.1		

10.5.11.2.1.1 Receive TCO LAN status payload transaction

This transaction is the last transaction that the XL710 issues when a packet received from the LAN is transferred to the MC. The transaction contains the status of the received packet.

The format of the status transaction is as follows:

Function	Byte Count	Data 1 (Opcode)	Data 2 – Data 17 (Status Data)
Receive TCO Long Status	9 (0x9)	0x50	See Below

The status is 8 bytes where byte 0 (bits 7:0) is set in Data 2 of the status and byte 7 in Data 9 of the status. Table 10-16 lists the content of the status data.

Table 10-16. TCO LAN packet status data

Name	Bits	Description
Packet Length	13:0	Packet length including CRC, only 14 LSB bits.
Reserved	15:14	Reserved.
Packet status	31:16	See Table 10-17.
VLAN	47:32	The two bytes of the VLAN header tag.
MNG status	63:48	See Table 10-19. This field should be ignored if Receive TCO is not enabled.

Table 10-17. Packet status info

Field	Bit(s)	Description
Reserved	15:4	Reserved.
LAN#	3:2	Indicates the source port of the packet.
VP	1	VLAN Stripped (indicates if the VLAN is part of the packet, or was removed).
CRC stripped	0	Insertion of CRC is needed.

Table 10-19. MNG status

Name	Bits	Description
Reserved	15:9	Reserved.
Decision Filter match	8	Set when there is a match to one of the decision filters.
Decision Filter index	7:4	Indicates which of the decision filters match the packet. (allows for up to 16 filters - although only 8 are currently supported).
MNG VLAN Address Match	3	Set when the manageability packet matches one of the manageability VLAN filters.
Pass MNG VLAN Filter Index	2:0	Indicates which of the VLAN filters match the packet.

10.5.11.2.2 Read status command

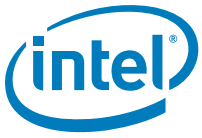
The MC should use this command after receiving a notification from the XL710 (such as SMBus alert). The XL710 also sends a notification to the MC in either of the following two cases:

- The MC asserts a request for reading the status.
- The XL710 detects a change in one of the Status Data 1 bits or NVM error bit in Data 2 (and was set to send status to the MC on status change) in the Receive Enable command.

Note: Commands 0xC0/0xD0 are for backward compatibility and can be used for other payloads. The XL710 defines these commands in the opcode as well as which payload this transaction is. When the 0XDE command is set, the XL710 always returns opcode 0XDD with the XL710 status. The MC reads the event causing the notification, using the Read Status command as follows.

The XL710's response to one of the commands (0xC0 or 0xD0) in a given time as defined in the SMBus Notification Timeout and Flags word in the NVM.

Function	Command
Read Status	0XC0 or 0XD0 or 0XDE



Function	Byte Count	Data 1 (Opcode)	Data 2 (Status Data 1)	Data 3 (Status Data 2)
Receive TCO Partial Status	3	0XDD	See Below	

This command can also be executed using the I^2C quick read format as follows:

1	7	1	1	8	1	8	1	8	1	1
Start	Slave Address	Rd	Ack	Byte Count	Ack	Status Data 1	Ack	Status Data 2	Ack	Stop
		1	0	0000 0002	0		0		1	

Table 10-20 lists the status data byte 1 parameters.

Table 10-20. Status data byte 1

Bit	Name	Description
7	LAN Port LSB	LAN port LSB together with LAN Port MSB define port that sent status. See further information in the description of LAN Port MSB (bit 2).
6	TCO Command Aborted	1b = A TCO command abort event occurred since the last read status cycle. 0b = A TCO command abort event did not occur since the last read status cycle.
5	Link Status Indication	0b = LAN link down. 1b = LAN link up.
4	PHY Link Forced Up	Contains the value of the *PHY_Link_Up* bit. When set, indicates that the PHY link is configured to keep the link up.
3	Initialization Indication	0b = An NVM reload event has not occurred since the last read status cycle. 1b = An NVM reload event has occurred since the last read status cycle[1].
2	LAN Port MSB	Defines together with LAN Port LSB the port that sent the status: Lan Port MSB / Lan Port LSB 0 / 0 — Status came from LAN port 0. 0 / 1 — Status came from LAN port 1. 1 / 0 — Status came from LAN port 2. 1 / 1 — Status came from LAN port 3.
1:0	Power State	00b = Dr state. 01b = D0u state. 10b = D0 state. 11b = D3 state. ***Note:*** When more than one function is mapped to the same port, the highest power state of the mapped functions is reported according to the following order: Dr < D3 < D0u < D0.

1. This indication is asserted when the XL710 the manageability block reloads the NVM and its internal database is updated to the NVM default values. This is an indication that the external MC should reconfigure the XL710, if other values other than the NVM default should be configured.

Status data byte 2 is used by the MC to indicate whether the LAN device driver is up and running.

The LAN device driver valid indication is a bit set by the LAN device driver during initialization; the bit is cleared when the LAN device driver enters a Dx state or is cleared by the hardware on a PCI reset.

Table 10-21 lists status data byte 2.

Table 10-21. Status data byte 2

Bit	Name	Description
7:6	Reserved	Reserved.
5	NVM Error	If set, indicates that a CRC/checksum error was detected in one of the manageability related NVM sections.
4	Reserved	Reserved.
3	Driver Valid Indication	0b = LAN driver is not up. 1b = LAN driver is up.
2:0	Reserved	Reserved.

10.5.11.2.3 Get system MAC address

The get system MAC address returns the system MAC address over to the SMBus. This command is a single-fragment read block transaction that returns the system MAC address.

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. When more than one function is defined on the port, it returns the address of the lowest defined function on this port.

Get system MAC address format:

Function	Command
Get system MAC address	0xD4

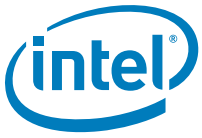
Data returned from the XL710:

Function	Byte Count	Data 1 (Opcode)	Data 2	...	Data 7
Get system MAC address	7	0xD4	MAC address MSB	...	MAC address LSB

10.5.11.2.4 Read management parameters

In order to read the management parameters, the MC should execute two SMBus transactions. The first transaction is a block write that sets the parameter that the MC wants to read. The second transaction is block read that reads the parameter.

Block write transaction:



Function	Command	Byte Count	Data 1
Management control request	0xC1	1	Parameter number

Following the block write, the MC should issue a block read that reads the parameter that was set in the Block Write command:

Function	Command
Read management parameter	0xD1

Data returned:

Function	Byte Count	Data 1 (Opcode)	Data 2	Data 3	...	Data N
Read management parameter	N	0xD1	Parameter number	Parameter dependent		

The returned data is in the same format of the MC command.

The returned data is as follow:

Parameter	#	Parameter Data
Keep PHY Link Up	0x00	A single byte parameter: Data 2 — Bit 0 = Set to indicate that the PHY link for this port should be kept up. Sets the keep_PHY_link_up bit. When cleared, clears the keep_PHY_link_up bit. Bit [7:1] = Reserved.
Wrong parameter request	0xFE	Returned by the XL710 only. This parameter is returned on a read transaction, if in the previous Read command the MC sets a parameter that is not supported by the XL710.
XL710 is not ready	0xFF	Returned by the XL710 only, on a Read Parameters command when the data that should have been read is not ready. This parameter has no data. The MC should retry the read transaction. This value is also returned if the byte count is illegal or if the read command is not preceded by a Write command.

The parameter that is returned might not be the parameter requested by the MC. The MC should verify the parameter number (default parameter to be returned is 0x1).

If the parameter number is 0xFF, it means that the data that was requested from the XL710 is not ready yet. The MC should retry the read transaction.

It is responsibility of the MC to follow the procedure previously defined. When the MC sends a Block Read command (as previously described) that is not preceded by a Block Write command with bytecount=1, the XL710 sets the parameter number in the read block transaction to be 0xFF.

10.5.11.2.5 Read management receive filter parameters

In order to read the management receive filter parameters, the MC should execute two SMBus transactions. The first transaction is a block write that sets the parameter that the MC wants to read. The second transaction is block read that read the parameter.

Block write transaction:

Function	Command	Byte Count	Data 1	Data 2
Update MNG RCV filter parameters	0xCC	1 or 2	Parameter number	Parameter data

The different parameters supported for this command are the same as the parameters supported for the update management receive filter parameters.

Following the block write the MC should issue a block read that reads the parameter that was set in the Block Write command:

Function	Command
Request MNG RCV filter parameters	0xCD

Data returned from the XL710:

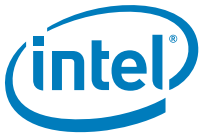
Function	Byte Count	Data 1 (Opcode)	Data 2	Data 3	...	Data N
Read MNG RCV filter parameters	N	0xCD	Parameter number	Parameter dependent		

The parameter that is returned might not be the parameter requested by the MC. The MC should verify the parameter number (default parameter to be returned is 0x1).

If the parameter number is 0xFF, it means that the data that was requested from the XL710 should supply is not ready yet. The MC should retry the read transaction.

It is MC's responsibility to follow the procedure previously defined. When the MC sends a Block Read command (as previously described) that is not preceded by a Block Write command with bytecount=1, the XL710 sets the parameter number in the read block transaction to be 0xFF.

Parameter	#	Parameter Data
Filters Enable	0x01	None.
MNGONLY Configuration	0x0F	None.
Flex Filter Enable Mask and Length	0x10	None.
Flex Filter Data	0x11	Data 2 — Group of Flex Filter's Bytes: 0x0 = Bytes 0-29. 0x1 = Bytes 30-59. 0x2 = Bytes 60-89. 0x3 = Bytes 90-119. 0x4 = Bytes 120-127.



Parameter	#	Parameter Data
Filters Valid	0x60	None.
Decision Filters	0x61	This command is obsolete. Please use 0x68 instead.
VLAN Filters	0x62	One byte to define the accessed VLAN tag filter (PRT_MNG_MAVTV). Data 2 — VLAN Filter number.
Flex Ports Filters	0x63	One byte to define the accessed manageability flex port filter (PRT_MNG_MFUTP). Data 2 — Flex Port Filter number.
IPv4 Filter	0x64	One byte to define the accessed IPv4 address filter (PRT_MNG_MIPAF4). Data 2 — IPv4 address filter number.
IPv6 Filters	0x65	One byte to define the accessed IPv6 address filter (PRT_MNG_MIPAF6). Data 2 — Pv6 address filter number.
MAC Filters	0x66	One byte to define the accessed MAC address filters pair (PRT_MNG_MMAL, PRT_MNG_MMAH). Data 2 — MAC address filters pair number (0-3).
EtherType Filters	0x67	1 byte to define Ethertype filters (PRT_MNG_METF). Data 2 — METF filter index (valid values are 0 - 3).
Extended Decision Filter	0x68	1 byte to define the extended decisions filters (PRT_MNG_MDEF_EXT & PRT_MNG_MDEF). Data 2 — MDEF filter index (valid values are 0 - 6).
Management Special Filter Modifiers	0x69	
Wrong parameter request	0xFE	Returned by the XL710 only. This parameter is returned on a read transaction, if in the previous Read command the MC sets a parameter that is not supported by the XL710.
XL710 is not ready	0xFF	Returned by the XL710 only on a Read Parameters command when the data that should have been read is not ready. This parameter has no data. This value is also returned if the byte count is illegal or if the Read command is not preceded by a Write command.

10.5.11.2.6 Get controller information command

The MC uses this command to get the controller identification. Each parameter is returned using a different parameter in the block write transaction.

In order to read the controller information, the MC should execute two SMBus transactions. The first transaction is a block write that sets the parameter that the MC wants to read. The second transaction is block read that read the parameter.

Block write transaction:

Function	Command	Byte Count	Data 1
Get Controller Information	0xD5	1	Parameter number

Following the block write, the MC should issue a block read that reads the parameter that was set in the Block Write command:

Function	Command
Get Controller Information	0xD5

Data returned from the XL710:

Function	Byte Count	Command	Data 2 (Op-Code)	Data 3 -n
Get Controller Information	Per Table 10-22	0xD5	Per Table 10-22	See Table 10-22 for the data for each opcode

Table 10-22. Get controller information data

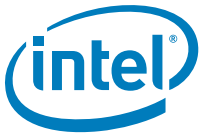
Parameter	Byte Count	Description	Notes
0x00	5	Data 4:3: Device ID. Data 5: Silicon Revision (RevID).	This is the hardware default value, not any value programmed via the NVM.
0x0B	4	Data 4:3 NVM Image version.	
0x0C	6	Data 6:3: Firmware ROM Internal version.	
0x0D	6	Data 6:3: Firmware Flash Internal version.	
0x0E	4	Data 4:3: PXE FW version.	MajorVersion.MinorVersion.Build.SubBuild. If a version is not found, a value of 0xFFFF is returned.
0x0F	4	Data 4:3: iSCSI FW version.	
0x10	4	Data 4:3: uEFI FW version.	
0x16	4	Data 4:3: FCoE Boot FW version.	
0xFE	2	Wrong parameter request.	Returned by the XL710 only. This parameter is returned on a read transaction, if in the previous Read command the MC sets a parameter that is not supported by the XL710.
0xFF	2	XL710 is not ready.	Returned by the XL710 only, on a Read Parameters command when the data that should have been read is not ready. This parameter has no data. The MC should retry the read transaction. This value is also returned if the byte count is illegal or if the Read command is not preceded by a Write command.

10.5.11.2.7 Get common filters command

The MC uses this command to get the common filters setting. This data can be configured when using Set Common Filters command. The first transaction is a block write that alerts that the MC wants to read the filters configuration. The second transaction is block read that read the configuration.

Block write transaction:

Function	Command	Byte count	Data
Get Common Filters	0xD3	1	0x00



Following the block write the MC should issue a block read that reads the filter settings:

Function	Command
Get Common filters	0xD3

Data returned from the XL710:

Function	Byte Count	Command	Data 1	Data 2:4	5:10	Data 11	Data 12	Data 13	Data 14:17
Get Common Filters	18	0xD3	0	Receive Control - see Table 10-14	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv4 Address

Function	Byte Count	Command	Data 1	Data 2:4	5:10	Data 11	Data 12	Data 13	Data 14:29
Get Common Filters	30	0xD3	0	Receive Control - see Table 10-14	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv6 Address

If an error occurs, the following answers might be returned:

Function	Command	Byte Count	Data 1
Get Common Filters	0xD3	1	0xFF

This response is by the XL710 on a Read Common Filter command when the data that should have been read is not ready. This parameter has no data. The MC should retry the read transaction.

This value is also returned if the byte count is illegal or if the Read command is not preceded by a Write command.

10.5.11.2.7.1 Read receive enable configuration

The MC uses this command to read the receive configuration data. This data can be configured when using the Receive Enable command or through the NVM.

The Read Receive Enable Configuration command format (SMBus read block) is as follows:

Function	Command
Read Receive Enable	0xDA

Data returned from the XL710:

Function	Byte Count	Data 1 (Opcode)	Data 2	Data 3	...	Data 8	Data 9	...	Data 12	Data 13	Data 14	Data 15
Read Receive Enable	15 (0x0F)	0xDA	Receive Control Byte	MAC Addr MSB	...	MAC Addr LSB	IP Addr MSB	...	IP Addr LSB	BMC SMBus Addr	I/F Data Byte	Alert Value Byte

The detailed description of each field is specified in the Receive Enable command description in Section 10.5.11.1.3.

10.5.12 Example configuration steps

This section provides sample configuration settings for common filtering configurations. Four examples are presented. The examples are in pseudo code format, with the name of the SMBus command followed by the parameters for that command and an explanation.

10.5.12.1 Example 1 - shared MAC, RMCP only ports

This example is the most basic configuration. The MAC address filtering is shared with the host operating system and only traffic directed the RMCP ports (0x26F and 0x298) is filtered. For this example, the MC must issue gratuitous ARPs because no filter is enabled to pass ARP requests to the MC.

10.5.12.1.1 Example 1 pseudo code

Step 1: Disable existing filtering

`Receive Enable[00]`

Using the basic form of the Receive Enable command, this prevents any packets from reaching the MC by disabling filtering:

Receive Enable Control 0x00:

- Bit 0 [0] – Disable Receiving of packets

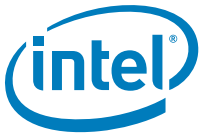
Step 2: Configure MDEF[0]

`Update Manageability Filter Parameters [68, 0, C0000000, 00000000]`

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x68). This updates MDEF[0], as indicated by the second parameter (0).

MDEF[0] value of 0xC0000000:

- Bit 30 [1] – port 0x298



- Bit 31 [1] – port 0x26F

  MDEF_EXT[0] value of 0x0000000:

Step 3: Configure *MNGONLY*

`Update Manageability Filter Parameters [F, 0, 00000001]`

Use the Update Manageability Filter Parameters command to update Manageability Only (*MNGONLY*) (parameter 0xF) so that port 0x298 and 0x26F would not be sent to the host.

- Bit [0] - *MDEF[0]* is exclusive to the MC.

Step 4: - Enable Filtering

`Receive Enable [05]`

Using the basic form of the Receive Enable command:

Receive Enable Control 0x05:

- Bit 0 [1] – Enable receiving of packets
- Bit 2 [1] – Enable status reporting (such as link lost)
- Bit 5:4 [00] – Notification method = SMB alert
- Bit 7 [0] – Use shared MAC

The resulting MDEF filters are as follows:

Table 10-23. Example 1 MDEF results

		Manageability Decision Filter (MDEF)							
Filter		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
L2 Unicast Address[3:0]	AND								
Broadcast	AND								
Manageability VLAN[7:0]	AND								
IPv6 Address[3:0]	AND								
IPv4 Address[3:0]	AND								
L2 Unicast Address[3:0]	OR								
Broadcast	OR								
Multicast	AND								
ARP Request	OR								
ARP Response	OR								
Neighbor Discovery	OR								
Port 0x298	OR	X							
Filter		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Port 0x26F	OR	X							
Flex Port 7:0	OR								
Flex TCO	OR								

10.5.12.2 Example 2 - dedicated MAC, auto ARP response and RMCP port filtering

This example shows a common configuration; the MC has a dedicated MAC and IP address. Automatic ARP responses are enabled as well as RMCP port filtering. By enabling automatic ARP responses, the MC is not required to send the gratuitous ARPs as it did in Example 1.

For demonstration purposes, the dedicated MAC address is calculated by reading the system MAC address and adding a one to it; assume the system MAC is AABBCCDC. The IP address for this example is 1.2.3.4. Additionally, the XSUM filtering is enabled.

Note that not all Intel Ethernet controllers support automatic ARP responses, please refer to product specific documentation.

10.5.12.2.1 Example 2 - pseudo code

Step 1: Disable existing filtering

`Receive Enable[00]`

Using the basic form of the Receive Enable command, this prevents any packets from reaching the MC by disabling filtering:

Receive Enable Control 0x00:

- Bit 0 [0] – Disable Receiving of packets

Step 2: Read System MAC Address

`Get System MAC Address []`

Reads the system MAC address. Assume a returned AABBCCDC for this example.

Step 3: Configure XSUM Filter

`Update Manageability Filter Parameters [01, 00800000]`

Use the Update Manageability Filter Parameters command to update Filters Enable settings (parameter 1). This sets the Manageability Control (*PRT_MNG_MANC*) register.

PRT_MNG_MANC Register 0x00800000:

- Bit 23 [1] - XSUM Filter Enable

Note that some of the following configuration steps manipulate the *PRT_MNG_MANC* register indirectly, this command sets all bits except XSUM to 0b. It is important to either do this step before the others, or to read the value of the *PRT_MNG_MANC* and then write it back with only bit 32 changed. Also note that the XSUM enable bit might differ between Ethernet controllers, refer to product specific documentation.

Step 4: Configure MDEF[0]

`Update Manageability Filter Parameters [68, 0, C0000000, 00000000]`

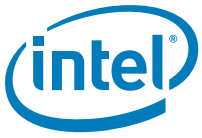
Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x68). This updates MDEF[0], as indicated by the second parameter (0).

MDEF value of 0x00000C00:

- Bit 30 [1] – port 0x298
- Bit 31 [1] – port 0x26F

MDEF_EXT[0] value of 0x0000000:

Step 5: Configure MDEF[1]



Update Manageability Filter Parameters [68, 1, 10000000, 00000000]

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x61). This updates MDEF[1], as indicated by the second parameter (1).

MDEF value of 0x10000000:

- Bit 28 [1] – ARP Requests

  MDEF_EXT[1] value of 0x0000000:

  When enabling automatic ARP responses, the ARP requests still go into the manageability filtering system and as such need to be designated as also needing to be sent to the host. For this reason a separate MDEF is created with only ARP request filtering enabled.

  Refer to the next step for more details.

Step 6: Configure Manageability only

Update Manageability Filter Parameters [F, 0, 00000001]

Use the Update Manageability Filter Parameters command to update Manageability Only (MNGONLY) (parameter 0xF) so that port 0x298 and 0x26F would not be sent to the host.

- Bit [0] - MDEF[0] is exclusive to the MC.

  This enables ARP requests to be passed to both manageability and to the host. Specified separate MDEF filter for ARP requests. If ARP requests had been added to *MDEF[0]* and then *MDEF[0]* specified in management only configuration then not only would RMCP traffic (ports 0x26F and 0x298) be sent only to the MC, ARP requests would have also been sent to the MC only.

Step 7: Enable Filtering

Receive Enable [8D, AABBCCDD, 01020304, 00, 00, 00]

Using the advanced version Receive Enable command, the first parameter:

Receive Enable Control 0x8D:

- Bit 0 [1] – Enable receiving of packets
- Bit 2 [1] – Enable status reporting (such as link lost)
- Bit 3 [1] – Enable automatic ARP responses
- Bit 5:4 [00] – Notification method = SMB alert
- Bit 7 [1] - Use dedicated MAC

  Second parameter is the MAC address (AABBCCDD).

  Third parameter is the IP address(01020304).

The last three parameters are zero when the notification method is SMB alert.

The resulting MDEF filters are as follows:

Table 10-24. Example 2 MDEF results

		Manageability Decision Filter (MDEF)							
Filter		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
L2 Unicast Address[3:0]	AND								
Broadcast	AND								
Manageability VLAN[7:0]	AND								
IPv6 Address[3:0]	AND								
IPv4 Address[3:0]	AND								
L2 Unicast Address[3:0]	OR								

Table 10-24. Example 2 MDEF results

		Manageability Decision Filter (MDEF)							
Broadcast	OR								
Multicast	AND								
ARP Request	OR		X						
ARP Response	OR								
Neighbor Discovery	OR								
Port 0x298	OR	X							
Port 0x26F	OR	X							
Flex Port 7:0	OR								
Flex TCO	OR								

10.5.12.3 Example 3 - dedicated MAC and IP address

This example provided, the MC with a dedicated MAC and IP address enables it to receive ARP requests. The MC is then responsible for responding to ARP requests.

For demonstration purposes, the dedicated MAC address is calculated by reading the system MAC address and adding a one to it; assume the system MAC is AABBCCDC. The IP address for this example is 1.2.3.4. For this example, the Receive Enable command is used to configure the MAC address filter.

In order for the MC to be able to receive ARP requests, it needs to specify a filter for this, and that filter needs to be included in the manageability-to-host filtering so that the host operating system can also receive ARP requests.

10.5.12.3.1 Example 3 - pseudo code

Step 1: Disable existing filtering

`Receive Enable[00]`

Using the basic form of the Receive Enable command, this prevents any packets from reaching the MC by disabling filtering:

Receive Enable Control 0x00:

- Bit 0 [0] – Disable receiving of packets

Step 2: Read System MAC Address

`Get System MAC Address []`

Reads the system MAC address. Assume a returned AABBCCDC for this example.

Step 3: Configure IP Address Filter

`Update Manageability Filter Parameters [64, 00, 01020304]`

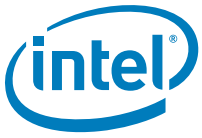
Use the update manageability filter parameters to configure an IPv4 filter.

The first parameter (0x64) specifies that we are configuring an IPv4 filter.

The second parameter (0x00) indicates which IPv4 filter is being configured; in this case filter 0.

The third parameter is the IP address – 1.2.3.4.

Step 4: Configure MAC Address Filter



`Update Manageability Filter Parameters [66, 00, AABBCCDD]`

Use the update manageability filter parameters to configure a MAC address filter.

The first parameter (0x66) specifies that we are configuring a MAC address filter.

The second parameter (0x00) indicates which MAC address filter is being configured; in this case filter 0.

The third parameter is the MAC address - AABBCCDD

Step 5: Configure MDEF[0] for IP and MAC Filtering

`Update Manageability Filter Parameters [68, 0, 00002001, 00000000]`

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x68). This updates MDEF[0], as indicated by the second parameter (0).

MDEF value of 00002001:

- Bit 0 [1] – MAC[0] address filtering
- Bit 13 [1] – IP[0] address filtering

MDEF_EXT[0] value of 0x0000000:

Step 6: Configure MDEF[1]

`Update Manageability Filter Parameters [68, 1, 10000000]`

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x68). This updates MDEF[1], as indicated by the second parameter (1).

MDEF value of 10000000:

- Bit 28 [1] – ARP requests

MDEF_EXT[1] value of 0x0000000:

Step 7: Configure the Management to Host Filter

`Update Manageability Filter Parameters [F, 0, 00000001]`

Use the Update Manageability Filter Parameters command to update Manageability Only (*MNGONLY*) (parameter 0xF) so that the dedicated MAC/IP traffic would not be sent to the host. Note that given the host does not program this address in it's L2 filtering, this step is not a must, unless the host chooses to work in promiscuous mode.

- Bit [0] - MDEF[0] is exclusive to the MC.

Step 8: Enable Filtering

`Receive Enable [05]`

Using the basic form of the Receive Enable command,:

Receive Enable Control 0x05:

- Bit 0 [1] – Enable receiving of packets
- Bit 2 [1] – Enable status reporting (such as link lost)
- Bit 5:4 [00] – Notification method = SMB alert

The resulting MDEF filters are as follows:

Table 10-25. Example 3 MDEF results

		Manageability Decision Filter (MDEF)							
Filter		0	1	2	3	4	5	6	7
L2 Unicast Address[3:0]	AND	0001							
Broadcast	AND								
Manageability VLAN[7:0]	AND								
IPv6 Address[3:0]	AND								
IPv4 Address[3:0]	AND	0001							
L2 Unicast Address[3:0]	OR								
Broadcast	OR								
Multicast	AND								
ARP Request	OR		X						
ARP Response	OR								
Neighbor Discovery	OR								
Port 0x298	OR								
Port 0x26F	OR								
Flex Port 7:0	OR								
Flex TCO	OR								

10.5.12.4 Example 4 - dedicated MAC and VLAN tag

This example shows an alternate configuration; the MC has a dedicated MAC and IP address, along with a VLAN tag of 0x32 are required for traffic to be sent to the MC. This means that all traffic with a VLAN and matching tag is sent to the MC.

For demonstration purposes, the dedicated MAC address is calculated by reading the system MAC address and adding a one to it; assume the system MAC is AABBCCDC. The IP address for this example is 1.2.3.4 and the VLAN tag is 0x0032.

Additionally, the XSUM filtering is enabled.

10.5.12.4.1 Example 4 - pseudo code

Step 1: Disable existing filtering

`Receive Enable[00]`

Using the basic form of the Receive Enable command, this prevents any packets from reaching the MC by disabling filtering:

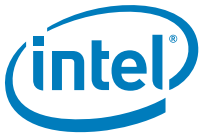
Receive Enable Control 0x00:

- Bit 0 [0] – Disable receiving of packets

Step 2: - Read System MAC Address

`Get System MAC Address []`

Reads the system MAC address. Assume a returned AABBCCDC for this example.



Step 3: Configure XSUM Filter

`Update Manageability Filter Parameters [01, 00800000]`

Use the Update Manageability Filter Parameters command to update filters enable settings (parameter 1). This sets the Manageability Control (*PRT_MNG_MANC*) register.

PRT_MNG_MANC Register 0x00800000:

- Bit 23 [1] – XSUM filter enable

Note that some of the following configuration steps manipulate the *PRT_MNG_MANC* register indirectly. This command sets all bits except XSUM to 0b. It is important to either do this step before the others, or to read the value of the *PRT_MNG_MANC* and then write it back with only bit 32 changed. Also note that the XSUM enable bit might differ between Ethernet controllers, refer to product specific documentation.

Step 4: Configure VLAN 0 Filter

`Update Manageability Filter Parameters [62, 0, 0032]`

Use the Update Manageability Filter Parameters command to configure VLAN filters. Parameter 0x62 indicates an update to the VLAN filter. The second parameter indicates which VLAN filter (0 in this case). The last parameter is the VLAN ID (0x0032).

Step 5: Configure MDEF[0]

`Update Manageability Filter Parameters [68, 0, 00000020, 00000000]`

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x68). This updates MDEF[0], as indicated by the second parameter (0).

MDEF value of 0x00000020:

- Bit 5 [1] – VLAN[0] AND

MDEF_EXT[0] value of 0x0000000:

Step 6: Enable Filtering

`Receive Enable [85, AABBCCDD, 01020304, 00, 00, 00]`

Using the advanced version Receive Enable command, the first parameter:

Receive Enable Control 0x85:

- Bit 0 [1] – Enable receiving of packets
- Bit 2 [1] – Enable status reporting (such as link lost)
- Bit 5:4 [00] – Notification method = SMB alert
- Bit 7 [1] – Use dedicated MAC

Second parameter is the MAC address: AABBCCDD.

The third parameter is the IP address: 01020304.

The last three parameters are zero when the notification method is SMBus alert.

The resulting MDEF filters are as follows:

Table 10-26. Example 4 MDEF results

		Manageability Decision Filter (MDEF)							
Filter		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
L2 Unicast Address[3:0]	AND								0001
Broadcast	AND								

Table 10-26. Example 4 MDEF results

		Manageability Decision Filter (MDEF)							
Manageability VLAN[7:0]	AND	X							
IPv6 Address[3:0]	AND								
IPv4 Address[3:0]	AND								
L2 Unicast Address[3:0]	OR								
Broadcast	OR								
Multicast	AND								
ARP Request	OR								
ARP Response	OR								
Neighbor Discovery	OR								
Port 0x298	OR								
Port 0x26F	OR								
Flex Port 7:0	OR								
Flex TCO	OR								

10.5.13 SMBus troubleshooting

This section outlines the most common issues found while working with PT using the SMBus sideband interface.

10.5.13.1 TCO alert line stays asserted after a power cycle

After the XL710 resets, all its ports indicates 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 (see Section 10.5.10) 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.

10.5.13.2 When SMBus commands are always NACK'd

There are several reasons why all commands sent to the XL710 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 XL710 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 XL710 might be unstable.

10.5.13.3 SMBus clock speed is 16.6666 KHz

This can happen when the SMBus connecting the MC and the XL710 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 XL710 and the MC to this device.

10.5.13.4 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 sytem (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.

10.5.13.5 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 XL710 when it attempts to write a packet, the problem is most likely due to the fact that the buffers internal to the XL710 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 XL710 has limited buffers that are shared for receiving and transmitting data. If a MC does not keep the incoming data read, the XL710 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 XL710, and then use the Receive Enable command to enable incoming data.

10.5.13.6 SMBus fragment size

The SMBus specification indicates a maximum SMBus transaction size of 32 bytes. Most of the data passed between the XL710 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 XL710 to the MC or to send a data from the MC to the XL710.

Recognizing this bottleneck, the XL710 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 XL710 allocates buffers based upon the SMBus fragment size setting within the NVM. XL710 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, the XL710 firmware has 3 KB of RAM it can use for buffering SMBus fragments. If the SMBus fragment size is 32 bytes then 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, the XL710 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 XL710 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 XL710 NACKs the SMBus transaction due to not enough memory to store the KVM data.

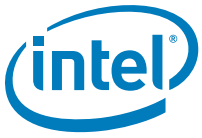
In summary, if a programmer increases the size of the SMBus fragment size in the NVM (recommended for efficiency purposes) take care to ensure that the MC implementation reflects this change and uses that fragment size to its fullest when sending SMBus fragments.

10.5.13.7 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. In order to avoid this possible situation, the MC can use the Management Control command detailed in Section 10.5.11.1.5 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.

10.5.13.8 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.

10.5.13.9 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 XL710, rather than the NVM image provided by Intel. Parts of the NVM image are changed after writing (such as the physical NVM size).

10.6 Network Controller Sideband Interface (NC-SI) PT interface

The NC-SI is a DMTF industry standard protocol for the sideband interface. NC-SI uses a modified version of the industry standard RMII interface for the physical layer as well as defining a new logical layer.

The NC-SI specification can be found at:

http://www.dmtf.org/

10.6.1 Overview

10.6.1.1 Terminology

The terminology in this section is taken from the NC-SI specification.

Table 10-27. NC-SI terminology

Term	Definition
Frame Versus Packet	Frame is used in reference to Ethernet, whereas packet is used everywhere else.
External Network Interface	The interface of the network controller that provides connectivity to the external network infrastructure (port).
Internal Host Interface	The interface of the network controller that provides connectivity to the host OS running on the platform.
Management Controller (BMC)	An intelligent entity comprising of HW/FW/SW, that resides within a platform and is responsible for some or all management functions associated with the platform (BMC, service processor, etc.).
Network Controller (NC)	The component within a system that is responsible for providing connectivity to the external Ethernet network world.
Remote Media	The capability to allow remote media devices to appear as if they were attached locally to the host.
Network Controller Sideband Interface	The interface of the network controller that provides connectivity to a management controller. It can be shorten to sideband interface as appropriate in the context.
Interface	This refers to the entire physical interface, such as both the transmit and receive interface between the management controller and the network controller.
Integrated Controller	The term integrated controller refers to a network controller device that supports two or more channels for NC-SI that share a common NC-SI physical interface. For example, a network controller that has two or more physical network ports and a single NC-SI bus connection.
Multi-Drop	Multi-drop commonly refers to the case where multiple physical communication devices share an electrically common bus and a single device acts as the 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.
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.
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.
Package	One or more NC-SI channels in a network controller that share a common set of electrical buffers and common buffer control for the NC-SI bus. Typically, there will be a single, logical NC-SI package for a single physical network controller package (chip or module). However, the specification allows a single physical chip or module to hold multiple NC-SI logical packages.
Control Traffic/Messages/Packets	Command, response and notification packets transmitted between BMC and the XL710 for the purpose of managing NC-SI.
Pass-Through Traffic/Messages/ Packets	Non-control packets passed between the external network and the BMC through the XL710.
Channel Arbitration	Refer to operations where more than one of the network controller channels can be enabled to transmit pass-through packets to the BMC at the same time, where arbitration of access to the RXD, CRS_DV, and RX_ER signal lines is accomplished either by software of hardware means.
Logically Enabled/Disabled NC	Refers to the state of the network controller wherein pass-through traffic is able/unable to flow through the sideband interface to and from the management controller, as a result of issuing Enable/Disable Channel command.
NC RX	Defined as the direction of ingress traffic on the external network controller interface

Table 10-27. NC-SI terminology

Term	Definition
NC TX	Defined as the direction of egress traffic on the external network controller interface
NC-SI RX	Defined as the direction of ingress traffic on the sideband enhanced NC-SI Interface with respect to the network controller.
NC-SI TX	Defined as the direction of egress traffic on the sideband enhanced NC-SI Interface with respect to the network controller.

10.6.1.2 System topology

In NC-SI each physical endpoint (NC package) can have several logical 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 10-6 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.

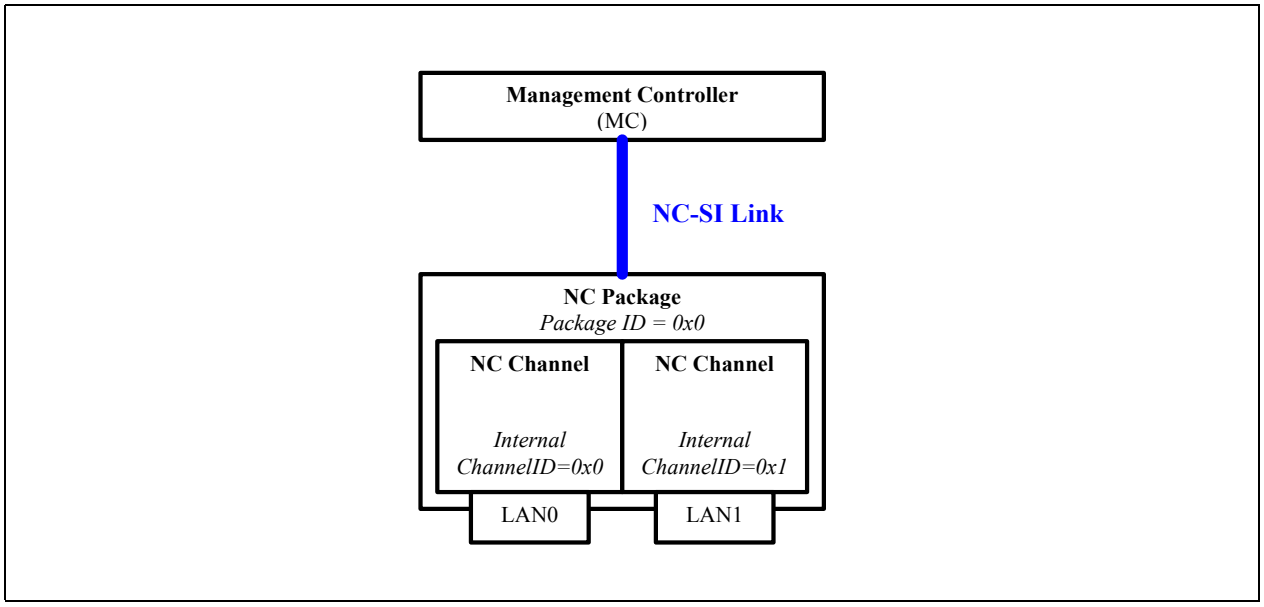


Figure 10-6. Single NC package, two NC channels

Figure 10-7 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 10-7) mandate an arbitration mechanism.

The arbitration mechanism is described in Section 10.6.9.1.

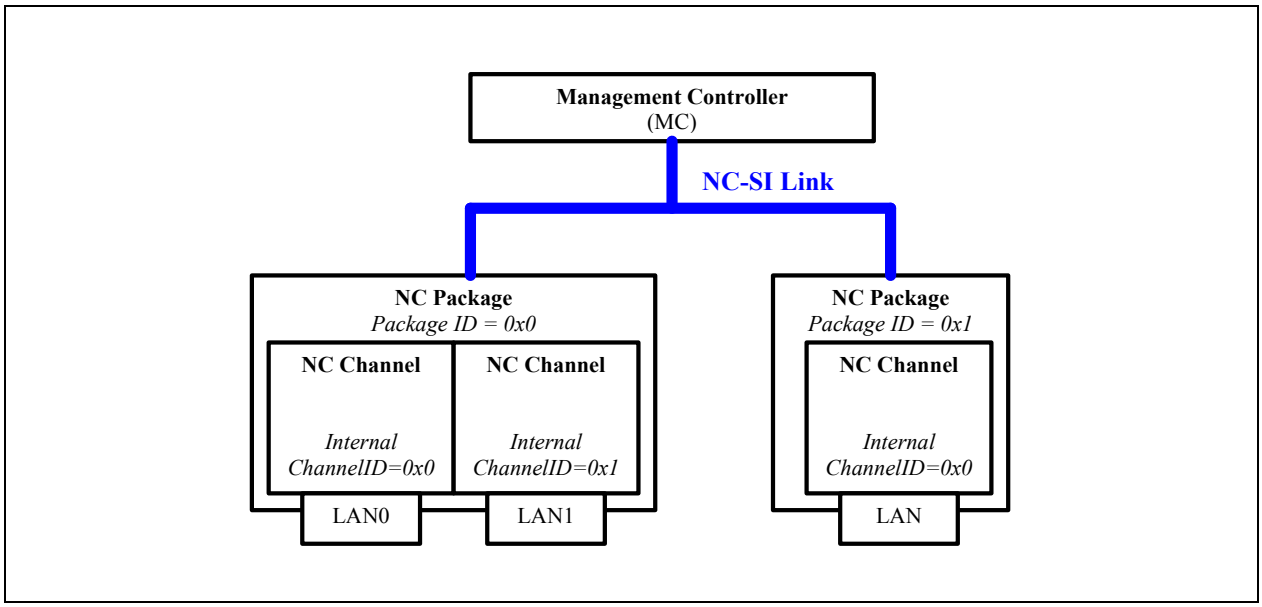


Figure 10-7. Two NC packages (left, with two NC channels and right, with one NC channel)

Note: Channel numbers should match PCI function numbers. If more than one function is defined on a port, the function with the lowest value associated with this port is used. See Section 10.2.2.2.2 for details.

10.6.1.3 Data transport

Since NC-SI is based upon the RMII transport layer, data is transferred in the form of Ethernet frames.

NC-SI defines two types of transmitted frames:

1. Control frames:
   a. Configures and control the interface.
   b. Identified by a unique EtherType in their L2 header.
2. PT frames:
   a. Actual LAN pass-through frames transferred from/to the MC.
   b. Identified as not being a control frame.
   c. Attributed to a specific NC channel by their source MAC address (as configured in the NC by the MC).

10.6.1.3.1 Control frames

NC-SI control frames are identified by a unique NC-SI EtherType (0x88F8).

Control frames are used in a single-threaded operation, meaning commands are generated only by the MC and can only be sent one at a time. Each command from the MC is followed by a single response from the NC (command-response flow), after which the MC is allowed to send a new command.

The only exception to the command-response flow is the Asynchronous Event Notification (AEN). These control frames are sent unsolicited from the NC to the MC.

AEN functionality by the NC must be disabled by default, until activated by the MC using the Enable AEN commands.

In order to be considered a valid command, a control frame must:

1. Comply with the NC-SI header format.
2. Be targeted to a valid channel in the package via the *Package ID* and *Channel ID* fields. For example, to target a NC channel with package ID of 0x2 and internal channel ID of 0x5, the MC must set the channel ID inside the control frame to 0x45. The channel ID is composed of three bits of package ID and five bits of internal channel ID.
3. Contain a correct payload checksum (if used).
4. Meet any other condition defined by NC-SI.

There are also commands (such as select package) targeted to the package as a whole. These commands must use an internal channel ID of 0x1F.

For details, refer to the NC-SI specification.

10.6.1.3.2 NC-SI frames receive flow

Figure 10-8 shows the flow for frames received on the NC from the MC.

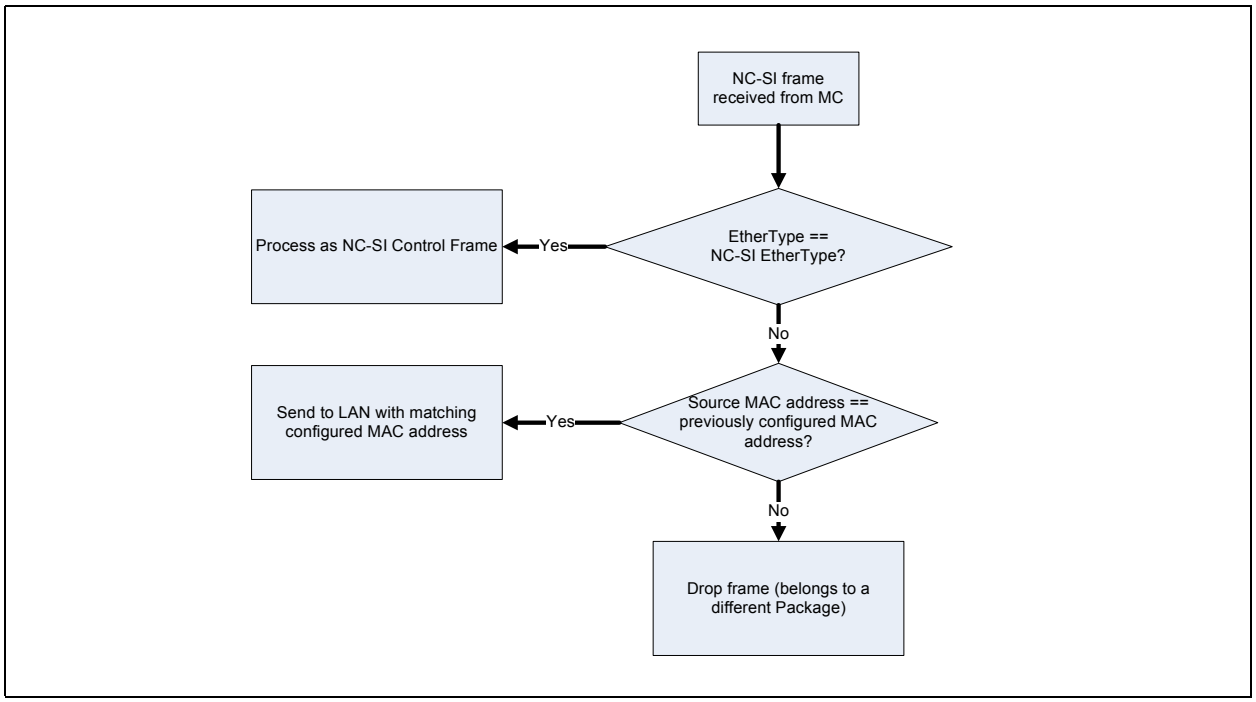


Figure 10-8. NC-SI frames receive flow for the NC

10.6.2 NC-SI standard support

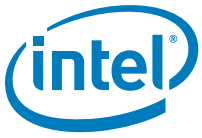
10.6.2.1 Supported features

The XL710 supports all the mandatory features of the NC-SI specification (rev 1.0.1). Table 10-28 lists the supported commands.

Table 10-29 lists optional features supported and the level of support for partially supported commands.

Table 10-28. Supported NC-SI commands

Command	Supported over RMII	Supported over MCTP with Pass Through	Supported over MCTP without Pass Through
Clear initial state	Yes	Yes	Yes
Get Version ID	Yes	Yes	Yes
Get Parameters	Yes	Yes	Yes
Get Controller Packet Statistics	Yes, partially	Yes, partially	Yes, partially
Get Link Status	Yes	Yes	Yes
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
OEM Command[5]	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 XL710 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. See Section 10.6.3.2 for details.

Table 10-29. Optional NC-SI features support

Feature	Implement	Details
AENs	Yes	The Driver state AEN might be emitted up to 1 minute after actual driver change if the driver was taken down unexpectedly. When more than one function is associated with a channel, the driver status is enabled if at least one driver is up.
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.
Get NC-SI statistics	Yes	Support all the counters[3]
Get NC-SI Pass-Through Statistics	Yes, partially	Support the following counters: 1, 6, 7.
VLAN Modes	Yes, partially	Support only modes 1, 3.
Buffering Capabilities	Yes	8 Kb
MAC Address Filters	Yes	Supports 2 MAC addresses per port.
Channel Count	Yes	Supports 4 channels.
VLAN Filters	Yes	Support 8 VLAN filters per port. Filtering is ignoring the *CFI* bit and the 802.1P priority bits
Broadcast Filters	Yes	Support the following filters: ARP DHCP Net BIOS
Multicast Filters	Yes	Supports the following filters: IPv6 neighbor advertisement IPv6 router advertisement DHCPv6 relay and server multicast
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 XL710 does not increment the NC-SI control packets dropped counter when packets with checksum errors are dropped. In this case, only the NC-SI command checksum errors counter is updated.

10.6.2.2 AEN handling

Asynchronous events might occur when the device is not allowed to send them. The following rules defines the behavior of the XL710 in these cases:

1. While the device is disabled, for each type of AEN only the last event is kept.
2. Outstanding AENs that occurred while a package was deselected is transmitted when a package is selected.
3. On a transition from channel disabled to channel enabled, all outstanding events are erased to prevent stale event notifications.
4. If the AEN becomes outdated before being sent (for example a link down, link up sequence occurring before the AEN is sent), then no AEN is sent.

Table 10-30. Get NC-SI PT statistics generation

Counter Number	Statistic Name	Source
1	Total PT Tx Packets Received	Sum of GLV_UPTCL, GLV_MPTCL and GLV_BPTCL for relevant VSI.
6	Total PT Rx Packets	Sum of GLV_UPRCL, GLV_MPRCL and GLV_BPRCT
7	Total PT Rx Packets Dropped	GLV_RDPC.

10.6.3 External link control via NC-SI

10.6.3.1 NC-SI link state control

In NC-SI mode, the device might dynamically change the PHY power mode according to the NC-SI channel state assuming no other functionality requires the PHY to be active (host or wake up).

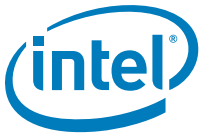
The following algorithm is used to define if PHY activity is required:

- At initialization time, the PHY is required to be active only if the *EMP_LINK_ON* bit in Common Firmware Parameters 2 NVM word is set.
- Once a channel is enabled via a Enable Channel NC-SI command, The PHY is powered up.
- If the channel is disabled via a Disable Channel command with the *ALD* bit set, the PHY is disabled.
- If the channel is disabled via a Reset Channel command, the PHY power state is set back to the initial value as define by the *EMP_LINK_ON* bit.

Note: Before transitioning to D3 it is the responsibility of the software device driver to request the PHY to be active for wake-up activities.

10.6.3.2 Set link error codes

The following rules are used to define the error code returned for a Set Link command in case an invalid configuration is requested:



1. Host Driver Check: If a host device driver is present, return a Command 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.

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).

10.6.4 NC-SI mode — Intel specific commands

In addition to regular NC-SI commands, the following Intel vendor specific commands are supported. The purpose of these commands is to provide a means for the MC to access some of the Intel-specific features present in the XL710.

10.6.4.1 Overview

The following features are available via the NC-SI OEM specific commands:

- Receive filters:
- Packet addition decision filters 0x0…0x4

- Packet reduction decision rilters 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 XL710.
- 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. This command is as follows.

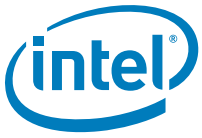
10.6.4.1.1 OEM command (0x50)

The OEM command can be used by the MC to request the sideband interface to provide vendor-specific information. The Vendor Enterprise Number (VEN) is the unique MIB/SNMP private enterprise number assigned by IANA per organization. Vendors are free to define their own internal data structures in the vendor data fields.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...	Intel Command Number	Optional Data		
...	...			
...	Optional Data		Padding to 32 bits (0x00)	
...	Checksum			

10.6.4.1.2 OEM response (0xD0)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	



	Bits	
20...23	Manufacturer ID (Intel 0x157)	
24...27	Intel Command Number	Optional Return Data
...	...	
...	Optional Return Data	Padding to 32 bits (0x00)
...	Checksum	

Note: Responses have no command-specific reason code, unless otherwise specified within the command.

Note: The commands/responses described as follows includes only the part up to the data. The padding and checksum are implied.

10.6.4.2 OEM commands summary

Table 10-31. 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 is shared mode. Returned when shared mode commands are used while not in shared MAC/IP mode.
		0x008E	Returned when a request to disable a VLAN or a MAC address used in a active binding is received.

Table 10-32. OEM commands summary

Intel Command	Parameter	Command Name	Supported in MCTP without PT	Section
0x00	0x00	Set IP Filters Control	No	10.6.4.3
0x01	0x00	Get IP Filters Control	No	10.6.4.4

Table 10-32. OEM commands summary

Intel Command	Parameter	Command Name	Supported in MCTP without PT	Section
0x02	0x0F	Set Manageability Only	No	10.6.4.5.3
	0x10	Set Flexible 128 Filter Mask and Length		10.6.4.5.5
	0x11	Set Flexible 128 Filter Data		10.6.4.5.7
	0x63	Set Flex TCP/UDP Port Filters		10.6.4.5.10
	0x64	Set Flex IPv4 Address Filters		10.6.4.5.14
	0x65	Set Flex IPv6 Address Filters		10.6.4.5.16
	0x67	Set EtherType Filter		10.6.4.5.18
	0x68	Set Packet Addition Extended Filter		10.6.4.5.20
	0x69	Set Special Filter Modifiers		10.6.4.5.22
0x03	0x0F	Get Manageability Only	No	10.6.4.6.3
	0x10	Get Flexible 128 Filter Mask and Length		10.6.4.6.5
	0x11	Get Flexible 128 Filter Data		10.6.4.6.7
	0x63	Get Flex TCP/UDP Port Filters		10.6.4.6.10
	0x64	Get Flex IPv4 Address Filters		10.6.4.6.13
	0x65	Get Flex IPv6 Address Filters		10.6.4.6.15
	0x67	Get EtherType Filter		10.6.4.6.17
	0x68	Get Packet Addition Extended Filter		10.6.4.6.19
	0x69	Get Special Filter Modifiers		10.6.4.6.21
0x04	0x10	Set Extended Unicast Packet Reduction	No	10.6.4.7.3
	0x11	Set Extended Multicast Packet Reduction		10.6.4.7.5
	0x12	Set Extended Broadcast Packet Reduction		10.6.4.7.7
0x5	0x10	Get Extended Unicast Packet Reduction	No	10.6.4.8.1
	0x11	Get Extended Multicast Packet Reduction		10.6.4.8.3
	0x12	Get Extended Broadcast Packet Reduction		10.6.4.8.5
0x06	N/A	Get System MAC Address	Yes	10.6.4.9
0x20	N/A	Set Intel Management Control	No	10.6.4.10
0x21	N/A	Get Intel Management Control	No	10.6.4.11
0x22	N/A	TCO Reset	Yes	10.6.4.12
0x23	N/A	Enable IP/UDP/TCP Checksum Offloading	No	10.6.4.13.1
0x24	N/A	Disable IP/UDP/TCP Checksum Offloading	No	10.6.4.13.3

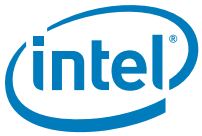


Table 10-32. OEM commands summary

Intel Command	Parameter	Command Name	Supported in MCTP without PT	Section
0x25	0x0	Set IP Address	No	10.6.4.14.1
	0x1	Get IP Address		10.6.4.14.2
	0x2	Set Port		10.6.4.14.3
	0x3	Get Port		10.6.4.14.4
	0x4	Enable Unicast Infrastructure Filter		10.6.4.14.5
	0x5	Get Shared IP Capabilities Command		10.6.4.14.6
	0x6	Shared IP Enable Broadcast filtering		10.6.4.14.7
	0x7	Shared IP Enable Global Multicast filtering		10.6.4.14.8
	0x8	Get Shared IP parameters		10.6.4.14.9
	0x9	Set Binding		10.6.4.14.10
	0xA	Get Binding		10.6.4.14.11
	0xB	Set Shared Mode		10.6.4.14.12
0x40	0x01	Enable OS2BMC flow	No	10.6.4.15.1
	0x02	Enable Network to BMC flow		10.6.4.15.3
	0x03	Enable Both Network to BMC and Host to BMC flow		10.6.4.15.5
0x40	0x04	Set BMC IP address	No	10.6.4.15.7
0x41	N/A	Get OS2BMC parameters	No	10.6.4.15.9
0x48	0x1	Get Controller Information	Yes	10.6.4.16

Note: All the commands are supported both over RMII NC-SI and over MCTP.

10.6.4.3 Set Intel filters control — IP filters control command (Intel command 0x00, filter control index 0x00)

This command controls different aspects of the Intel filters.

10.6.4.3.1 Set Intel filters control — IP filters control command

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x00	0x00	IP Filters control (3-2)	
24...27	IP Filters Control (1-0)			

Where IP Filters Control has the following format.

Bit #	Name	Description	Default Value
0	IPv4/IPv6 Mode	IPv6 (0b) = There are zero IPv4 filters and four IPv6 filters. IPv4 (1b) = There are four IPv4 filters and four IPv6 filters.	1b
1...31	Reserved		

Note: This command is kept for compatibility with other projects and has no effect in XL710.

10.6.4.3.2 Set Intel filters control — IP filters control response

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x00	0x00		

10.6.4.4 Get Intel filters control commands (Intel command 0x01)

10.6.4.4.1 Get Intel filters control — IP filters control command (Intel command 0x01, filter control index 0x00)

This command controls different aspects of the Intel filters.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x01	0x00		

10.6.4.4.2 Get Intel filters control — IP filters control response (Intel command 0x01, filter control index 0x00)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
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 XL710.

10.6.4.5 Set Intel filters formats

10.6.4.5.1 Set Intel filters command (Intel command 0x02)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x02	Parameter Number	Filters Data (optional)	

10.6.4.5.2 Set Intel filters response (Intel command 0x02)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...	0x02	Filter Control Index	Return Data (Optional)	

10.6.4.5.3 Set Intel filters — manageability only command (Intel command 0x02, filter parameter 0x0F)

This command sets the PRT_MNG_MNGONLY register. The PRT_MNG_MNGONLY register controls whether PT packets destined to the MC are not forwarded to the Host operating system. The PRT_MNG_MNGONLY register is listed in Table 10-4.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x0F	Manageability Only (3-2)	
24...25	Manageability Only (1-0)			

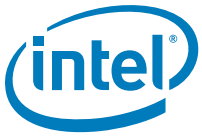
10.6.4.5.4 Set Intel filters — manageability only response (Intel command 0x02, filter parameter 0x0F)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x0F		

10.6.4.5.5 Set Intel filters — flex filter enable mask and length command (Intel command 0x02, filter parameter 0x10)

The following command sets the Intel flex filters mask and length.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
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	Reserved
38	Length			



10.6.4.5.6 Set Intel filters — flex filter enable mask and length response (Intel command 0x02, filter parameter 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x10		

10.6.4.5.7 Set Intel filters — flex filter data command (Intel command 0x02, filter parameter 0x11)

Table 10-33. Filter data group

Code	Bytes Programmed	Filter Data Length
0x0	bytes 0-29	1 - 30
0x1	bytes 30-59	1 - 30
0x2	bytes 60-89	1 - 30
0x3	bytes 90-119	1 - 30
0x4	bytes 120-127	1 - 8

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...	0x02	0x11	Filter Data Group	Filter Data 1
	...	Filter Data N		

Note: Using this command to configure the filters data must be done after the flex filter mask command is issued and the mask is set.

10.6.4.5.8 Set Intel filters — flex filter data response (Intel command 0x02, filter parameter 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x11		

10.6.4.5.9 Set Intel filters — packet addition decision filter command (Intel command 0x02, filter parameter 0x61)

This command is no longer supported. Use the Set Intel Filters - Packet Addition Extended Decision Filter Command (Intel Command 0x02, Filter Parameter 0x68 - Section 10.6.4.5.20) instead.

10.6.4.5.10 Set Intel filters — flex TCP/UDP port filter command

10.6.4.5.11 (Intel command 0x02, filter parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x63	Port filter index	TCP/UDP Port MSB
24..27	TCP/UDP Port LSB	Port flags		

Filter index range: 0x0...0xA.

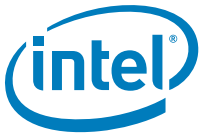
Port flags are as follows:

- 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 filter index is larger than 10, a command failed response code is returned with Invalid Intel Parameter Number reason (0x5082).

10.6.4.5.12 Set Intel filters — flex TCP/UDP port filter response



10.6.4.5.13 (Intel command 0x02, filter parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x63		

10.6.4.5.14 Set Intel filters — IPv4 filter command (Intel command 0x02, filter parameter 0x64)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x64	IP Filter Index	IPv4 Address (3)
24...26	IPv4 Address (2-0)			

Filter index range: 0x0...0x3.

10.6.4.5.15 Set Intel filters — IPv4 filter response (Intel command 0x02, filter parameter 0x64)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x64		

If the IP filter index is larger than 3, a command failed response code is returned Invalid Intel Parameter Number reason (0x5082).

10.6.4.5.16 Set Intel filters — IPv6 filter command (Intel command 0x02, filter parameter 0x65)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
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...37	...		IPv6 Address (LSB, byte 0)	

Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command.

IPv4 Mode: Filter index range: 0x1...0x3.

IPv6 Mode: Filter index range: 0x0...0x3.

10.6.4.5.17 Set Intel filters — IPv6 filter response (Intel command 0x02, filter parameter 0x65)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x65		

If the IP filter index is larger the 3, a command failed Response Code is returned, Invalid Intel Parameter Number reason (0x5082).

10.6.4.5.18 Set Intel filters - EtherType filter command (Intel command 0x02, filter parameter 0x67)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x67	EtherType Filter Index	EtherType Filter MSB
24...27	...	...	EtherType Filter LSB	

Where the EtherType filter has the format as described in Section 11.2.2.11.11.

10.6.4.5.19 Set Intel filters - EtherType filter response (Intel command 0x02, filter parameter 0x67)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x67		

If the Ethertype filter index is different than 2 or 3, a command failed Response Code is returned Invalid Intel Parameter Number reason (0x5082).

10.6.4.5.20 Set Intel filters - packet addition extended decision filter command (Intel command 0x02, filter parameter 0x68)

See Figure 10-2 for 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**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
20...23	0x02	0x68	Extended Decision filter Index	Extended Decision filter 1 MSB
24...27	...	...	Extended Decision filter 1 LSB	Extended Decision filter 0 MSB
28...30	...	...	Extended Decision filter 0 LSB	

Extended decision filter index range: 0...4

Filter 0: See Table 10-34.

Filter 1: See Table 10-35.

Table 10-34. Filter 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 10-35. Extended filter 1 values

Bit #	Name	Description
3:0	Ethertype 0 -3 (AND)	If set, packets must match the Ethertype filter 0 to 3, respectively.
7:4	Ethertype 0 -3 (OR)	If set, packets must match the Ethertype filter 0 to 3, respectively or a different OR filter.
18:8	Flex port 10:0 (OR)	If set, packets can pass if match the TCP/UDP Port filter 10:0.
19	DHCPv6 (OR)	If set, packets can pass if match the DHCPv6 port (0x0223).
20	DHCP Client (OR)	If set, packets can pass if match the DHCP Server port (0x0043).
21	DHCP Server (OR)	If set, packets can pass if match the DHCP Client port (0x0044).
22	NetBIOS Name Service (OR)	If set, packets can pass if match the NetBIOS Name Service port (0x0089).
23	NetBIOS Datagram Service (OR)	If set, packets can pass if match the NetBIOS Datagram Service port (0x008A).

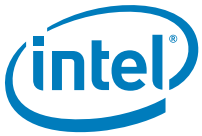


Table 10-35. Extended filter 1 values

Bit #	Name	Description
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

10.6.4.5.21 Set Intel filters – packet addition extended decision filter response (Intel command 0x02, filter parameter 0x68)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x68		

If the extended decision filter index is larger than 5, a command failed Response Code is returned Invalid Intel Parameter Number reason (0x5082).

10.6.4.5.22 Set Intel filters - special modifier command (Intel command 0x02, filter parameter 0x69)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
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 11.2.2.11.15. The value set is not checked.

10.6.4.5.23 Set Intel filters - special modifier response (Intel command 0x02, filter parameter 0x69)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x69		

10.6.4.6 Get Intel filters formats

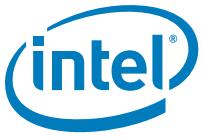
10.6.4.6.1 Get Intel filters command (Intel command 0x03)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	Parameter Number		

10.6.4.6.2 Get Intel filters response (Intel command 0x03)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x03	Parameter Number	Optional Return Data	

10.6.4.6.3 Get Intel filters — manageability only command (Intel command 0x03, filter parameter 0x0F)



This command retrieves the PRT_MNG_MNGONLY register. The PRT_MNG_MNGONLY register controls whether PT packets destined to the MC are also be forwarded to the host operating system.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	0x0F		

10.6.4.6.4 Get Intel filters — manageability only response (Intel command 0x03, filter parameter 0x0F)

The PRT_MNG_MNGONLY register structure is listed in Table 10-4.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x0F	Manageability to Host (3-2)	
28...29	Manageability to Host (1-0)			

10.6.4.6.5 Get Intel filters — flex filter 0 enable mask and length command (Intel command 0x03, filter parameter 0x10)

The following command retrieves the Intel flex filters mask and length. See Section 10.3.3.6 for details of the values returned by this command.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	0x10		

10.6.4.6.6 Get Intel filters — flex filter 0 enable mask and length response (Intel command 0x03, filter parameter 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
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	Reserved
44	Flexible Filter Length			

10.6.4.6.7 Get Intel filters — flex filter 0 data command (Intel command 0x03, filter parameter 0x11)

The following command retrieves the Intel flex filters data.

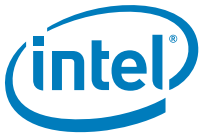
	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	0x11	Filter Data Group 0...4	

The filter data group parameter defines which bytes of the flex filter are returned by this command:

Table 10-36. Filter data group

Code	Bytes Returned
0x0	bytes 0-29
0x1	bytes 30-59
0x2	bytes 60-89
0x3	bytes 90-119
0x4	bytes 120-127

10.6.4.6.8 Get Intel filters — flex filter 0 data response (Intel command 0x03, filter parameter 0x11)



	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...	0x03	0x11	Filter Group Number	Filter Data 1
	...	Filter Data N		

10.6.4.6.9 Get Intel filters — packet addition decision filter command (Intel command 0x03, filter parameter 0x61)

This command is no longer supported. Use the Get Intel Filters - Packet Addition Extended Decision Filter Command (Intel Command 0x03, Filter Parameter 0x68 - Section 10.6.4.6.19) instead.

10.6.4.6.10 Get Intel filters — flex TCP/UDP port filter command (Intel command 0x03, filter parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x63	TCP/UDP Filter Index	

Filter index range: 0x0...0x2.

10.6.4.6.11 Get Intel filters — flex TCP/UDP port filter response

10.6.4.6.12 (Intel command 0x03, filter parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x63	TCP/UDP Filter Index	TCP/UDP Port (1)
28..29	TCP/UDP Port (0)	Port flags		

Filter index range: 0x0...0x2.

10.6.4.6.13 Get Intel filters — IPv4 filter command (Intel command 0x03, filter parameter 0x64)

	Bits			
Bytes	31...24	23...16	15...08	07...00
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x64	IPv4 Filter Index	

Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command.

IPv4 Mode: Filter index range: 0x0...0x3.

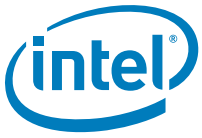
IPv6 Mode: This command should not be used in IPv6 mode.

10.6.4.6.14 Get Intel filters — IPv4 filter response (Intel command 0x03, filter parameter 0x64)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x64	IPv4 Filter Index	IPv4 Address (3)
28...29	IPv4 Address (2-0)			

10.6.4.6.15 Get Intel filters — IPv6 filter command (Intel command 0x03, filter parameter 0x65)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x65	IPv6 Filter Index	



Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command

IPv4 Mode: Filter index range: 0x0...0x2.

IPv6 Mode: Filter index range: 0x0...0x3.

10.6.4.6.16 Get Intel filters — IPv6 filter response Intel command 0x03, filter parameter 0x65)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
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)	

10.6.4.6.17 Get Intel filters - EtherType filter command (Intel command 0x03, filter parameter 0x67)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x67	EtherType Filter Index	

Valid indices: 0...3

10.6.4.6.18 Get Intel filters - EtherType filter response (Intel command 0x03, filter parameter 0x67)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x67	EtherType Filter Index	EtherType Filter MSB
28...30	..	..	EtherType Filter LSB	

If the Ethertype filter index is larger than 3, a command failed Response Code is returned Invalid Intel Parameter Number reason (0x5082).

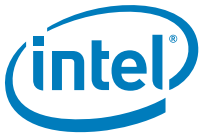
10.6.4.6.19 Get Intel filters – packet addition extended decision filter command (Intel command 0x03, filter parameter 0x68)

This command enables the MC to retrieve the extended decision filter.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x68	Extended Decision Filter Index	

10.6.4.6.20 Get Intel filters – Packet addition extended decision filter response (Intel command 0x03, filter parameter 0x68)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x68	Decision Filter Index	Decision Filter 1 MSB
28...31	..	..	Decision Filter 1 LSB	Decision Filter 0 MSB
32...34	..	..	Decision Filter 0 LSB	



Where decision filter 0 and decision filter 1 have the structure as detailed in the respective Set commands.

If the extended decision filter index is larger than 4, a command failed Response Code is returned Invalid Intel Parameter Number reason (0x5082).

10.6.4.6.21 Get Intel filters – special modifier command (Intel command 0x03, filter parameter 0x69)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x03	0x69	Padding	

Where the special modifier filter has the format as described in Section 11.2.2.11.15.

10.6.4.6.22 Get Intel filters - special modifier response (Intel command 0x02, filter parameter 0x69)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x69	Special Modifier Register MSB	
28...29	Special Modifier Register LSB		Padding	

10.6.4.7 Set Intel Packet Reduction Filters Formats

The non-extended commands are obsolete. The extended commands (Section 10.6.4.7.3 to Section 10.6.4.7.8) should be used instead.

10.6.4.7.1 Set Intel packet reduction filters command (Intel command 0x04)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x04	Packet Reduction Index	Packet Reduction Data...	

Note: It is advised that the MC only use the extended packet reduction commands.

The *Packet Reduction* data field has the following structure:

Table 10-37. Packet reduction field description

Bit #	Name	Description
12:0	Reserved	Reserved
16:13	IPv4 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively.
20:17	IPv6 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively.
27:21	Reserved	Reserved.
28	ARP Response (OR)	If set, packets can pass if match the ARP response filter or a different OR filter.
29	Reserved	Reserved.
30	Port 0x298	If set, packets can pass if match a fixed TCP/UDP port 0x298 filter.
31	Port 0x26F	If set, packets can pass if match a fixed TCP/UDP port 0x26F filter.

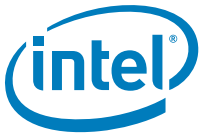
Table 10-38. Extended packet reduction field description

Bit #	Name	Description
3:0	Ethertype 0 -3 (AND)	If set, packets must match the Ethertype filter 0 to 3, respectively.
7:4	Ethertype 0-3 (OR)	If set, packets can pass if match the Ethertype filter 0 to 3, respectively.
15:12	Reserved	Reserved.
8:18	Flex port 10:0 (OR)	If set, packets can pass if match the TCP/UDP Port filter 10:0.
23:19	Reserved	Reserved.
24	Flex TCO (OR)	If set, packets can pass if match the Flex 128 TCO filter.
27:25	Reserved	Reserved.
28	ICMPv4	Is set, ICMPv4 packets can pass.
31:29	Reserved	Reserved.

The filtering is divided into two decisions:

- Bit 20:13 in Table 10-37 and Bits 3:2 in Table 10-38 works in an AND manner; it must be true in order for a packet to pass (if was set).

Bits 28 in Table 10-37 and Bits 24:10 in Table 10-38 work in an OR manner; at least one of them must be true for a packet to pass (if any were set).



10.6.4.7.2 Set Intel packet reduction filters response (Intel command 0x04)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...	0x04	Packet Reduction Index		

10.6.4.7.3 Set unicast extended packet reduction command (Intel command 0x04, reduction filter Index 0x10)

The command has the following format:

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x04	0x10	Extended Unicast Reduction Filter MSB	..
24..27	..	Extended Unicast Reduction Filter LSB	Unicast Reduction Filter MSB	..
28..29	..	Unicast Reduction Filter LSB		

The command overwrites any previously stored value.

Note: See Table 10-37 and Table 10-38 for description of the unicast extended packet reduction format.

10.6.4.7.4 Set unicast extended packet reduction response (Intel command 0x04, reduction filter index 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			

	Bits			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x04	0x10		

10.6.4.7.5 Set multicast extended packet reduction command (Intel command 0x04, reduction filter index 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x04	0x11	Extended Multicast Reduction Filter MSB	..
24..27	..	Extended Multicast Reduction Filter LSB	Multicast Reduction Filter MSB	..
28..29	..	Multicast Reduction Filter LSB		

Note: See Table 10-37 and Table 10-38 for description of the multicast extended packet reduction format.

The command overwrites any previously stored value.

10.6.4.7.6 Set multicast extended packet reduction response (Intel command 0x04, reduction filter index 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x04	0x11		

10.6.4.7.7 Set broadcast extended packet reduction command (Intel command 0x04, reduction filter index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x04	0x12	Extended Broadcast Reduction Filter MSB	..
24..27	..	Extended Broadcast Reduction Filter LSB	Broadcast Reduction Filter MSB	..
28..29	..	Broadcast Reduction Filter LSB		

Note: See Table 10-37 and Table 10-38 for description of the broadcast extended packet reduction format.

The command overwrites any previously stored value.

10.6.4.7.8 Set broadcast extended packet reduction response (Intel command 0x04, reduction filter index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x04	0x12		

10.6.4.8 Get Intel packet reduction filters formats

Note: The non extended commands are obsolete. Use the extended commands (Section 10.6.4.8.1 to Section 10.6.4.8.6) instead.

10.6.4.8.1 Get unicast extended packet reduction command (Intel command 0x05, reduction filter index 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x10		

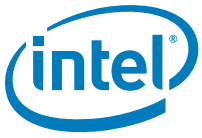
10.6.4.8.2 Get unicast extended packet reduction response (Intel command 0x05, reduction filter index 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x00	Extended Unicast Packet Reduction (3-2)	
28...29	Extended Unicast Packet Reduction (1-0)		Unicast Packet Reduction (3-2)	
30...31	Unicast Packet Reduction (1-0)			

10.6.4.8.3 Get multicast extended packet reduction command (Intel command 0x05, reduction filter index 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x11		

10.6.4.8.4 Get multicast extended packet reduction response (Intel command 0x05, reduction filter index 0x11)



	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x11	Extended Multicast Packet Reduction (3-2)	
28...29	Extended Multicast Packet Reduction (1-0)		Multicast Packet Reduction (3-2)	
30...31	Multicast Packet Reduction (1-0)			

10.6.4.8.5 Get broadcast extended packet reduction command (Intel command 0x05, reduction filter index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x12		

10.6.4.8.6 Get broadcast extended packet reduction response (Intel command 0x05, reduction filter index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x12	Extended Broadcast Packet Reduction (3-2)	
28...29	Extended Broadcast Packet Reduction (1-0)		Broadcast Packet Reduction (3-2)	
30...31	Broadcast Packet Reduction (1-0)			

10.6.4.9 System MAC address

10.6.4.9.1 Get system MAC address command (Intel command 0x06)

In order to support a system configuration that requires the NC to hold the MAC address for the MC (such as shared MAC address mode), the following command is provided to enable the MC to query the NC for a valid MAC address.

The NC must return the system MAC addresses. The MC should use the returned MAC addressing as a shared MAC address by setting it using the Set MAC Address command as defined in NC-SI 1.0.

When a single function is defined on the port, it returns the LAN MAC address of this function as read from the PF allocations NVM section or from the alternate RAM. 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.

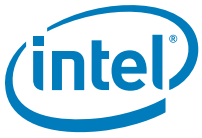
	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20	0x06			

10.6.4.9.2 Get system MAC address response (Intel command 0x06)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x06	MAC Address		
28...30	MAC Address			

10.6.4.10 Set Intel management control formats

10.6.4.10.1 Set Intel management control command (Intel command 0x20)



	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x20	0x00	Intel Management Control 1	

Where Intel Management Control 1 is as follows:

Bit #	Default value	Description
0	0b	Enable Critical Session Mode (Keep PHY Link Up and Veto Bit). 0b = Disabled. 1b = Enabled. When critical session mode is enabled, the following behaviors are disabled: • The PHY is not reset on PE_RST# and PCIe resets (in-band and link drop). Other reset events are not affected — Internal_Power_On_Reset, device disable, Force TCO, and PHY reset by software. • 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.

10.6.4.10.2 Set Intel management control response (Intel command 0x20)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x20	0x00		

10.6.4.11 Get Intel management control formats

10.6.4.11.1 Get Intel management control command (Intel command 0x21)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x21	0x00		

Where Intel Management Control 1 is as described in Section 10.6.4.10.2.

10.6.4.11.2 Get Intel management control response (Intel command 0x21)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...26	0x21	0x00	Intel Management Control 1	

10.6.4.12 TCO reset

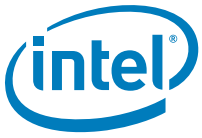
Depending on the bit set in the TCO mode field this command causes the XL710 to perform either:

1. TCO reset — If force TCO reset is enabled in the NVM (see Section 7.2.31.2). The force TCO reset clears the data path (Rx/Tx) of the XL710 to enable the MC to transmit/receive packets through the XL710.

- 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.
- When this command is issued to a channel in a package, it applies only to the specific channel.
- 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, the XL710 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 7.2.31.3). The TCO Isolate command disables PCIe write operations to the LAN port.

- If TCO isolate is disabled in NVM, the XL710 does not execute the command but sends a response to the MC with successful completion.
- Following a TCO isolate, management sets *EMP_TCO_ISOLATE.EMP_TCO_ISOLATE* to 1b for all PFs associated with the port on which this command is received.



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 FW Semaphore register (FWSM) is re-initialized.

Note: Applying this command resets the entire device and also has an effect on TCO reset.TCO isolate affects only the channel (port) that the command was issued to. Force TCO resets the entire device (all channels in the package).

Following firmware reset, the MC needs to re-initialize all ports. A firmware reset causes a global reset of the entire device (GLOBR).

Note: Only one of the fields should be set in a given command. Setting more than one field might yield unexpected results.

10.6.4.12.1 Perform Intel TCO reset command (Intel command 0x22)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20	0x22	TCO Mode		

Where TCO mode is:

Field	Bit(s)	Description
DO_TCO_RST	0	Do TCO Reset. 0b = Do nothing. 1b = Perform TCO reset.
DO_TCO_ISOLATE[1]	1	Do TCO Isolate. 0b = Enable PCIe write access to LAN port. 1b = Isolate Host PCIe write operation to the port ***Note:*** Should be used for debug only. ***Note:*** The TCO Isolate do not impact MCTP traffic ***Note:*** When isolate is set, the OS2BMC flow is also disabled.
RESET_MGMT	2	Reset Manageability; Re-load Manageability NVM Words. 0b = Do nothing. 1b = Issue firmware reset to manageability. Setting this bit generates a one-time firmware reset. Following the reset, management related data from the NVM is loaded. ***Note:*** A reset of the internal firmware causes a reset of the entire device.
Reserved	7:3	Reserved (set to 0x00).

Note: For compatibility, the TCO Reset command without the TCO mode parameter is accepted (TCO reset is done).

1. TCO isolate host write operation enabled in the NVM.

10.6.4.12.2 Perform Intel TCO reset response (Intel command 0x22)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...26	0x22			

10.6.4.13 Checksum offloading

This command enables the checksum offloading filters in the NC.

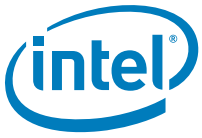
When enabled, these filters block any packets that did not pass IP, UDP or TCP checksum from being forwarded to the MC.

10.6.4.13.1 Enable checksum offloading command (Intel command 0x23)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20	0x23			

10.6.4.13.2 Enable checksum offloading response (Intel command 0x23)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			



	Bits	
16...19	Response Code	Reason Code
20...23	Manufacturer ID (Intel 0x157)	
24...26	0x23	

10.6.4.13.3 Disable checksum offloading command (Intel command 0x24)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20	0x24			

10.6.4.13.4 Disable checksum offloading response (Intel command 0x24)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...26	0x24			

10.6.4.14 Shared MAC and shared IP support commands (Intel command 0x25)

To meet the requirements introduced by sharing IP addresses, modifications and additions to the NC-SI command set are required. These changes include the new commands in this section and the modifications described in Section 10.3.6.

Note: All indexes in this command set starts at one to match the NC-SI methodology.

10.6.4.14.1 Set IP address command (Intel command 0x25, index = 0x0)

The Set IP Address command is used by the MC to communicate its IP address to a NC. The format of a Set IP Address command packet is listed in Table 10-39.

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.

Table 10-39. Set IP address command packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
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			

- MC IP Address — An IP address that is used by the MC. If the *IP Version* bit of the *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 *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 the following table:

Table 10-40. IP filters 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 refers to the Get Shared IP Capabilities Response (Section 10.6.4.14.6.1).

- Table 10-41 lists the bits fields in the *Set IP Flags* field.

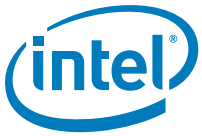


Table 10-41. Set IP flag field

Bit Position	Field Description	Value Description
0	Enable	0b = Disable the filter. 1b = Enable the filter.
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. 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.

10.6.4.14.1.1 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 format listed in Table 10-42.

Table 10-42. Set IP address response packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x25	0x0	Reserved	
28..31	Checksum			

10.6.4.14.2 Get IP address command (Intel command 0x25, index = 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. The format of a Get IP Address command packet is listed in Table 10-43.

Table 10-43. Get IP address command packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x25	0x1	Reserved	
24..27	Reserved		IP Address Number	IP filter pool
28..31	Checksum			

- 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 10-40.
- IP filter pool:
  — 0x0: Mixed IP filters
  — 0x1: IPv4 filters
  — 0x2: IPv6 filters
  — 0x3 - 0xFF: Reserved

10.6.4.14.2.1 Get IP address response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Get IP Address command and send a response using the format listed in Table 10-44.

Table 10-44. Get IP address response packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
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			

- MC IP Address — An IP address that is used by the MC. If the *IP Version* bit of the *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.
- Table 10-45 lists the bits fields in the *Get IP Flags* field.

Table 10-45. Get IP flag field

Bit Position	Field Description	Value Description
0	Enable	0b = Filter is disabled. 1b = Filter is enabled.
1	IP version	0b = IPv4. 1b = IPv6.

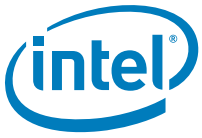


Table 10-45. Get IP flag field

Bit Position	Field Description	Value 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 flag defines if the IPpv6 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. 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.

10.6.4.14.3 Set port command (Intel command 0x25, index = 0x2)

An MC uses the Set Port command to communicate one of its TCP or UDP ports to a NC. The format of a Set Port command packet is listed in Table 10.6.4.14.3.1.

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.

10.6.4.14.3.1 Set port command packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x25	0x2	Set Port Flags	Reserved
24..27	Port Index	Protocol	Port	
28..31	Checksum			

- 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.

Table 10.6.4.14.3.2 lists the fields in the *Set Port Flags* field.

10.6.4.14.3.2 Set port flags field descriptions

Bit Position	Field Description	Value 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.

10.6.4.14.3.3 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 format listed in Table 10-46.

Table 10-46. Set port response packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
12..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x25	0x2	Reserved	
28..31	Checksum			

10.6.4.14.4 Get port command (Intel command 0x25, index = 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. The format of a Get Port command packet is listed in Table 10-47.

Table 10-47. Get port command packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x25	0x3	Reserved	
24..27	Reserved		Port Index	Reserved
28..31	Checksum			

Table 10-48 lists the fields in the Get Port command.

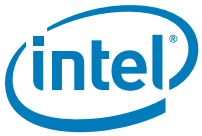


Table 10-48. Get port command field descriptions

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.

10.6.4.14.4.1 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 format listed in Table 10-49.

Table 10-49. Get port response packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
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			

- 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.

Table 10-50 lists the fields in the *Get Port Flags* field.

Table 10-50. Get port flags field descriptions

Bit Position	Field Description	Value 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.

10.6.4.14.5 Enable unicast infrastructure filter command (Intel command 0x25, index = 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.

The format of an Enable Unicast Infrastructure Filter command packet is listed in Table 10-51.

Table 10-51. Enable unicast infrastructure filter command

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
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..29	Padding			

Table 10-52 lists the sub fields of the *Unicast Infrastructure Filter Settings* field.

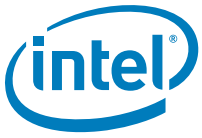
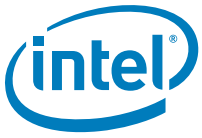


Table 10-52. Unicast infrastructure packet filter settings field

Bit Position	Field Description	Value Description
0	ARP Response Packets Received From Wire	0x1 = Forward this packet type to both the host and the MC. 0x0 = Forward this packet type to the host only. 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 ARP *Target Protocol Address* field contains the IP address assigned to the MC.
1	ICMPv4 Request Packets Received From Wire	0x1 = Forward this packet type to both the host and the MC. 0x0 = Forward this packet type to the host only. 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	0x1 = Forward this packet type to both the host and the MC. 0x0 = Forward this packet type to the host only. 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 XL710.
3	DHCP Server Unicast Packets Received From Wire	0x1 = Forward this packet type to both the host and the MC. 0x0 = Forward this packet type to the host only. 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	0x1 = Forward this packet type to both the host and the MC. 0x0 = Forward this packet type to the host only. 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	0x1 = Forward this packet type to both the wire and the MC. 0x0 = Forward this packet type to the wire only. 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 10-52. Unicast infrastructure packet filter settings field

Bit Position	Field Description	Value Description
6	DHCPv6 Server Unicast Packets Received From Wire	0x1 = Forward this packet type to both the host and MC. 0x0 = Forward this packet type to the host only. 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	0x1 = Forward this packet type to the MC only. 0x0 = Forward this packet type to the host. 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 address 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	0x1 = Forward this packet type to the MC only. 0x0 = Forward this packet type to the host. 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 address 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	0x1 = Forward this packet type to the MC only. 0x0 = Forward this packet type to the host 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) or 0x0800 (IPv4). • The IP *Destination Address* field contains the one of the IP address 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	0x1 = Forward this packet type to the MC only. 0x0 = Forward this packet type to the host. 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 address assigned to the MC. • The IP *Protocol* field contains 6 (TCP). • The TCP *Destination Port* field contains 664 [secure aux bus (secondary RMCP port)].
31:11	Reserved	None.



10.6.4.14.5.1 Enable unicast infrastructure filter response

The NC, in the absence of a checksum error or identifier mismatch, always accept the Enable Unicast Infrastructure Filter command and send a response using the format listed in Table 10-53. Currently no command-specific reason codes are identified for this response.

Table 10-53. Enable unicast infrastructure filter response packet format

	Bits			
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x25	0x4	Reserved	
28..29	Checksum			

10.6.4.14.6 Get shared IP capabilities command (Intel command 0x25, index = 0x5)

An MC uses the Get Shared IP Capabilities command to determine the level of support of shared IP of the device. The format of a Get Shared IP Capabilities command packet is listed in Table 10-54.

Table 10-54. Get shared IP capabilities command packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x25	0x5	Reserved	
24..27	Checksum			

10.6.4.14.6.1 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 format listed in Table 10-55. Currently no command-specific reason codes are identified for this response.

Table 10-55. Get shared IP capabilities response packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x25	0x5	Number of Mixed IP address	Number of IPv4 only addresses

Table 10-55. Get shared IP capabilities response packet format

	Bits			
28..31	Number of IPv6 only addresses	Number of Ports	Number of bindings	Filtering capabilities
32..35	Unicast Infrastructure Filter capabilities			
36..30	Checksum			

- Number of mixed IP addresses — The number of supported IP filters that can be used for IPv4 or IPv6. The XL710 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 XL710 supports three IPv4 address filters.
- Number of IPv6 only addresses — The number of supported IP filters that can be used for IPv6 only. The XL710 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 10-52. 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 XL710 supports all filters but ICMPv6 filtering, so the returned value is 0x7FB.
- Table 10-56 lists the bits fields in the *Filtering Capabilities* field.

Table 10-56. Filtering capabilities field

Bit Position	Field Description	Value 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.

10.6.4.14.7 Shared IP enable broadcast filtering command (Intel command 0x25, index = 0x6)

A new shared IP enable broadcast filtering 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.

The format of an Shared IP Enable Broadcast Filtering command packet is listed in Table 10-51.

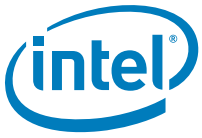


Table 10-57. Shared IP enable broadcast filtering command

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
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..29	Padding			

The content of the *Shared IP Broadcast Packet Filter Settings* field is listed in Table 10-58. Bit 4 has been added to the standard enable broadcast filtering command limit ARP broadcast packets to the MC IP address.

Table 10-58. Shared IP broadcast packet filter settings field

Bit Position	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: 0x1 = Forward only ARP broadcast packets that are targeted at IP addresses bound to the MC. 0x0 = Forward all ARP broadcast packets to the MC. All the IPs set by the Set IP command before this command is given will be 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	None.

10.6.4.14.7.1 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 format listed in Table 10-53. Currently no command-specific reason codes are identified for this response.

Table 10-59. Shared IP enable broadcast filtering packet format

	Bits			
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x25	0x6	Reserved	
28..29	Checksum			

10.6.4.14.8 Shared IP enable global multicast filtering command (Intel command 0x25, index = 0x7)

A new shared IP enable global multicast filtering 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.

The format of an Shared IP Enable Global Multicast Filtering command packet is listed in Table 10-60.

Table 10-60. Shared IP enable global multicast filtering command

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
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..29	Padding			

The content of the *Shared IP Multicast Packet Filter Settings* field is listed in Table 10-61. Bit 4 has been added to the standard enable broadcast filtering command limit ARP broadcast packets to MC IP address.

Table 10-61. Shared IP multicast packet filter settings field

Bit Position	Field Description	Value Description
0:2	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	0x1 = Forward this packet type to the MC. 0x0 = Filter out this packet type. 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). 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.
4:31	Reserved	None.

10.6.4.14.8.1 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 format listed in Table 10-62. Currently no command-specific reason codes are identified for this response.

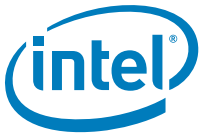


Table 10-62. Shared IP enable global multicast filtering packet format

	Bits		
00..15	NC-SI Header		
16..19	Response Code		Reason Code
20..23	Manufacturer ID (Intel 0x157)		
24..27	0x25	0x7	Reserved
28..29	Checksum		

10.6.4.14.9 Get shared IP parameters command (Intel command 0x25, index = 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. The format of a Get Shared IP Capabilities command packet is listed in Table 10-63.

Table 10-63. Get shared IP parameters command packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x25	0x8	Reserved	
24..27	Checksum			

10.6.4.14.9.1 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 format listed in Table 10-64. Currently no command-specific reason codes are identified for this response.

Table 10-64. Get shared IP parameters response packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x25	0x5	Reserved	
28..31	Number of IP Addresses	IP addresses Flags		
32..35	Number of ports	Ports Flags		
36..39	Unicast Infrastructure Filter Settings			

Table 10-64. Get shared IP parameters response packet format

	Bits
40..43	Broadcast Filtering Settings
44..47	Multicast Filtering Settings
48..51	Checksum

- 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 10-65.

Table 10-65. IP address flags field

Bit Position	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.

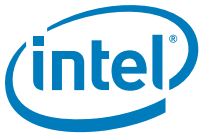
For example, if the interface reports four IPv4 filters, two IPv6 filters, and two mixed filters, then IP addresses 1 through 4 are those currently configured through the interface's IPv4 filters, IP addresses 5 and 6 are those configured through the IPv6 filters, and 7 and 8 are those configured through 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 10-65.

Table 10-66. Port flags field

Bit Position	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 10-52. 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 10-58. A bit set to 1b indicates that the filter associated with that bit position is enabled; otherwise, the filter is not enabled.
- Global 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 10-58. A bit set to 1b indicates that the filter associated with that bit position is enabled; otherwise, the filter is not enabled.

10.6.4.14.10 Set binding command (Intel command 0x25, index = 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. The format of a Set Binding command packet is listed in Table 10-67.

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 10-67. Set binding command packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
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			

Table 10-68 lists the fields in the *Set Binding Flags* field.

Table 10-68. Set binding flags field descriptions

Bit Position	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.

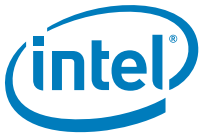
- Binding Index — Indicates which binding is configured by the command.The value should be smaller than the number of supported bindings as reported in the get shared IP capabilities response in the *Number of Bindings* field.
- Enabled MAC Addresses — The MAC addresses participating in this binding. 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 10.6.4.14.9. 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.

10.6.4.14.10.1 Set binding address 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 format listed in Table 10-69.

Table 10-69. Set binding response packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x25	0x9	Reserved	
28..31	Checksum			



10.6.4.14.11 Get binding command (Intel command 0x25, index = 0xA)

A MCr uses the Get Binding command to determine the current programming of one of the bindings in a NC. The format of a *Get Binding* command packet is listed in Table 10-70.

Table 10-70. Get binding command packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x25	0xA	Binding Number	Reserved
24..27	Checksum			

- Binding Index: 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.

10.6.4.14.11.1 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 format listed in Table 10-71.

Table 10-71. Get binding response packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
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.

10.6.4.14.12 Set shared mode command (Intel command 0x25, index = 0xB)

An MC uses the Set Shared Mode command to indicate to the NIC 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 only needed when the Intel OEM commands with command ID 0x25 are used to configure the shared behavior. If other commands are used, users should take care of the right 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 (10.6.4.14) received before shared mode is set fails with a Not is Shared Mode (0x5092) reason.

The format of a Set Shared Mode command packet is listed in Table 10-72.

Table 10-72. Set shared mode command packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x25	0xB	Shared Mode	Reserved
24..27	Checksum			

- Shared mode:
  - 0x0: Dedicated MAC mode.
  - 0x1: Shared MAC/IP mode.

10.6.4.14.12.1 Set shared mode response

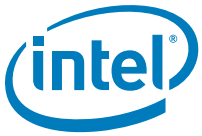
The NC must, in the absence of a checksum error or identifier mismatch, always accept the Set Shared Mode command and send a response using the format listed in Table 10-73.

Table 10-73. Set shared mode response packet format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x25	0xB	Shared Mode	Reserved
48..51	Checksum			

10.6.4.15 OS2BMC configuration

These commands control enabling of the OS2BMC flow.



10.6.4.15.1 EnableOS2BMC flow command (Intel command 0x40, index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x01		

10.6.4.15.2 EnableOS2BMC flow response (Intel command 0x40, index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x01		

10.6.4.15.3 Enable network-to-BMC flow command (Intel command 0x40, index 0x2)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x02		

10.6.4.15.4 Enable network-to-BMC flow response (Intel command 0x40, index 0x2)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x02		

10.6.4.15.5 Enable both host and network-to-BMC flows command (Intel command 0x40, index 0x3)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x03		

10.6.4.15.6 Enable both host and network-to-BMC flows response (Intel command 0x40, index 0x3)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x03		

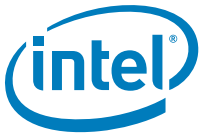
10.6.4.15.7 Set MC IP address command (Intel command 0x40, index 0x4)

This command is used to expose the MC IP address to the host. This command is supported by the XL710, but no action is taken upon reception. The IP address is not stored and when a Get OS2BMC Parameters command is received, the IP valid flag is cleared.

The IP type entry indicate whether the IP address is an IPv4 or an IPv6 address:

0 = IPv4.

1 = IPv6.



2 = No IP address, then the command should not include an IP address.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
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	

10.6.4.15.8 Set BMC IP address response (Intel command 0x40, index 0x4)

	Bits			
Bytes	**31...24**	**23...16**	**15...08**	**07...00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x04		

10.6.4.15.9 Get OS2BMC parameters command (Intel command 0x41)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20	0x41			

10.6.4.15.10 Get OS2BMC parameters response (Intel command 0x41)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
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	..	..	..	..
39..40	..	IPv6 Address (LSB, byte 0)/Reserved		

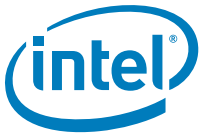
Where the status byte partition is as follows:

Table 10-74. Status byte description

Bits	Content
0	0b = IPv4. 1b = IPv6. Relevant only if the IP address valid bit is set.
1	IP address valid. Never valid for the XL710.
1:0	Reserved.
2	Network to BMC Status. 0b = Network-to-BMC flow is disabled. 1b = Network-to-BMC flow is enabled.
3	OS2BMC Status. 0b = OS2BMC flow is disabled. 1b = OS2BMC flow is enabled.
7:4	Reserved.

10.6.4.16 Get controller information command (Intel command 0x48, Index 0x1)

This command gathers the controller identification information and return it back to the MC.



	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x48	0x1		

10.6.4.16.1 Get controller information response (Intel command 0x48, Index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x48	0x01	Reserved	Number of Inventory entries
28...31	Controller Info Item 1 ID	Controller Info Item 1 length	Controller Info Item 1 Data	
...	....			
...	Controller Info Item 2 ID	Controller Info Item 2 length	Controller Info Item 2 Data	
...	....			
...	Controller Info Item n ID	Controller Info Item n length	Controller Info Item n Data	
...	....			

The possible inventory items are described as follows. Note that not all the inventory items would be present in all the implementations of this command.

Table 10-75. 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.

Table 10-75. Controller information items

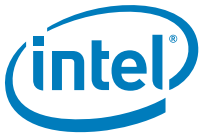
ID	Length (in bytes)	Data	Notes
0x00	3	Device ID (2 bytes) + RevID	This is the hardware default value (no value programmed via the NVM).
0x0B	2	NVM Image Version	
0x0C	4	EMP ROM Internal Version	
0x0E	2	PXE FW version	MajorVersion.MinorVersion.Build.
0x0F	2	iSCSI FW version	
0x10	2	uEFI FW version	
0x16	2	FCoE Boot FW version	

10.6.4.17 NVM error AEN (Intel AEN 0x82)

The following is the AEN that might be sent by the NC following a detection of a wrong CRC/checksum on a firmware related section in the NVM. NC is required to store this AEN internally until a connection to the MC is established so the report could be issued in all cases.

This AEN must be enabled using the NC-SI AEN Enable command, using bit 18 (0x40000) of the AEN enable mask.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00...15	NC-SI AEN Header			
20...23	Reserved			0x82
24...27	Index of module on which the error was found. The encoding is as defined in *GL_MNG_FWSM.EXT_ERR_IND* field.			



10.6.5 Basic NC-SI workflows

10.6.5.1 Package states

A NC package can be in one of the following two states:

1. Selected — The package is allowed to use the NC-SI lines, meaning the NC package might send data to the MC.
2. De-selected — The package is not allowed to use the NC-SI lines, meaning, the NC package cannot send data to the MC.

The MC must select no more than one NC package at any given time. Package selection can be accomplished in one of two methods:

1. Select Package command — This command explicitly selects the NC package.
2. Any other command targeted to a channel in the package also implicitly selects that NC package.

Package de-select can be accomplished only by issuing the De-Select Package command. The MC should always issue the Select Package command as the first command to the package before issuing channel-specific commands. For further details on package selection, refer to the NC-SI specification.

10.6.5.2 Channel states

A NC channel can be in one of the following states:

1. Initial State — The channel only accepts the Clear Initial State command (the package also accepts the Select Package and De-Select Package commands).
2. Active State — This is the normal operational mode. All commands are accepted.

For normal operation mode, the MC should always send the Clear Initial State command as the first command to the channel.

10.6.5.3 Discovery

After interface power-up, the MC should perform a discovery process to discover the NCs that are connected to it. This process should include an algorithm similar to the following:

1. For package_id=0x0 to MAX_PACKAGE_ID:
    a. Issue a Select Package command to package ID package_id.
    b. If a response was received:
    c. For internal_channel_id = 0x0 to MAX_INTERNAL_CHANNEL_ID.
    d. Issue a Clear Initial State command for package_id | internal_channel_id (the combination of package_id and internal_channel_id to create the channel ID).
    e. If a response was received:
    f. Consider internal_channel_id as a valid channel for the package_id package.
    g. The MC can now optionally discover channel capabilities and version ID for the channel.
    h. Else, if a response was not received, then issue a Clear Initial State command three times.
    i. Issue a De-Select Package command to the package (and continue to the next package).

j. Else, if a response was not received, issue a Select Packet command three times.

10.6.5.4 Configurations

This section details different configurations that should be performed by the MC.

It is good practice that the MC not consider any configuration valid unless the MC has explicitly configured it after every reset (entry into the initial state). As a result, it is recommended that the MC re-configure everything at power-up and channel/package resets.

10.6.5.4.1 NC capabilities advertisement

NC-SI defines the Get Capabilities command. It is recommended that the MC use this command and verify that the capabilities match its requirements before performing any configurations. For example, the MC should verify that the NC supports a specific AEN before enabling it.

10.6.5.4.2 Receive filtering

In order to receive traffic, the BMC must configure the NC with receive filtering rules. These rules are checked on every packet received on the LAN interface (such as from the network). Only if the rules matched, will the packet be forwarded to the BMC.

10.6.5.4.2.1 MAC address filtering

NC-SI defines three types of MAC address filters: unicast, multicast and broadcast. To be received (not dropped) a packet must match at least one of these filters. The MC should set one MAC address using the Set MAC Address command and enable broadcast and global multicast filtering.

Unicast/Exact Match (Set MAC Address command)

This filter filters on specific 48-bit MAC addresses. The MC must configure this filter with a dedicated MAC address.

The NC might expose three types of unicast/exact match filters (such as MAC filters that match on the entire 48 bits of the MAC address): unicast, multicast and mixed. The XL710 exposes two mixed filters, which might be used both for unicast and multicast filtering. The MC should use one mixed filter for its MAC address.

Refer to NC-SI specification — Set MAC Address for further details.

Broadcast (Enable/Disable Broadcast Filter command)

NC-SI defines a broadcast filtering mechanism that has the following states:

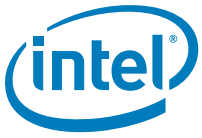
1. Enabled — All broadcast traffic is blocked (not forwarded) to the BMC, except for specific filters (such as ARP request, DHCP, and NetBIOS).
2. Disabled — All broadcast traffic is forwarded to the BMC, with no exceptions.

Refer to NC-SI specification Enable/Disable Broadcast Filter command.

Global Multicast (Enable/Disable Global Multicast Filter)

NC-SI defines a multicast filtering mechanism which has the following states:

1. Enabled — All multicast traffic is blocked (not forwarded) to the BMC.



2. Disabled — All multicast traffic is forwarded to the BMC, with no exceptions.

The recommended operational mode is Enabled, with specific filters set. Not all multicast filtering modes are necessarily supported. Refer to NC-SI specification Enable/Disable Global Multicast Filter command for further details.

10.6.5.4.3 VLAN

NC-SI defines the following VLAN work modes:

Mode	Command and Name	Descriptions
Disabled	Disable VLAN command	In this mode, no VLAN frames are received.
Enabled #1	Enable VLAN command with VLAN only	In this mode, only packets that matched a VLAN filter are forwarded to the MC.
Enabled #2	Enable VLAN command with VLAN only + non-VLAN	In this mode, packets from mode 1 + non-VLAN packets are forwarded.
Enabled #3	Enable VLAN command with Any-VLAN + non-VLAN	In this mode, packets are forwarded regardless of their VLAN state.

Refer to NC-SI specification — Enable VLAN command for further details.

The XL710 only supports modes #1 and #3. Recommendation:

1. Modes:
   a. If VLAN is not required — Use the disabled mode.
   b. If VLAN is required — Use the enabled #1 mode.
2. If enabling VLAN, The MC should also set the active VLAN ID filters using the NC-SI Set VLAN Filter command prior to setting the VLAN mode.

10.6.5.5 PT traffic states

The MC has independent, separate controls for enablement states of the receive (from LAN) and of the transmit (to LAN) PT paths.

10.6.5.6 Channel enable

This mode controls the state of the receive path:

1. Disabled — The channel does not pass any traffic from the network to the MC.
2. Enabled — The channel passes any traffic from the network (that matched the configured filters) to the MC.

This state also affects AENs: AENs is only sent in the enabled state.

The default state is disabled.

It is recommended that the MC complete all filtering configuration before enabling the channel.

10.6.5.7 Network transmit enable

This mode controls the state of the transmit path:

1. Disabled — the channel does not pass any traffic from the MC to the network.
2. Enabled — the channel passes any traffic from the MC (that matched the source MAC address filters) to the network.

The default state is disabled.

The NC filters PT packets according to their source MAC address. The NC tries to match that source MAC address to one of the MAC addresses configured by the Set MAC Address command. As a result, the MC should enable network transmit only after configuring the MAC address.

It is recommended that the MC complete all filtering configuration (especially MAC addresses) before enabling the network transmit.

This feature can be used for fail-over scenarios. See Section 10.6.9.3.

10.6.6 Asynchronous Event Notifications (AENs)

AENs are unsolicited messages sent from the NC to the MC to report status changes (such as link change, operating system state change, etc.).

Recommendations:

- The MC firmware designer should use AENs. To do so, the designer must take into account the possibility that a NC-SI response frame (such as a frame with the NC-SI EtherType), arrives out-of-context (not immediately after a command, but rather after an out-of-context AEN).
- To enable AENs, the MC should first query which AENs are supported, using the Get Capabilities command, then enable desired AEN(s) using the Enable AEN command, and only then enable the channel using the Enable Channel command.

10.6.7 Querying active parameters

The MC can use the Get Parameters command to query the current status of the operational parameters.

10.6.8 Resets

In NC-SI there are two types of resets defined:

1. Synchronous entry into the initial state.
2. Asynchronous entry into the initial state.

Recommendations:

- It is very important that the MC firmware designer keep in mind that following any type of reset, all configurations are considered as lost and thus the MC must re-configure both the synchronous and asynchronous entries.

- As an asynchronous entry into the initial state might not be reported and/or explicitly noticed, the MC should periodically poll the NC with NC-SI commands (such as Get Version ID, Get Parameters, etc.) to verify that the channel is not in the initial state. Should the NC channel respond to the command with a Clear Initial State Command Expected reason code, the MC should consider the channel (and most probably the entire NC package) as if it underwent a (possibly unexpected) reset event. Thus, the MC should re-configure the NC. See the NC-SI specification section on Detecting Pass-through Traffic Interruption.
- The Intel recommended polling interval is 2-3 seconds.

For exact details on the resets, refer to NC-SI specification.

10.6.9 Advanced workflows

10.6.9.1 Multi-NC arbitration

As described in Section 10.6.1.2, in a multi-NC environment, there is a need to arbitrate the NC-SI lines.

Figure 10-9 shows the system topology of such an environment.

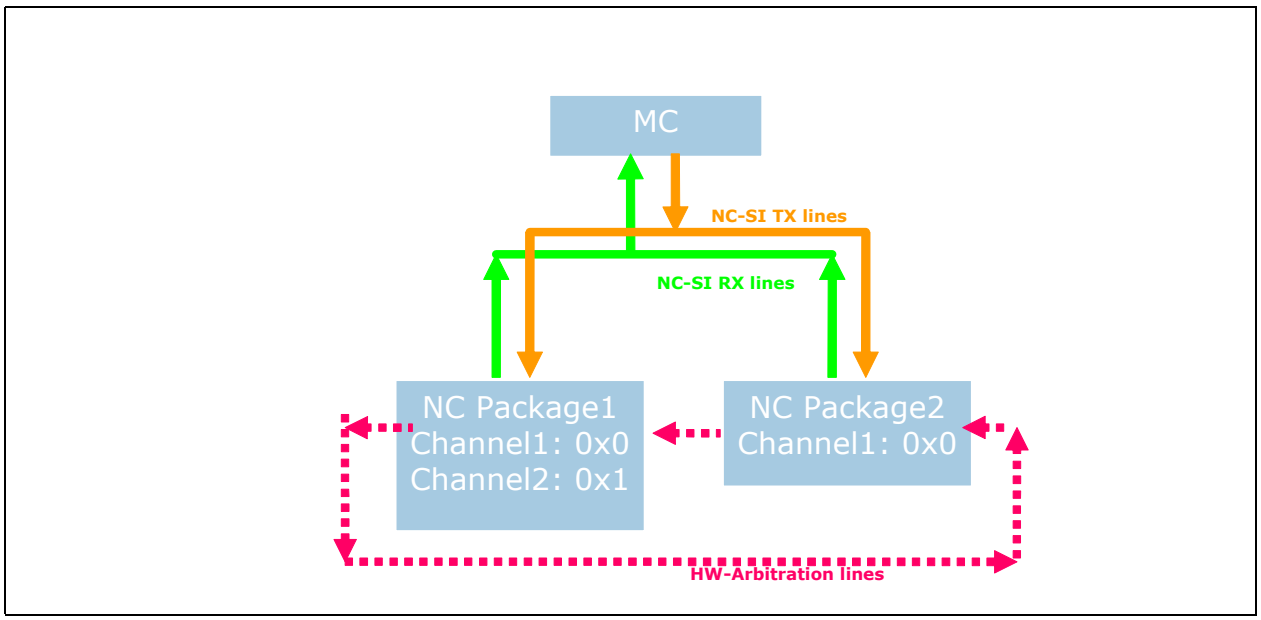


Figure 10-9. Multi-NC environment

See Figure 10-9. 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 de-selected.

NC-SI has defined two mechanisms for the arbitration scheme:

1. Package selection by the MC. In this mechanism, the MC is responsible for arbitrating between the packages by issuing NC-SI commands (Select/De-Select Package). The MC is responsible for having only one package selected at any given time.
2. Hardware arbitration. In this mechanism, two additional pins on each NC package are used to synchronize the NC package. Each NC package has an ARB_IN and ARB_OUT line and these lines are used to transfer tokens. A NC package that has a token is considered selected.

Note: Hardware arbitration is enabled by the NC-SI *HW Arbitration Enable* configuration bit in the NC-SI Configuration 1 NVM word.

For details, refer to the NC-SI specification.

10.6.9.2 Package selection sequence example

Following is an example work flow for a MC and occurs after the discovery, initialization, and configuration.

Assuming the MC needs to share the NC-SI bus between packages, the MC should:

1. Define a time-slot for each device.
2. Discover, initialize, and configure all the NC packages and channels.
3. Issue a De-Select Package command to all the channels.
4. Set active_package to 0x0 (or the lowest existing package ID).
5. At the beginning of each time slot the MC should:
   a. Issue a De-Select Package command to the active_package. The MC must then wait for a response and then an additional timeout for the package to become de-selected (200 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.

10.6.9.3 Multiple channels (fail-over)

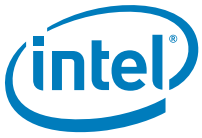
In order to support a fail-over scenario, it is required from the MC to operate two or more channels. These channels might or might not be in the same package.

The key element of a fault-tolerance fail-over scenario is having two (or more) channels identifying to the switch with the same MAC address, but only one of them being active at any given time (such as switching the MAC address between channels). To accomplish this, NC-SI provides the following commands:

1. Enable Network Tx command — This command enables shutting off the network transmit path of a specific channel. This enables the MC to configure all the participating channels with the same MAC address but only enable one of them.
2. Link Status Change AEN or Get Link Status command.

10.6.9.3.1 Fail-over algorithm example

The following is a sample workflow for a fail-over scenario for the XL710 (one package and four channels):



1. The MC initializes and configures all channels after power-up. However, the MC uses the same MAC address for all of the channels.
2. The MC queries the link status of all the participating channels. The MC should continuously monitor the link status of these channels. This can be accomplished by listening to AENs (if used) and/or periodically polling using the Get Link Status command.
3. The MC then only enables channel 0 for network transmission.
4. The MC then issues a gratuitous ARP (or any other packet with its source MAC address) to the network. This packet informs the switch that this specific MAC address is registered to channel 0's specific LAN port.
5. The MC begins normal workflow.
6. Should the MC receive an indication (AEN or polling) that the link status for the active channel (channel 0) has changed, the MC should:
   a. Disable channel 0 for network transmission.
   b. Check if a different channel is available (link is up).
   c. If found:
      - Enable network Tx for that specific channel.
      - Issue a gratuitous ARP (or any other packet with its source MAC address) to the network. This packet informs the switch that this specific MAC address is registered to channel 0's specific LAN port.
      - Resume normal workflow.
      - If not found, report the error and continue polling until a valid channel is found.

The previous algorithm can be generalized such that the start-up and normal workflow are the same. In addition, the MC might need to use a specific channel (such as channel 0). In this case, the MC should switch the network transmit to that specific channel as soon as that channel becomes valid (link is up).

Recommendations:

- Wait for a link-down-tolerance timeout before a channel is considered invalid. For example, a link re-negotiation might take a few seconds (normally 2 to 3 or might be up to 9). Thus, the link must be re-established after a short time.
- Typically, this timeout is recommended to be three seconds.
- Even when enabling and using AENs, periodically poll the link status, as dropped AENs might not be detected.

10.6.9.4 Statistics

The MC might use the statistics commands as defined in NC-SI. These counters are intended for debug purposes and are not all supported.

The statistics are divided into three commands:

1. Controller statistics — These are statistics on the network interface (to the host operating system and the PT traffic). See the NC-SI specification for details.
2. NC-SI statistics — These are statistics on the NC-SI control frames (such as commands, responses, AENs, etc.). See the NC-SI specification for details.
3. NC-SI PT statistics — These are statistics on the NC-SI PT frames. See the NC-SI specification for details.

10.6.10 External link control

The MC can use the NC-SI Set Link command to control the external interface link settings. This command enables the MC to set the auto-negotiation, link speed, duplex, and other parameters.

This command is only available when the host operating system is not present. Indicating the host operating system status can be obtained via the Get Link Status command and/or Host OS Status Change AEN command.

Recommendation:

- Unless explicitly needed, it is not recommended to use this feature. The NC-SI Set Link command does not expose all the possible link settings and/or features. This might cause issues under different scenarios. Even if you decided to use this feature, use it only if the link is down (trust the XL710 until proven otherwise).
- It is recommended that the MC first query the link status using the Get Link Status command. The MC should then use this data as a basis and change only the needed parameters when issuing the Set Link command.

For details, refer to the NC-SI specification.

10.6.10.1 Set link while LAN PCIe functionality is disabled

In cases where the XL710 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 XL710 is only used for manageability and the link speed advertisement is configured by the MC, changes in the power state of the LAN device is not effected and the link speed is not re-negotiated by the LAN device.

10.7 MCTP

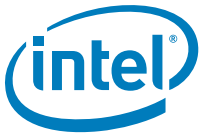
10.7.1 Management Component Transport Protocol (MCTP) overview

MCTP defines a communication model intended to facilitate communication between:

- MCs and other MCs
- MCs and management devices

The communication model includes a message format, transport description, message exchange patterns, and configuration and initialization messages.

The basic MCTP specification is described in DMTF's DSP0236 document.



MCTP is designed so that it can potentially be used on many bus types. The protocol is intended to be used for intercommunication between elements of platform management subsystems used in computer systems, and is suitable for use in mobile, desktop, workstation, and server platforms.

Currently, specifications 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.

10.7.1.1 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 XL710 supports both messages. When used only for control, then only the NC-SI (0x2) message type is supported.

In addition to the previous message types supported by the XL710, the PCIe based VDM message type is also supported over PCIe to support ACL commands.

Details of the NC-SI over MCTP can be found in Section 10.7.5.

10.7.1.2 MCTP usage model

The XL710 supports NC-SI over MCTP protocol over the PCIe and SMBus busses. The XL710 can connect through MCTP to a MC or the ME engine in the chipset as described in Figure 10-10.

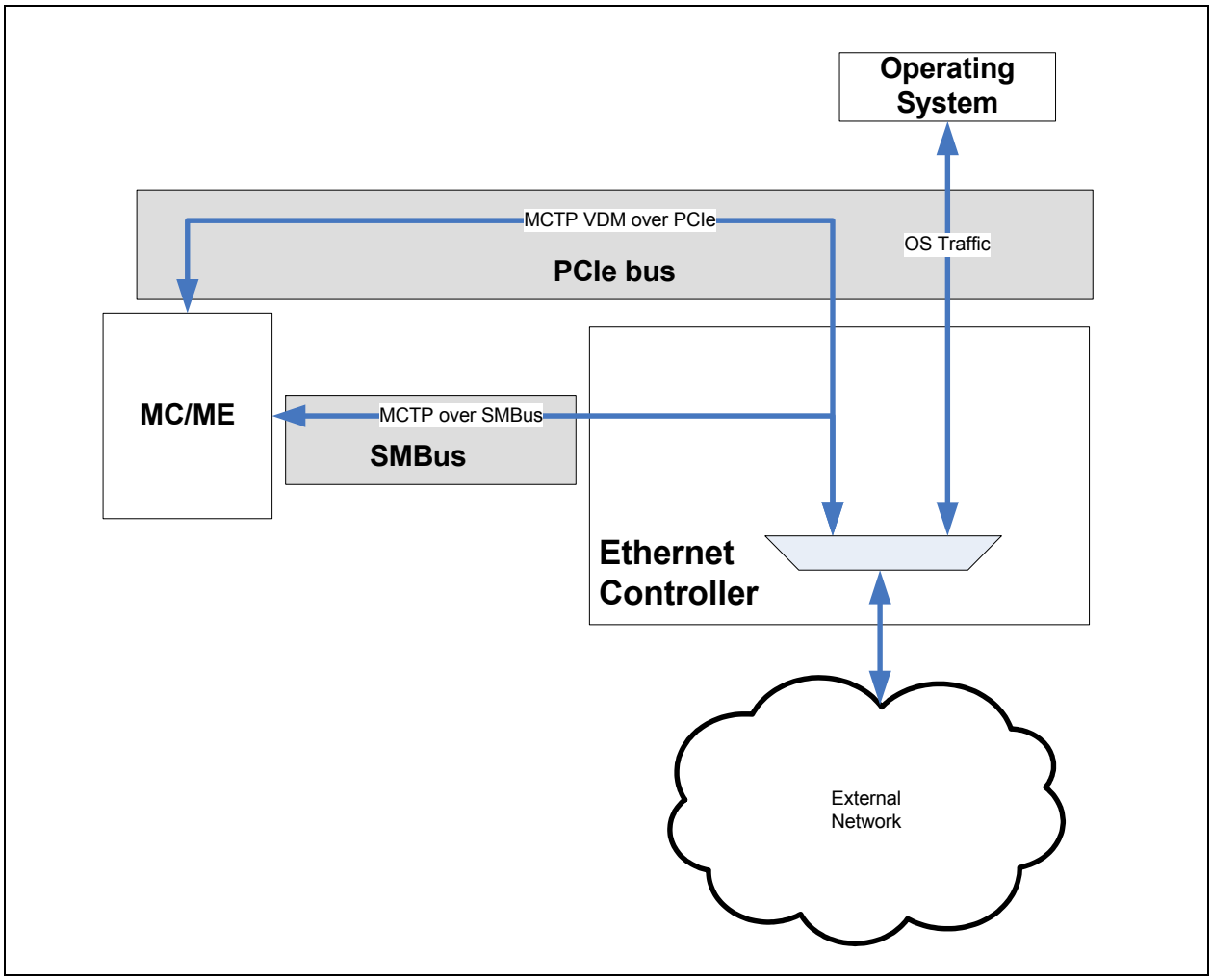


Figure 10-10. the XL710 MCTP connections

10.7.2 NC-SI to MCTP mapping

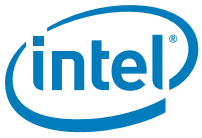
The four network ports of the XL710 (mapped to two NC-SI channels) are mapped to a single MCTP endpoint on SMBus and to another endpoint over PCIe.

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 function 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 10.7.2.1 describes the transition between the two busses.



Both endpoints (SMBus and PCIe) might be active concurrently. However, PT traffic might be transferred only through one of them. If the PCIe endpoint is active, it is used for PT traffic; otherwise, the SMbus endpoint is used. The Set EID command can be used to force the transition for the PCIe endpoint to the SMBus endpoint if the bus owner determines the PCie channel is not functional.

For each channel (SMBus or PCIe), the XL710 should expect MCTP commands from two sources: the bus owner and the MC. In addition, it should expect PT traffic through one interface only. Thus, it should be able to process up to five interleaved commands/data:

- An MCTP control/OEM command from the PCIe bus owner (single packet message).
- An MCTP control/OEM command from the SMBus bus owner (single packet message).
- An MCTP control/OEM command from the MC over SMBus (single packet message).
- An MCTP control/OEM command from the MC over PCIe (single packet message).
- An NC-SI command or Ethernet packet from the MC over the active channel.

A single source should not interleave packets it sends.

The topology used for MCTP connection is shown in Figure 10-11.

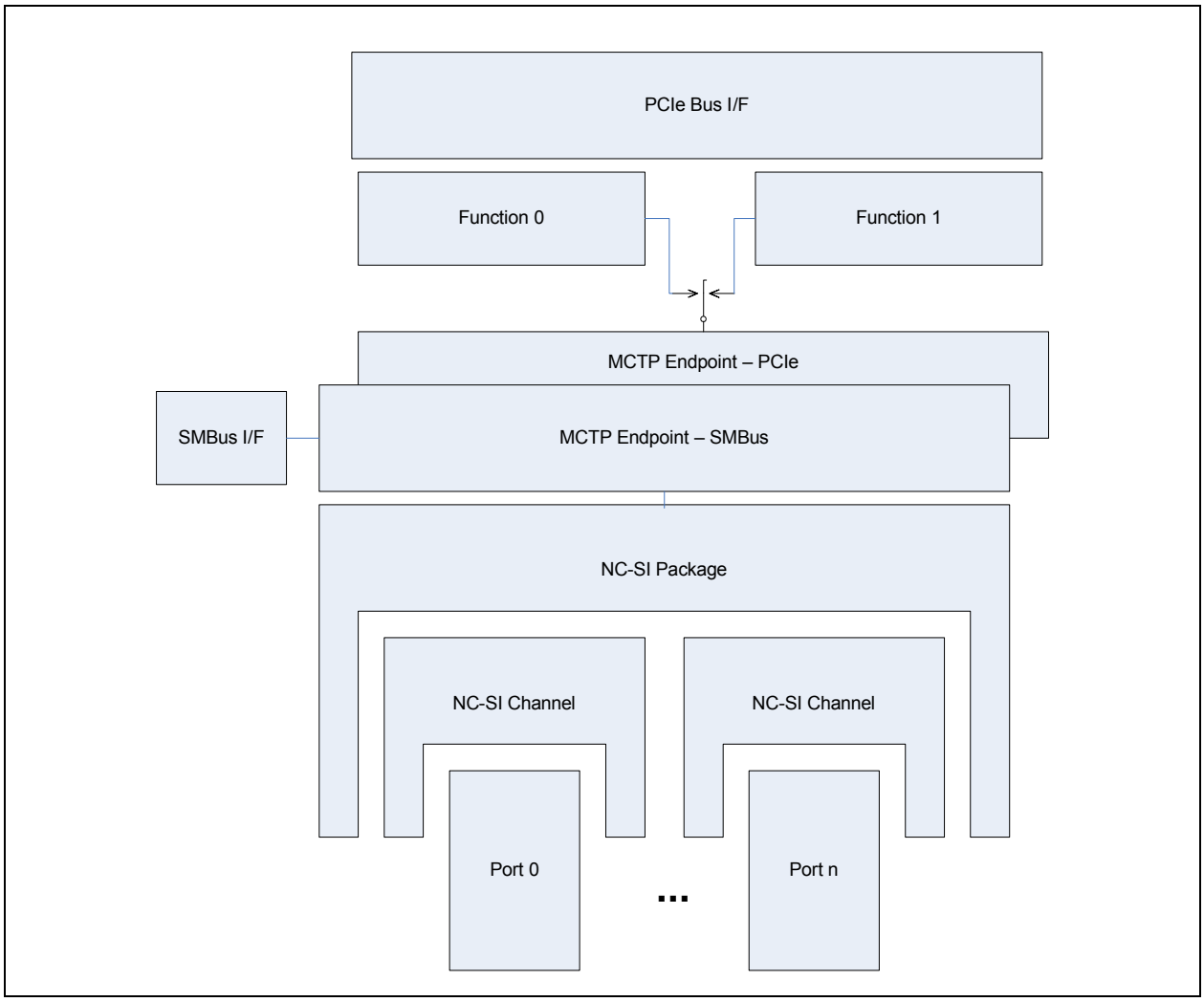


Figure 10-11. MCTP endpoints topology

10.7.2.1 Detecting an MC EID and physical address

In order to enable transactions between the MC and the NIC, the bus physical address (SMBus or PCIe) and the EID of the partner needs to be discovered. NICs don't try to discover the MC and assume the MC initiates the connection. If the NIC is in an NC-SI initial state, then the EID and the physical address of the MC are extracted from the Clear Initial State command parameters or any other NC-SI command received later with a channel ID of the XL710. Subsequent PT traffic is received from or sent to this address only.

If the EID or the physical address of the NIC changes, it indicates the changes to bus owner so that the routing tables can be updated. There is no attempt to directly send an indication to the MC about the change.

See more details in next section.

10.7.2.2 Bus transition

The following section defines the transition flow between PCIe and SMBus as the bus on which MCTP flows. Figure 10-12 describes the flow to transition between PCIe and SMBus. The following parameters are used to define the flow:

- NIC EID on PCIe
- NIC EID on SMBus
- NIC PCIe Target ID
- Bus Owner EID on PCIe
- Bus Owner EID on SMBus
- Bus Owner PCIe Target ID
- Bus Owner SMBus Address
- MC EID on PCIe
- MC EID on SMBus
- MC PCIe Target ID
- MC SMBus Address
- NIC SMBus Address

All these variables are initialized to zero at power on apart from the SMBus address of the endpoint (NIC), which might be initialized from a NVM value.

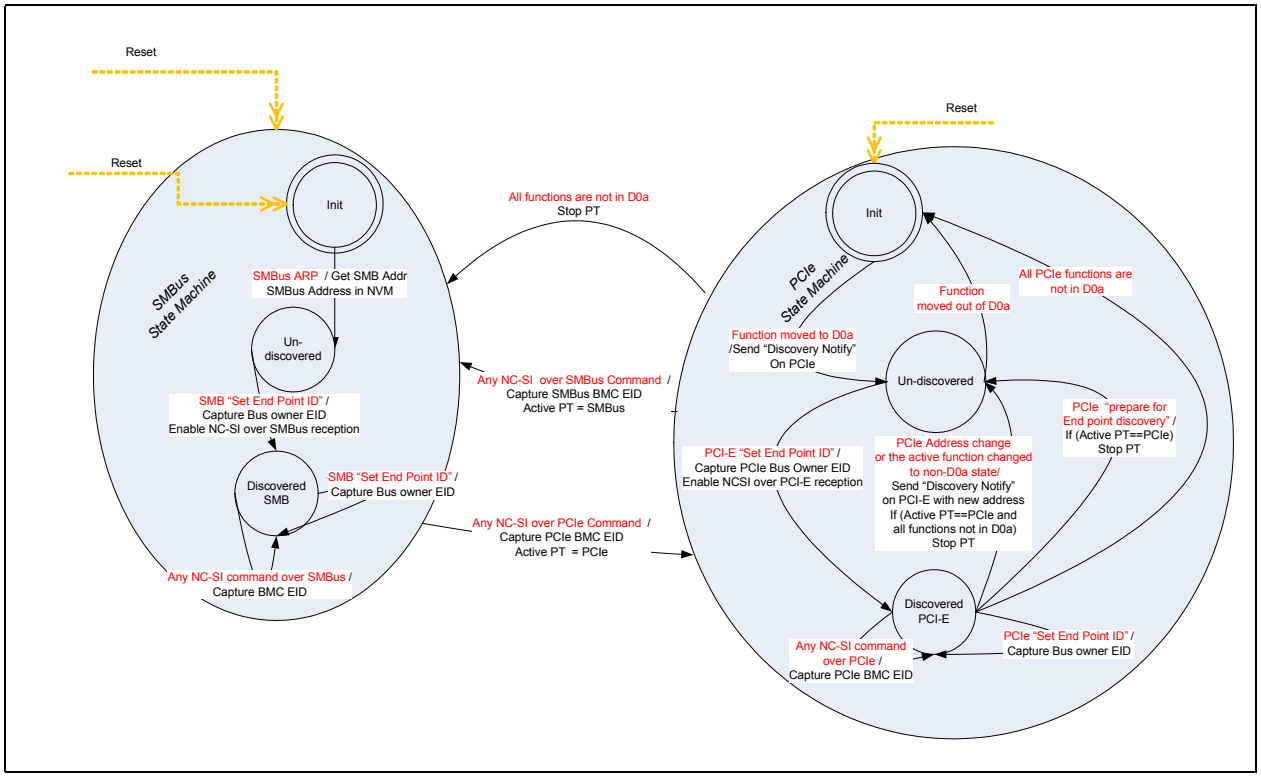


Figure 10-12. MCTP bus transition state machine

10.7.2.2.1 Initial assignment flow

- At power on, the NIC or MC MCTP channel is connected to the SMBus, is not assigned an EID and is in an undiscovered state.
- The bus owner might preform an SMBus ARP cycle to assign an SMBus address to the NIC or to the MC. Otherwise, a fixed address might be used. It is assumed that the SMBus address does not change after initialization time.
- The bus owner performs an EID assignment using a Set Endpoint ID MCTP command. The NIC or the MC captures the SMBus address of the bus owner from the *SMBus Source 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.
- The MC might detect the NIC EID using one of the two following modes:
  - Static configuration of the NIC SMBus address in the MC database and Get Routing Table Entries command to find the EID matching the SMBus address.
  - Get all endpoints through a Get Routing Table Entries command and find endpoints supporting NC-SI using the Get Message Type Support command for each endpoint.
- Once the NIC is found, the MC might send a Clear Initial State command to the NIC to start the NC-SI configuration. The NIC captures the MC SMBus address and MC EID from any NC-SI command received.
- After the NC-SI channels are enabled, traffic might be sent using the MC and NIC addresses previously discovered.
- The MC might also send a Get UUID command to get a unique identifier of the NIC that might be used later for re-connection upon topology changes.

10.7.2.2.2 SMBus to PCIe transition

- If the NIC or the MC detects that the PCIe bus is available by detecting a function that moved to D0a 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.
- After receiving the Discovery Notify MCTP command on the PCIe bus, the bus owner sends a Set Endpoint ID MCTP command on the PCIe bus and updates the routing table. The bus owner might choose to wait for the Discovery Notify MCTP command of both the MC and the NIC to do the transition. The bus owner should try to keep the EID previously assigned on the SMBus as the EID on PCIe bus.
- After receiving the Set Endpoint ID MCTP command, the NIC waits for an NC-SI command from the MC indicating it is ready to transition the connection to PCIe. After receiving such a command, the NIC transitions its PT traffic to the PCIe bus using the newly received addresses.
- The MC on its side, needs to discover the PCIe address of the NIC. This can be done using the Resolve Endpoint ID command if only the physical address changed or using the Resolve Endpoint UUID command also if both EID and physical address changed. It can then send an NC-SI command to the NIC to initiate the transition. The MC should not send any PT packets from the moment it sent the first NC-SI command on the PCIe and the moment a response is received for this command.
- The transition of NC-SI traffic (PT or commands/responses) from SMBus to PCIe should be done on a packet boundary and should not interrupt a packet fragmentation or reassembly.

10.7.2.2.3 PCIe target ID change

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 D0a state). In this case, the following flow should be used:

- The endpoint should send a Discovery Notify MCTP command on the PCIe bus using the new Requester ID.
- After receiving the Discovery Notify MCTP command with the new requester ID, the bus owner sends a Set Endpoint ID MCTP command on the PCIe bus and updates the routing table. The bus owner should try to keep the EID previously assigned on the SMBus as the EID on the previous requester ID.
- The bus owner sends an Routing Information Update command to all supporting endpoints that might then update the parameters of their counterpart they use.

10.7.2.2.4 PCIe to SMBus transition

- If the NIC or the MC detects that the PCIe bus is not available by detecting a transition of all functions to a non D0a state, it stops using the PCIe for PT traffic or NC-SI traffic.
- Upon detection of the unavailability of the PCIe bus, the MC transitions the NC-SI channel to the MCTP over SMBus as previously described.
- The transition of NC-SI traffic (PT or commands/responses) from PCIe to SMBus might done at any stage and might interrupt a packet fragmentation or reassembly, as it is assumed that such a transition occurs only when the PCIe bus is not available anymore.

10.7.3 MCTP over PCIe

10.7.3.1 Message format

The message format used for NC-SI over MCTP over PCIe is as follows:

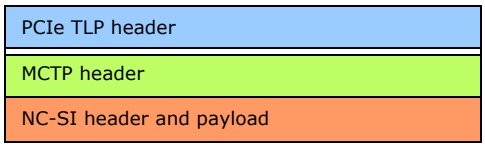


Table 10-76. NC-SI/Ethernet over MCTP over PCIe message format

+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 r[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)															

Table 10-76. 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
MCTP Reserved				Header version = 1				Destination endpoint ID								Source endpoint ID								SOM	EOM	SEQ#			TO	Tag			
IC[2]	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, IC = 0,TH = 0, Attr[2:0] = 0 for sent packets and is ignored for received packets.

10.7.3.2 PCIe discovery process

The XL710 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 XL710 exposes the endpoint on the selected function as previously described.

If the selected function moves to D0u after the endpoint was discovered, or if the bus number of the XL710 changes due to a re-enumeration of the bus, the XL710 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.

10.7.3.3 MCTP over PCIe special features

The XL710 supports the following optional features of MCTP when running over PCIe:

1. Rate limiting
2. ACLs

10.7.3.3.1 MCTP uplink rate limiting

As the PCIe link can carry a traffic bandwidth much higher than what the MC can sustain, in order to avoid drop of packets, the XL710 allows rate limiting of the MCTP PT traffic. The XL710 supports rate limiting between 1 Mb/s and 1 Gb/s. The following parameters define the behavior of the rate limiter:

- Max rate limit (fixed from NVM via the MCTP rate in the MCTP rate limiter config 1 word).
- The max burst size (fixed from NVM via the *MCTP max credits* field in the in the MCTP rate limiter config 2 word).To limit the max burst to one VDM, set this parameter to 5.
- Decision point (fixed from NVM via the *decision point* field in the in the MCTP rate limiter config 2 word).

10.7.3.3.2 Service provider MCTP endpoint ACLs

The XL710 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 (using the TBD NVM bit) the following flow is used decide which packets are accepted.

Commands can be divided to three types:

1. ACL programming commands: Such commands can be received only from the address that sent the *Prepare For Endpoint Discovery* command via broadcast routing.
2. 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, then the SPMEACLD counter is increased. This counter can be read by the MCTP bus owner using the Get ACL Violation Counters command.
3. 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 XL710 supports 4 ACL entries.

10.7.4 MCTP Over SMBus

The message format used for NC-SI over MCTP over SMBus is as follows:

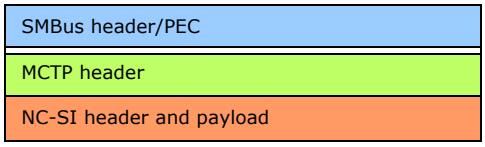


Table 10-77. NC-SI/Ethernet over MCTP over SMBus message format

+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.

10.7.4.1 SMBus discovery process

The XL710 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 10.5.5.4). It responds to any SMBus command using the MCTP command code. This ensures that the bus owner knows the XL710 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.

10.7.4.2 MCTP over SMBus special features

The XL710 supports the following optional feature of MCTP when running over SMBus:

1. Simplified MCTP mode.
2. Fairness arbitration.

10.7.4.2.1 Simplified MCTP mode

For some point-to-point implementations of MCTP, the assembly process is simplified. In this mode, the destination EID, source EID, packet sequence number, Tag Owner (TO) bit and message tag are ignored and the assembly is based only on the *SOM* and *EOM* bits. This bit is set according to the *Simplified MCTP* bit in the MCTP configuration word in the NVM.

Note: This mode is not compliant with the MCTP specification.

In this mode, a Set EID command is not needed to start operation.

This mode is relevant only for MCTP over SMBus traffic and when the Redirection Sideband Interface is set to 10b (MCTP over SMBus only - no pass through).

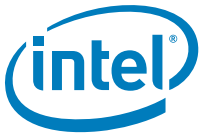
10.7.4.2.2 Fairness arbitration

When sending MCTP messages over SMBus and when fairness arbitration is enabled (see Section 7.2.34.3), the XL710 should adhere to the fairness arbitration as defined in section 5.13 of DSP0237 when sending MCTP messages.

10.7.5 NC-SI over MCTP

The XL710 support for NC-SI over MCTP is similar to the support for NC-SI over RBT with the following exceptions:

1. A set of new NC-SI OEM commands used to expose the NC-SI over MCTP capabilities.
2. The format of the packets is modified to account for the new transport layer as described in the sections that follow.



10.7.5.1 NC-SI packets format

NC-SI over MCTP defines two different message type for pass through and for control packets.

Packets with a message type equal to the *Control packets message type* field (default = 0x02) in the NVM are NC-SI control packets (commands, responses and AENs) and packets with a message type equal to the *Pass through packets message type* field (default = 0x03) in the NVM are NC-SI pass through packets

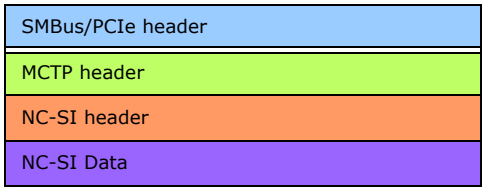
10.7.5.1.1 Control packets

The format used for control packets (commands, responses and AENs) is as follows:

Table 10-78. 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 10.2.2.2.



Note that the MAC header and MAC FCS present when working over NC-SI are not part of the packet in MCTP mode.

10.7.5.1.2 PT packets

The format used for PT 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 XL710 in transmit, the CRC is added by the XL710.

Table 10-79. Ethernet over MCTP over PCIe/SMBus message format

+0								+1								+2								+3							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SMBus or PCIe header																															
MCTP Reserved				Header version = 1				Destination endpoint ID								Source endpoint ID								SOM		EOM		SEQ#		TO = 1	Tag
IC = 0	Message Type = Pass Through Packets Control Type							DA																							
DA																								SA							
SA																															
SA								Ether type																Ethernet Packet							
Ethernet packet																															
....																															
....																															

10.7.6 MCTP programming

The MCTP programming model is based on:

1. A set of MCTP commands used for the discovery process and for the link management. The list of supported commands is described in section Section 10.7.6.1.
2. A subset of the NC-SI commands used in the regular NC-SI interface, including all the OEM commands as described in Section 10.6.2 (NC-SI programming I/F). The specific commands supported are listed in Table 10-28 and Table 10-32.

Note: For all MCTP commands (both native MCTP commands and NCSI over MCTP), the response uses the Msg tag received in the request with *TO* bit cleared.

10.7.6.1 MCTP commands support

Table 10-80 lists the MCTP commands supported by the XL710.

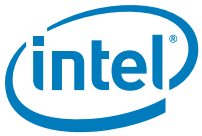


Table 10-80. MCTP commands support

Command Code	Command Name	General Description	XL710 Support As Initiator	XL710 Support As Responder
0x00	Reserved	Reserved	–	–
0x01	Set Endpoint ID	Assigns an EID to the endpoint at the given physical address.	N/A	Yes
0x02	Get Endpoint ID	Returns the EID presently assigned to an endpoint. Also returns information about what type the endpoint is and its level of use of static EIDs. See Section 10.7.6.1.1 for details.	No	Yes
0x03	Get Endpoint UUID	Retrieves a per-device unique UUID associated with the endpoint. See Section 10.7.6.1.2 for details.	No	Yes
0x04	Get MCTP Version Support	Lists which versions of the MCTP control protocol are supported on an endpoint. See Section 10.7.6.1.3 for details.	No	Yes
0x05	Get Message Type Support	Lists the message types that an endpoint supports. See Section 10.7.6.1.4 for details.	No	Yes
0x06	Get Vendor Defined Message Support	Used to discover an MCTP endpoint's vendor specific MCTP extensions and capabilities. See Section 10.7.6.1.5 for details.	No	Yes[1]
0x07	Resolve Endpoint ID	Used to get the physical address associated with a given EID.	No	N/A
0x08	Allocate Endpoint IDs	Used by the bus owner to allocate a pool of EIDs to an MCTP bridge.	N/A	N/A
0x09	Routing Information Update	Used by the bus owner to extend or update the routing information that is maintained by an MCTP bridge.	N/A	N/A
0x0A	Get Routing Table Entries	Used to request an MCTP bridge to return data corresponding to its present routing table entries.	No	N/A
0x0B	Prepare for Endpoint Discovery	Used to direct endpoints to clear their discovered flags to enable them to respond to the Endpoint Discovery command.	N/A	Yes[1]
0x0C	Endpoint Discovery	Used to discover MCTP-capable devices on a bus, provided that another discovery mechanism is not defined for the particular physical medium.	No	Yes[1]
0x0D	Discovery Notify	Used to notify the bus owner that an MCTP device has become available on the bus.	Yes[1]	N/A
0x0E	Get Network ID	Used to get the MCTP network ID.	No	No
0x0F	Query Hop	Used to discover what bridges, if any, are in the path to a given target endpoint and what transmission unit sizes the bridges will pass for a given message type when routing to the target endpoint.	No	No

1. These commands are supported only for MCTP over PCIe.

10.7.6.1.1 Get endpoint ID

The get endpoint ID response of the XL710 is listed in the following table:

Byte	Description	Value
1	Completion Code	
2	Endpoint ID	0x00 - EID not yet assigned. Otherwise - returns EID assigned using Set Endpoint ID command.
3	Endpoint Type	0x00 (Dynamic EID, simple endpoint).
4	Medium Specific	SMBus: 0x01 - Fairness arbitration protocol supported. PCIe: 0x00.

10.7.6.1.2 Get endpoint UUID

The UUID returned is calculated according to the following function:

- Time Low = Read from NVM words at offset 0x9 and 0xA of the sideband configuration structure.
- Time mid = Read from NVM word at offset 0xB of the sideband configuration structure.
- Time High and version = Read from NVM word at offset 0xC of the sideband configuration structure.
- Clock Sec and Reserved = Read from NVM word at offset 0xD of the sideband configuration structure.
- Node = MAC address as taken from the *GLPCI_SERL* and *GLPCI_SERH* registers.

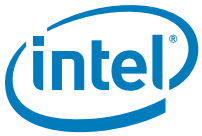
10.7.6.1.3 Get MCTP version support

The following table lists the returned value according to the requested message type.

Byte	Description	Message Type					
		0xFF(Base)	0x00 (Control Protocol Message)	0x02 (NC-SI Over MCTP)	0x03 (Ethernet)	0x7E (PCIe Based VDM Messages)	All Other Or Unsupported Messages
1	Completion Code	0x0					0x80
2	Version Number entry count	3	3	1	1	2	0
6:3	Version number entry	0xF1F0FF00 (1.0)	0xF1F0FF00 (1.0)	0xF1F0FF00 (1.0)	0xF1F0FF00 (1.0)	0xF1F0FF00 (1.0)	0
9:7	Version number entry 2	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)			0xF1F1F000 (1.1.0)	
13:10	Version number entry 3	0xF1F2F000 (1.2.0)	0xF1F2F000 (1.2.0)				

10.7.6.1.4 Get message type support command

The get message type support response of the XL710 is listed in the following table:



Byte	Description	Value
1	Completion Code	0x00.
2	MCTP Message Type Count	0x01/0x02/0x03 - The XL710 supports up to three additional message types, depending on the mode of operation and the bus used.
3:5	List of Message Type Numbers	0x02 (NC-SI over MCTP).
		0x03 (Ethernet). If PT is supported.
		0x7E (PCIe based VDM messages) - over PCIe only.

10.7.6.1.5 Get vendor defined message support command

The get vendor defined message type support response of XL710 is listed in the following table if vendor ID set selector equals 0x00:

Byte	Description	Value
1	Completion Code	0x00.
2	Vendor ID Set Selector	0xFF = No more capability sets.
2:4	Vendor ID	0x008086 (PCI ID indicator + Intel vendor ID).
5:6	Version	0x0100 (version 1.0).

10.7.6.1.6 Set endpoint ID command

The XL710 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 XL710 can be set only through their own interface, set EID and force EID are equivalent. The Reset EID operation is not supported by the XL710.

The Set Endpoint ID response of the XL710 is described in the following table:

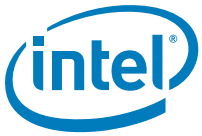
Byte	Description	Value
1	Completion Code	0x00.
2	Completion Status	[7:6] = 00b - Reserved.
		[5:4] = 00b - EID assignment accepted.
		[3:2] = 00b - Reserved.
		[1:0] = 00b - Device does not use an EID pool.
3	EID Setting	If the EID setting was accepted, this value matches the EID passed in the request. Otherwise, this value returns the present EID setting.
4	EID Pool Size	Always return a zero.

10.8 Host isolate support

If a MC decides that malicious software prevents its usage of the LAN, it might decide to isolate the NIC from its driver. This is done using the TCO reset command (Section 10.6.4.12).

If TCO isolate is enabled in the NVM (Section 7.2.31.4), The TCO Isolate command disables PCIe write operations to the LAN port. As the software device driver needs to access the CSR space in order 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.



11.0 Programming interface

11.1 Introduction

This section details the programmer visible state inside the XL710. In some cases, it describes hardware structures invisible to software in order to clarify a concept.

The XL710 address space is mapped into four regions with PCI Base Address registers described in Section 12.2.6.1. These regions are listed in the following table.

Table 11-1. Address space regions

Addressable Content	How Mapped	Size of Region
Memory BAR (Internal registers, memories and Flash)	Direct memory-mapped	4 MB - 16 MB
I/O BAR (optional Internal registers)	I/O Window mapped	32 bytes[1]
MSI-X BAR (optional)	Direct memory-mapped	32 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 responded 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.

11.1.1 Access mechanisms

11.1.1.1 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 12.2.6.1).

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.

11.1.1.2 Memory-mapped access 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 12.2.6.1). See Table 12-6 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 3.3 for details on accessing the NVM.

11.1.1.3 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 12.2.6.1). See Section 12.3.3 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.

11.1.1.4 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 12.2.6.2) 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.

11.1.1.5 I/O-mapped access to internal registers

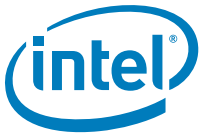
To support pre-boot operation, all internal registers can be accessed using I/O operations. I/O accesses are supported only if an I/O base address is allocated and mapped (BAR2; see Section 12.2.6.1), and I/O address decoding is enabled in the PCIe configuration.

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 11-2.

Table 11-2. IOADDR and IODATA in I/O address space

Offset	Abbreviation	Name	RW	Size
0x00	IOADDR	Internal Register Address. Covers the (4 MB - 64 KB) CSR space 0x00000-0x3EFFFF – 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

11.1.1.5.1 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.

For IA 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.

11.1.1.5.2 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.

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).

Note: There are no special software timing requirements on accesses to IOADDR or IODATA. All accesses are immediate, except when data is not readily available or acceptable. In this case, the XL710 delays the results through normal bus methods (for example, split transaction or transaction retry).

Note: 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.

11.1.1.5.3 Undefined I/O offsets

I/O offsets 0x08 through 0x1F are considered to be reserved offsets with the I/O window. Write accesses to these addresses are discarded. Read accesses to theses addresses return undefined content. .

11.1.1.6 Configuration access to internal registers

To support legacy pre-boot 16-bit operating environments without requiring I/O address space, the XL710 enables accessing CSRs via configuration address space by mapping the *IOADDR* and *IODATA* registers into 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 mappings in this case are listed in Table 11-3.

Table 11-3. IOADDR and IODATA in configuration address space

Configuration Address	Abbreviation	Name	RW	Size
0x98	IOADDR	Internal Register Address. Covers the (4 MB - 64 KB) CSR space. 0x00000-0x3EFFFF – Internal registers. 0x3F0000-0x7FFFFF – 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 configuration space in the following manner:

1. CSR address is written to IOADDR where:
   a. Bit 31 (*IOADDR.Configuration IO Access Enable*) of IOADDR should be set to 1b.
   b. 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 un-intentional CSR read operations (that might cause clear by read) by other applications scanning the configuration space.

Software reads data from an internal CSR via configuration space in the following manner:

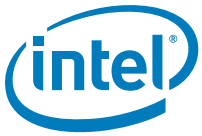
1. The CSR address is written to IOADDR where:
   a. Bit 31 (*IOADDR.Configuration IO Access Enable*) of IOADDR should be set to 1b.
   b. Bits 30:0 of IOADDR should hold the actual address of the internal register being read.
2. The CSR value is read from IODATA.
   a. IODATA is used as a window to 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 un-intentional CSR read operations (that might cause clear by read) by other applications scanning the configuration space.

Note: 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.

Note: When a function is in D3 state, software should not attempt to access CSRs via IOADDR and IODATA.

Note: 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.

Note: Bit 31 of IOADDR (*IOADDR.Configuration IO Access Enable*) has no effect when initiating access via IO address space.



11.1.2 Memory BAR

11.1.2.1 PF BAR structure

The memory BAR provides access to internal CSRs 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.
- Flash *Size* field from the NVM (or the GLPCI_LBARCTRL.FL_SIZE CSR field) - size is calculated as 64 KB * (2 ** Flash Size). Default value is 8 MB

Table 11-4 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 (4 MB-64 KB-1) (such as the first 4 MB - 64 KB)
- The Flash space (if exposed) is always aligned to multiples of its size

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 11-4. Structure of the PF memory BAR

Flash Expose	Flash Size	Flash Space		BAR Size
		Min Addr	Max Addr	
0	x	x	x	4 MB
"	"	x	x	4 MB
"	"	x	x	8 MB
1	2 MB	4 MB	6 MB	8 MB
"	"	4 MB	6 MB	8 MB
"	"	8 MB	10 MB	16 MB
"	4 MB	4 MB	8 MB	8 MB
"	"	4 MB	8 MB	8 MB
"	"	8 MB	12 MB	16 MB
"	8 MB	8 MB	16 MB	16 MB
"	"	8 MB	16 MB	16 MB
"	"	8 MB	16 MB	16 MB

11.1.2.2 VF BAR Structure

The VF memory BAR provides access to on-die CSRs for VFs.

The size of the VF BAR is a maximum page size of 64 KB.

11.1.3 The MSI-X BAR

The structure of the MSI-X BAR is described in Section 12.3.3 (for a PF) and Section 12.5.3.1 (for a VF).

11.1.4 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)

11.1.4.1 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. The following table lists the different scopes.

Table 11-5. 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	4	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	4	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	16	Each PF has access to its copy only	
VF	PF, VF	128	Each PF has access to the registers of its VFs only Each VF has access to its copy only	
VP	PF	128	Each PF has access to the registers of its VFs only	These registers control VF functionality

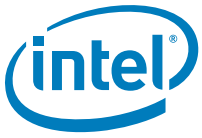


Table 11-5. Scope mapping

Scope	PF/VF	Quantity	Exposure	Comments
Q	PF, VF	1536	Each PF has access to the registers allocated to it (including. its VFs) Each VF has access to the registers allocated to it	
VSI	PF	384	All PFs	Registers are shared by all PFs
INTPF	PF	512	Each PF has access to the registers allocated to it[1]	
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]	
INTVP	PF	512	Each PF has access to the registers of its VFs only[1]	These registers handle VF interrupts

1. Note that the register descriptions in section Device Registers - PF and section Device Registers - VF list 512 instances per register. However, the actual number of registers exposed to a function is listed in Table 7-139.

11.1.5 Register conventions

All registers in the XL710 are defined to be 32 bits, 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 11.1.1.5).
- Accesses to BAR0 of a VF might be byte, word or Dword accesses. 64-bit (Qword) accesses to this BAR are completed with an 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 should be 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 in order for proper operation to occur; this need is dependent on the function of the bit. Other registers might cite a hardware default which 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.

Note: 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.

Note: 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.

11.1.6 Register terminologies

The following table 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.

Abbreviation	Description
RO	Read Only - A register bit field with this attribute can be read. Writes have no effect on the bit field value.
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.
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.

11.2 Device Registers - PF

11.2.1 BAR0 Registers Summary

Table 11-6. BAR0 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Page
PF - General Registers			
0x00074400 + 0x4*VF, VF=0...127	VFGEN_RSTAT[VF]	VF Reset Status	1323
0x00083048	GL_FWSTS	Firmware Status Register	1323
0x00088000	PFGEN_STATE	PF State	1324
0x00088100 + 0x4*n, n=0...29	GLGEN_GPIO_CTL[n]	Global GPIO Control	1324
0x00088178	GLGEN_LED_CTL	Global LED Control	1326
0x0008817C	GLGEN_GPIO_STAT	Global GPIO Status	1326
0x00088180	GLGEN_GPIO_TRANSIT	Global GPIO Transition Status	1327
0x00088184	GLGEN_GPIO_SET	Global GPIO Set	1327
0x0008818C + 0x4*n, n=0...3	GLGEN_MSCA[n]	MDI Single Command and Address	1327
0x0008819C + 0x4*n, n=0...3	GLGEN_MSRWD[n]	MDI Single Read and Write Data	1328
0x000881AC + 0x4*n, n=0...3	GLGEN_I2CPARAMS[n]	I^2C Parameters	1328
0x000881BC	GLVFGEN_TIMER	Global Device Timer	1329
0x000881C0 + 0x4*n, n=0...3	GLGEN_MDIO_I2C_SEL[n]	Global MDIO or I^2C Select	1329
0x000881D0 + 0x4*n, n=0...3	GLGEN_MDIO_CTRL[n]	MDIO Control	1330
0x000881E0 + 0x4*n, n=0...3	GLGEN_I2CCMD[n]	I^2C Command	1331
0x00090000 + 0x4*VSI, VSI=0...383	VSIGEN_RTRIG[VSI]	VM Reset Trigger	1331
0x00090800 + 0x4*VSI, VSI=0...383	VSIGEN_RSTAT[VSI]	VM Reset Status	1332
0x00091800 + 0x4*VF, VF=0...127	VPGEN_VFRTRIG[VF]	VF Reset Trigger	1332
0x00091C00 + 0x4*VF, VF=0...127	VPGEN_VFRSTAT[VF]	VF Reset Status	1332
0x00092400	PFGEN_CTRL	PF Control	1332
0x00092500	PFGEN_DRUN	PF Driver Unload	1332
0x00092600 + 0x4*n, n=0...3	GLGEN_VFLRSTAT[n]	Global VF Level Reset Status	1333
0x000B612C	GLGEN_STAT	Global Status	1333
0x000B8100	PRTGEN_STATUS	General Port Status	1333
0x000B8120	PRTGEN_CNF	General Port Configuration	1334
0x000B8160	PRTGEN_CNF2	General Port Configuration2	1334
0x000B8180	GLGEN_RSTCTL	Global Reset Delay	1335
0x000B8184	GLGEN_CLKSTAT	Global Clock Status	1335
0x000B8188	GLGEN_RSTAT	Global Reset Status	1336
0x000B818C	GLGEN_RSTENA_EMP	Global EMP Reset Enable	1337
0x000B8190	GLGEN_RTRIG	Global Reset Trigger	1337
0x001C0480	PFGEN_PORTNUM	LAN Port Number	1337
0x001C0AB4	GLGEN_PCIFCNCNT	PCI Function Count	1337

Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
PF - PCIe Registers			
0x0009C000	PF_FUNC_RID	Function Requester ID Information Register	1338
0x0009C080	PF_PCI_CIAA	PF PCIe Configuration Indirect Access Address	1338
0x0009C100	PF_PCI_CIAD	PCIe Configuration Indirect Access Data	1338
0x0009C180	PFPCI_FACTPS	Function Active and Power State	1338
0x0009C200	PFPCI_ICAUSE	PCIe Interrupt Cause	1339
0x0009C280	PFPCI_IENA	PCIe Interrupts Enable	1339
0x0009C300	PFPCI_VMINDEX	PCIe VM Pending Index	1339
0x0009C380	PFPCI_VMPEND	PCIe VM Pending Status	1339
0x0009C480	GLPCI_DREVID	PCIe Default Revision ID	1339
0x0009C484	GLPCI_BYTCTH	PCIe Byte Counter High	1340
0x0009C488	GLPCI_BYTCTL	PCIe Byte Counter Low	1340
0x0009C48C	GLPCI_GSCL_1	PCIe Statistic Control Register #1	1340
0x0009C490	GLPCI_GSCL_2	PCIe Statistic Control Register #2	1341
0x0009C494 + 0x4*n, n=0...3	GLPCI_GSCL_5_8[n]	PCIe Statistic Control Registers #5...#8	1341
0x0009C4A4 + 0x4*n, n=0...3	GLPCI_GSCN_0_3[n]	PCIe Statistic Counter Registers #0...#3	1341
0x0009C4BC	GLPCI_PKTCT	PCIe Packet Counter	1342
0x0009C4EC	GLPCI_PQ_MAX_USED_SPC	PCIe PQs Max Used Space	1342
0x0009C4F0	GLPCI_PM_MUX_PFB	PCIe Mux Selector for PFB	1342
0x0009C4F4	GLPCI_PM_MUX_NPQ	PCIe Mux Selector for NPQs	1342
0x0009C4F8	GLPCI_SPARE_BITS_0	PCIe Regs Spare Bits 0	1342
0x0009C4FC	GLPCI_SPARE_BITS_1	PCIe Regs Spare Bits 1	1343
0x0009C600 + 0x4*VF, VF=0...127	PFPCI_VF_FLUSH_DONE[VF]	PCIe VF Flush Done	1343
0x0009C800	PFPCI_PF_FLUSH_DONE	PCIe PF Flush Done	1343
0x0009C880	PFPCI_VM_FLUSH_DONE	PCIe VM Flush Done	1343
0x000BE000	PFPCI_CNF	PCIe PF Configuration	1343
0x000BE080	PFPCI_DEVID	PCIe PF Device ID	1344
0x000BE100	PFPCI_SUBSYSID	PFPCIe Subsystem ID	1344
0x000BE180	PFPCI_FUNC2	PCIe Functions Configuration 2	1344
0x000BE200	PFPCI_FUNC	PCIe Functions Configuration	1344
0x000BE280	PFPCI_STATUS1	PCIe Function Status 1	1345
0x000BE300	PFPCI_PM	PCIe PM	1345
0x000BE400	PFPCI_CLASS	PCIe Storage Class	1345
0x000BE480	GLTPH_CTRL	TPH Control Register	1346
0x000BE484	GLPCI_LBARCTRL	PCI BAR Control	1346
0x000BE48C	GLPCI_SUBVENID	PCIe Subsystem ID	1347
0x000BE490	GLPCI_PWRDATA	PCIe Power Data Register	1347

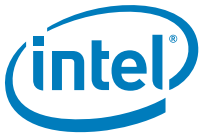


Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x000BE494	GLPCI_CNF2	PCIe Global Config 2	1347
0x000BE498	GLPCI_SERL	PCIe Serial Number MAC Address Low	1348
0x000BE49C	GLPCI_SERH	PCIe Serial Number MAC Address High	1348
0x000BE4A4	GLPCI_CAPCTRL	PCIe Capabilities Control	1348
0x000BE4A8	GLPCI_CAPSUP	PCIe Capabilities Support	1348
0x000BE4AC	GLPCI_LINKCAP	PCIe Link Capabilities	1349
0x000BE4B0	GLPCI_PMSUP	PCIe PM Support	1350
0x000BE4B4	GLPCI_REVID	PCIe Revision ID	1350
0x000BE4B8	GLPCI_VFSUP	PCIe VF Capabilities Support	1350
0x000BE4C0	GLPCI_CNF	PCIe Global Config	1350
0x000BE4F8	GLPCI_UPADD	PCIe Upper Address	1351
0x000BE4FC	GLPCI_PCIERR	PCIe Errors Reported	1351
0x000BE518	GLPCI_VENDORID	PCIe Vendor ID	1351
PF - MAC Registers			
0x0008C480	PRTMAC_PCS_XAUI_SWAP_A	PCS_XAUI_SWAP_A	1352
0x0008C484	PRTMAC_PCS_XAUI_SWAP_B	PCS_XAUI_SWAP_B	1353
0x001E2120	PRTGL_SAL	Port MAC Address Low	1354
0x001E2140	PRTGL_SAH	Port MAC Address High	1354
0x001E30C0	PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE	HSEC CONTROL Receive PFC ENABLE	1354
0x001E30D0	PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE	HSEC CONTROL Transmit PAUSE_ENABLE	1354
0x001E30E0	PRTMAC_HSEC_CTL_RX_ENABLE_GCP	HSEC CONTROL Receive ENABLE_GCP	1355
0x001E3110	PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1	HSEC CONTROL Receive PAUSE_DA_UCAST_PART1	1355
0x001E3120	PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2	HSEC CONTROL Receive PAUSE_DA_UCAST_PART2	1355
0x001E3140	PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART1	HSEC CONTROL Receive PAUSE_SA_PART1	1355
0x001E3150	PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART2	HSEC CONTROL Receive PAUSE_SA_PART2	1355
0x001E3260	PRTMAC_HSEC_CTL_RX_ENABLE_GPP	HSEC CONTROL Receive ENABLE_GPP	1356
0x001E32E0	PRTMAC_HSEC_CTL_RX_ENABLE_PPP	HSEC CONTROL Receive ENABLE_PPP	1356
0x001E3360	PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL	HSEC CONTROL Receive FORWARD_CONTROL	1356
0x001E3370 + 0x10*n, n=0...8	PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n]	HSEC CONTROL Transmit PAUSE_QUANTA	1356
0x001E3400 + 0x10*n, n=0...8	PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n]	HSEC CONTROL Transmit PAUSE_REFRESH_TIMER	1357
0x001E34B0	PRTMAC_HSEC_CTL_TX_SA_PART1	HSEC CONTROL Transmit SA_GPP_PART1	1357
0x001E34C0	PRTMAC_HSEC_CTL_TX_SA_PART2	HSEC CONTROL Transmit SA_GPP_PART2	1357
PF - Power Management Registers			
0x000B8140	PRTPM_GC	General Control	1358

Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x001E4320	PRTPM_EEE_STAT	Energy Efficient Ethernet (EEE) STATUS	1358
0x001E4360	PRTPM_EEER	Energy Efficient Ethernet (EEE) Register	1359
0x001E4380	PRTPM_EEEC	Energy Efficient Ethernet (EEE) Control	1359
0x001E43A0	PRTPM_RLPIC	EEE RX LPI Count	1360
0x001E43C0	PRTPM_TLPIC	EEE TX LPI Count	1360
0x001E43E0	PRTPM_EEETXC	EEE TX Control	1360
0x001E4400	PRTPM_EEEFWD	EEE TX Control	1360
PF - Wake-Up and Proxying Registers			
0x0006A000 + 0x80*n, n=0...7	PFPM_FHFT_LENGTH[n]	Flexible Host Filter Table Length	1361
0x0006B200	PFPM_WUC	Wake Up Control Register	1361
0x0006B400	PFPM_WUFC	Wake Up Filter Control Register	1361
0x0006B600	PFPM_WUS	Wake Up Status Register	1362
0x0006C000	PRTPM_FHFHR	Flexible Host Filter Header Removal	1362
0x0006C800	GLPM_WUMC	WU on MNG Control	1363
0x000B8080	PFPM_APM	APM Control Register	1363
0x001E4440 + 0x20*n, n=0...3	PRTPM_SAL[n]	MAC Address Low	1363
0x001E44C0 + 0x20*n, n=0...3	PRTPM_SAH[n]	MAC Address High	1363
PF - NVM Registers			
0x000B6008	GLNVM_ULD	Unit Load Status	1364
0x000B6010 + 0x4*n, n=0...59	GLNVM_PROTCSR[n]	Protected CSR List	1364
0x000B6100	GLNVM_GENS	Global NVM General Status Register	1365
0x000B6104	GLNVM_FLASHID	Flash ID Register	1365
0x000B6108	GLNVM_FLA	Flash Access Register	1365
0x000B6110	GLNVM_SRCTL	Shadow RAM Control Register	1367
0x000B6114	GLNVM_SRDATA	Shadow RAM Read/Write Data	1367
PF - Analyzer Registers			
0x001C0B20	PRT_L2TAGSEN	L2 Tag - Enable	1368
PF - Switch Registers			
0x0026CFB8 + 0x4*n, n=0...1	GL_SWR_DEF_ACT_EN[n]	Switching Table Default Action Enable Bitmap	1368
0x00270200 + 0x4*n, n=0...35	GL_SWR_DEF_ACT[n]	Switching Table Default Action	1368
PF - Interrupt Registers			
0x00020000 + 0x800*n + 0x4*INTVF, n=0...2, INTVF=0...511	VFINT_ITRN[n,INTVF]	VF Interrupt Throttling for Interrupt N	1369
0x00024800 + 0x4*INTVF, INTVF=0...511	VFINT_DYN_CTLN[INTVF]	VF Interrupt N Dynamic Control	1369
0x00025000 + 0x4*INTVF, INTVF=0...511	VPINT_LNKLSTN[INTVF]	Protected VF Interrupt N Linked List	1370
0x00025800 + 0x4*INTVF, INTVF=0...511	VPINT_RATEN[INTVF]	Protected VF Interrupt N Rate Limit	1370

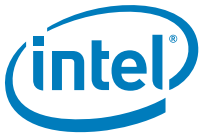


Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x00026800 + 0x4*INTVF, INTVF=0...511	VPINT_CEQCTL[INTVF]	VF PE Completion Event Queue Interrupt Cause Control	1371
0x00028000 + 0x400*n + 0x4*VF, n=0...2, VF=0...127	VFINT_ITR0[n,VF]	VF Interrupt Throttling for Interrupt Zero	1371
0x0002A000 + 0x4*VF, VF=0...127	VFINT_STAT_CTL0[VF]	VF Interrupt Zero Static Control	1372
0x0002A400 + 0x4*VF, VF=0...127	VFINT_DYN_CTL0[VF]	VF Interrupt Zero Dynamic Control	1372
0x0002A800 + 0x4*VF, VF=0...127	VPINT_LNKLST0[VF]	Protected VF Interrupt Zero Linked List	1373
0x0002AC00 + 0x4*VF, VF=0...127	VPINT_RATE0[VF]	Protected VF Interrupt Zero Rate Limit	1373
0x0002B800 + 0x4*VF, VF=0...127	VPINT_AEQCTL[VF]	VF PE Asynchronous Event Queue Interrupt Cause Control	1373
0x0002BC00 + 0x4*VF, VF=0...127	VFINT_ICR0[VF]	VF Interrupt Zero Cause	1374
0x0002C000 + 0x4*VF, VF=0...127	VFINT_ICR0_ENA[VF]	VF Interrupt Zero Cause Enablement	1374
0x00030000 + 0x800*n + 0x4*INTPF, n=0...2, INTPF=0...511	PFINT_ITRN[n,INTPF]	PF Interrupt Throttling for Interrupt N	1374
0x00034800 + 0x4*INTPF, INTPF=0...511	PFINT_DYN_CTLN[INTPF]	PF Interrupt N Dynamic Control	1375
0x00035000 + 0x4*INTPF, INTPF=0...511	PFINT_LNKLSTN[INTPF]	PF Interrupt N Linked List	1376
0x00035800 + 0x4*INTPF, INTPF=0...511	PFINT_RATEN[INTPF]	PF Interrupt N Rate Limit	1376
0x00036800 + 0x4*INTPF, INTPF=0...511	PFINT_CEQCTL[INTPF]	PF PE Completion Event Queue Interrupt Cause Control	1376
0x00038000 + 0x80*n, n=0...2	PFINT_ITR0[n]	PF Interrupt Throttling for Interrupt Zero	1377
0x00038400	PFINT_STAT_CTL0	PF Interrupt Zero Static Control	1377
0x00038480	PFINT_DYN_CTL0	PF Interrupt Zero Dynamic Control	1378
0x00038500	PFINT_LNKLST0	PF Interrupt Zero Linked List	1379
0x00038580	PFINT_RATE0	PF Interrupt Zero Rate Limit	1379
0x00038700	PFINT_AEQCTL	PF PE Asynchronous Event Queue Interrupt Cause Control	1379
0x00038780	PFINT_ICR0	PF Interrupt Zero Cause	1380
0x00038800	PFINT_ICR0_ENA	PF Interrupt Zero Cause Enablement	1380
0x0003A000 + 0x4*Q, Q=0...1535	QINT_RQCTL[Q]	Receive Queue Interrupt Cause Control	1381
0x0003C000 + 0x4*Q, Q=0...1535	QINT_TQCTL[Q]	Transmit Queue Interrupt Cause Control	1382
0x0003F100	PFGEN_PORTMDIO_NUM	LAN Port MDIO Number	1382
0x00088080	PFINT_GPIO_ENA	PF General Purpose IO Interrupt Enablement	1383
0x00088188	EMPINT_GPIO_ENA	EMP General Purpose IO Interrupt Enablement	1384
PF - Virtualization PF Registers			
0x000E6000 + 0x4*VF, VF=0...127	VP_MDET_TX[VF]	Malicious Driver Detected on TX	1385
0x000E6400	PF_MDET_TX	Malicious Driver Detected on TX	1385
0x000E6480	GL_MDET_TX	Malicious Driver TX Event Details	1385
0x0012A000 + 0x4*VF, VF=0...127	VP_MDET_RX[VF]	Malicious Driver Detected on RX	1385
0x0012A400	PF_MDET_RX	Malicious Driver Detected on RX	1386

Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x0012A510	GL_MDET_RX	Malicious Driver RX Event Details	1386
0x001C0500	PF_VT_PFALLOC	PF Resources Allocation	1386
PF - DCB Registers			
0x00083000	PRTDCB_GENC	Port DCB General Control	1387
0x00083020	PRTDCB_GENS	Port DCB General Status	1387
0x00083044	GLDCB_GENC	Global DCB General Control	1387
0x00098060	PRTDCB_TETSC_TPB	DCB Transmit ETS Control for TPB	1388
0x000A0040 + 0x20*n, n=0...7	PRTDCB_TCMSTC[n]	DCB Transmit Frame Monitoring Status per TC	1388
0x000A0180	PRTDCB_TDPMC	DCB Transmit Data Pipe Monitor Control	1388
0x000A2040 + 0x20*n, n=0...7	PRTDCB_TCWSTC[n]	DCB Transmit Command Waiting Status per TC	1388
0x000A21A0	PRTDCB_TCPMC	DCB Transmit Command Pipe Monitor Control	1389
0x000AE060	PRTDCB_TETSC_TCB	DCB Transmit ETS Control for TCB	1389
0x00122180 + 0x20*n, n=0...7	PRTDCB_RETSTCC[n]	DCB Receive ETS per TC Control	1389
0x001223A0	PRTDCB_RPPMC	DCB Receive per Port Pipe Monitor Control	1390
0x001223E0	PRTDCB_RETSC	DCB Receive ETS Control	1390
0x00122400 + 0x20*n, n=0...7	PRTDCB_RUPTQ[n]	DCB Receive per UP PFC Timer Queue	1390
0x00122618	GLDCB_RUPTI	DCB Receive per UP PFC Timer Indication	1391
0x001C0980	PRTDCB_TC2PFC	DCB TC to PFC Mapping	1391
0x001C09A0	PRTDCB_RUP2TC	DCB Receive UP to TC Mapping for RCB	1391
0x001C0B00	PRTDCB_RUP	DCB Receive UP in PPRS	1392
0x001E2400	PRTDCB_MFLCN	MAC Flow Control Register	1392
0x001E4560	PRTDCB_TFCS	Transmit Flow Control Status	1393
0x001E4580 + 0x20*n, n=0...3	PRTDCB_FCTTVN[n]	Flow Control Transmit Timer Value n	1393
0x001E4600	PRTDCB_FCRTV	Flow Control Refresh Threshold Value	1394
0x001E4640	PRTDCB_FCCFG	Flow Control Configuration	1394
0x001E4660 + 0x20*n, n=0...7	PRTDCB_TPFCTS[n]	DCB Transmit PFC Timer Status	1394
PF - Receive Packet Buffer Registers			
0x000AC100 + 0x20*n, n=0...7	PRTRPB_DHW[n]	RPB Dedicated Pool High Watermark	1395
0x000AC220 + 0x20*n, n=0...7	PRTRPB_DLW[n]	RPB Dedicated Pool Low Watermark	1395
0x000AC320 + 0x20*n, n=0...7	PRTRPB_DPS[n]	RPB Dedicated Pool Size	1395
0x000AC480 + 0x20*n, n=0...7	PRTRPB_SHT[n]	RPB Shared Pool High Threshold	1395
0x000AC580	PRTRPB_SHW	RPB Shared Pool High Watermark	1396
0x000AC5A0 + 0x20*n, n=0...7	PRTRPB_SLT[n]	RPB Shared Pool Low Threshold	1396
0x000AC6A0	PRTRPB_SLW	RPB Shared Pool Low Watermark	1396
0x000AC7C0	PRTRPB_SPS	RPB Shared Pool Size	1396
0x000AC828	GLRPB_DPSS	RPB Dedicated Pool Size for Single Shared Buffer State	1396

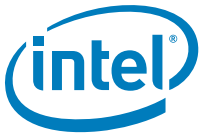


Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x000AC830	GLRPB_GHW	RPB Global High Watermark	1397
0x000AC834	GLRPB_GLW	RPB Global Low Watermark	1397
0x000AC844	GLRPB_PHW	RPB Packet High Watermark	1397
0x000AC848	GLRPB_PLW	RPB Packet Low Watermark	1397
PF - Transmit Scheduler Registers			
0x000B2080	GLSCD_QUANTA	Transmit Scheduler Quanta Register	1398
PF - Host Memory Cache Registers			
0x000C0000	PFHMC_SDCMD	Private Memory Space Segment Descriptor Command	1398
0x000C0100	PFHMC_SDDATALOW	Private Memory Space Segment Descriptor Data Low	1399
0x000C0200	PFHMC_SDDATAHIGH	Private Memory Space Segment Descriptor Data High	1399
0x000C0300	PFHMC_PDINV	Private Memory Space Page Descriptor Invalidate	1399
0x000C0400	PFHMC_ERRORINFO	Host Memory Cache Error Information Register	1400
0x000C0500	PFHMC_ERRORDATA	Host Memory Cache Error Data Register	1401
0x000C0800 + 0x4*n, n=0...15	GLHMC_SDPART[n]	Private Memory Segment Table Partitioning Registers	1401
0x000C0C00 + 0x4*n, n=0...15	GLHMC_PFASSIGN[n]	Private Memory Physical Function Table	1402
0x000C2004	GLHMC_LANTXOBJSZ	Private Memory LAN TX Object Size	1402
0x000C2008	GLHMC_LANQMAX	Private Memory LAN Queue Maximum	1402
0x000C200C	GLHMC_LANRXOBJSZ	Private Memory LAN RX Object Size	1403
0x000C2010	GLHMC_FCOEDDPOBJSZ	Private Memory FCoE DDP Object Size	1403
0x000C2014	GLHMC_FCOEMAX	Private Memory FCoE Resource Maximum	1403
0x000C2018	GLHMC_FCOEFOBJSZ	Private Memory FCoE Filter Object Size	1403
0x000C205C	GLHMC_FSIMCOBJSZ	Private Memory FSI Multicast Group Object Size	1404
0x000C2060	GLHMC_FSIMCMAX	Private Memory FSI Multicast Group Max	1404
0x000C2064	GLHMC_FSIAVOBJSZ	Private Memory FSI Address Vector Object Size	1404
0x000C2068	GLHMC_FSIAVMAX	Private Memory FSI Address Vector Max	1404
0x000C20D0	GLHMC_FCOEFMAX	Private Memory FCoE Filter Resource Maximum	1405
0x000C5600 + 0x4*n, n=0...15	GLHMC_FSIAVBASE[n]	FPM FSI Address Vector Base	1405
0x000C5700 + 0x4*n, n=0...15	GLHMC_FSIAVCNT[n]	FPM FSI Address Vector Object Count	1405
0x000C6000 + 0x4*n, n=0...15	GLHMC_FSIMCBASE[n]	FPM FSI Multicast Group Base	1406
0x000C6100 + 0x4*n, n=0...15	GLHMC_FSIMCCNT[n]	FPM FSI Multicast Group Object Count	1406
0x000C6200 + 0x4*n, n=0...15	GLHMC_LANTXBASE[n]	FPM LAN TX Queue Base	1406
0x000C6300 + 0x4*n, n=0...15	GLHMC_LANTXCNT[n]	FPM LAN TX Queue Object Count	1406

Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x000C6400 + 0x4*n, n=0...15	GLHMC_LANRXBASE[n]	FPM LAN RX Queue Base	1407
0x000C6500 + 0x4*n, n=0...15	GLHMC_LANRXCNT[n]	FPM LAN RX Queue Object Count	1407
0x000C6600 + 0x4*n, n=0...15	GLHMC_FCOEDDPBASE[n]	FPM FCoE DDP Base	1407
0x000C6700 + 0x4*n, n=0...15	GLHMC_FCOEDDPCNT[n]	FPM FCoE DDP Object Count	1407
0x000C6800 + 0x4*n, n=0...15	GLHMC_FCOEFBASE[n]	FPM FCoE Filters Base	1408
0x000C6900 + 0x4*n, n=0...15	GLHMC_FCOEFCNT[n]	FPM FCoE Filters Object Count	1408
PF - Context Manager Registers			
0x0010C000	PFCM_LAN_ERRINFO	CMLAN Error Info	1409
0x0010C080	PFCM_LAN_ERRDATA	CMLAN Error Data	1409
0x0010C100 + 0x80*n, n=0...3	PFCM_LANCTXDATA[n]	CMLAN Context Data Registers	1409
0x0010C300	PFCM_LANCTXCTL	CMLAN Context Control Register	1410
0x0010C380	PFCM_LANCTXSTAT	CMLAN Context Status Register	1410
0x00138400 + 0x4*VF, VF=0...127	VFCM_PE_ERRINFO[VF]	CMPE VF Error Info	1411
0x00138800 + 0x4*VF, VF=0...127	VFCM_PE_ERRDATA[VF]	CMPE VF Error Data	1411
PF - Admin Queue			
0x00080000	PF_ATQBAL	PF Admin Transmit Queue Base Address Low	1412
0x00080040	GL_ATQBAL	Global Admin Transmit Queue Base Address Low	1412
0x00080080	PF_ARQBAL	PF Admin Receive Queue Base Address Low	1412
0x000800C0	GL_ARQBAL	Global Admin Receive Queue Base Address Low	1412
0x00080100	PF_ATQBAH	PF Admin Transmit Queue Base Address High	1412
0x00080140	GL_ATQBAH	Global Admin Transmit Queue Base Address High	1412
0x00080180	PF_ARQBAH	PF Admin Receive Queue Base Address High	1413
0x000801C0	GL_ARQBAH	Global Admin Receive Queue Base Address High	1413
0x00080200	PF_ATQLEN	PF Admin Transmit Queue Length	1413
0x00080240	GL_ATQLEN	Global Admin Transmit Queue Length	1413
0x00080280	PF_ARQLEN	PF Admin Receive Queue Length	1414
0x00080300	PF_ATQH	PF Admin Transmit Head	1414
0x00080340	GL_ATQH	Global Admin Transmit Head	1414
0x00080380	PF_ARQH	PF Admin Receive Queue Head	1414
0x000803C0	GL_ARQH	Global Admin Receive Queue Head	1415
0x00080400	PF_ATQT	PF Admin Transmit Tail	1415
0x00080440	GL_ATQT	Global Admin Transmit Tail	1415
0x00080480	PF_ARQT	PF Admin Receive Queue Tail	1415
0x000804C0	GL_ARQT	Global Admin Receive Queue Tail	1415
0x00080800 + 0x4*VF, VF=0...127	VF_ATQBAL[VF]	VF Admin Transmit Queue Base Address Low	1416

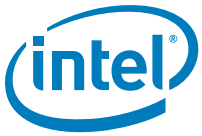


Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x00080C00 + 0x4*VF, VF=0...127	VF_ARQBAL[VF]	VF Admin Receive Queue Base Address Low	1416
0x00081000 + 0x4*VF, VF=0...127	VF_ATQBAH[VF]	VF Admin Transmit Queue Base Address High	1416
0x00081400 + 0x4*VF, VF=0...127	VF_ARQBAH[VF]	VF Admin Receive Queue Base Address High	1416
0x00081800 + 0x4*VF, VF=0...127	VF_ATQLEN[VF]	VF Admin Transmit Queue Length	1416
0x00081C00 + 0x4*VF, VF=0...127	VF_ARQLEN[VF]	VF Admin Receive Queue Length	1417
0x00082000 + 0x4*VF, VF=0...127	VF_ATQH[VF]	VF Admin Transmit Head	1417
0x00082400 + 0x4*VF, VF=0...127	VF_ARQH[VF]	VF Admin Receive Queue Head	1417
0x00082800 + 0x4*VF, VF=0...127	VF_ATQT[VF]	VF Admin Transmit Tail	1417
0x00082C00 + 0x4*VF, VF=0...127	VF_ARQT[VF]	VF Admin Receive Queue Tail	1418
PF - Statistics Registers			
0x00300000 + 0x8*n, n=0...3	GLPRT_GORCL[n]	Port Good Octets Received Count Low	1418
0x00300004 + 0x8*n, n=0...3	GLPRT_GORCH[n]	Port Good Octets Received Count High	1418
0x00300020 + 0x8*n, n=0...3	GLPRT_MLFC[n]	Port MAC Local Fault Count	1418
0x00300040 + 0x8*n, n=0...3	GLPRT_MRFC[n]	Port MAC Remote Fault Count	1418
0x00300080 + 0x8*n, n=0...3	GLPRT_CRCERRS[n]	Port CRC Error Count	1419
0x003000A0 + 0x8*n, n=0...3	GLPRT_RLEC[n]	Receive Length Error Count	1419
0x003000E0 + 0x8*n, n=0...3	GLPRT_ILLERRC[n]	Port Illegal Byte Error Count	1419
0x00300100 + 0x8*n, n=0...3	GLPRT_RUC[n]	Receive Undersize Count	1419
0x00300120 + 0x8*n, n=0...3	GLPRT_ROC[n]	Receive Oversize Count	1419
0x00300140 + 0x8*n, n=0...3	GLPRT_LXONRXC[n]	Port Link XON Received Count	1420
0x00300160 + 0x8*n, n=0...3	GLPRT_LXOFFRXC[n]	Port Link XOFF Received Count	1420
0x00300180 + 0x8*n + 0x20*m, n=0...3, m=0...7	GLPRT_PXONRXC[n,m]	Priority XON Received Count	1420
0x00300280 + 0x8*n + 0x20*m, n=0...3, m=0...7	GLPRT_PXOFFRXC[n,m]	Priority XOFF Received Count	1420
0x00300380 + 0x8*n + 0x20*m, n=0...3, m=0...7	GLPRT_RXON2OFFCNT[n,m]	Priority XON to XOFF Count	1420
0x00300480 + 0x8*n, n=0...3	GLPRT_PRC64L[n]	Packets Received [64 Bytes] Count Low	1421
0x00300484 + 0x8*n, n=0...3	GLPRT_PRC64H[n]	Packets Received [64 Bytes] Count High	1421
0x003004A0 + 0x8*n, n=0...3	GLPRT_PRC127L[n]	Packets Received [65-127 Bytes] Count Low	1421
0x003004A4 + 0x8*n, n=0...3	GLPRT_PRC127H[n]	Packets Received [65-127 Bytes] Count High	1421
0x003004C0 + 0x8*n, n=0...3	GLPRT_PRC255L[n]	Packets Received [128-255 Bytes] Count Low	1422
0x003004C4 + 0x8*n, n=0...3	GLPRT_PRC255H[n]	Packets Received [128-255 Bytes] Count High	1422
0x003004E0 + 0x8*n, n=0...3	GLPRT_PRC511L[n]	Packets Received [256-511 Bytes] Count Low	1422
0x003004E4 + 0x8*n, n=0...3	GLPRT_PRC511H[n]	Packets Received [256-511 Bytes] Count High	1422

Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x00300500 + 0x8*n, n=0...3	GLPRT_PRC1023L[n]	Packets Received [512-1023 Bytes] Count Low	1423
0x00300504 + 0x8*n, n=0...3	GLPRT_PRC1023H[n]	Packets Received [512-1023 Bytes] Count High	1423
0x00300520 + 0x8*n, n=0...3	GLPRT_PRC1522L[n]	Packets Received [1024-1522 Bytes] Count Low	1423
0x00300524 + 0x8*n, n=0...3	GLPRT_PRC1522H[n]	Packets Received [1024-1522 Bytes] Count High	1423
0x00300540 + 0x8*n, n=0...3	GLPRT_PRC9522L[n]	Packets Received [1523-9522 Bytes] Count Low	1424
0x00300544 + 0x8*n, n=0...3	GLPRT_PRC9522H[n]	Packets Received [1523-9522 Bytes] Count High	1424
0x00300560 + 0x8*n, n=0...3	GLPRT_RFC[n]	Receive Fragment Count	1424
0x00300580 + 0x8*n, n=0...3	GLPRT_RJC[n]	Receive Jabber Count	1424
0x003005A0 + 0x8*n, n=0...3	GLPRT_UPRCL[n]	Port Unicast Packets Received Count Low	1425
0x003005A4 + 0x8*n, n=0...3	GLPRT_UPRCH[n]	Port Unicast Packets Received Count High	1425
0x003005C0 + 0x8*n, n=0...3	GLPRT_MPRCL[n]	Port Multicast Packets Received Count Low	1425
0x003005C4 + 0x8*n, n=0...3	GLPRT_MPRCH[n]	Port Multicast Packets Received Count High	1425
0x003005E0 + 0x8*n, n=0...3	GLPRT_BPRCL[n]	Port Broadcast Packets Received Count Low	1425
0x003005E4 + 0x8*n, n=0...3	GLPRT_BPRCH[n]	Port Broadcast Packets Received Count High	1426
0x00300600 + 0x8*n, n=0...3	GLPRT_RDPC[n]	Port Receive Packets Discarded Count	1426
0x00300620 + 0x8*n, n=0...3	GLPRT_LDPC[n]	Loopback Packets Discarded Count	1426
0x00300660 + 0x8*n, n=0...3	GLPRT_RUPP[n]	Port Received with No Destination	1426
0x00300680 + 0x8*n, n=0...3	GLPRT_GOTCL[n]	Port Good Octets Transmit Count Low	1426
0x00300684 + 0x8*n, n=0...3	GLPRT_GOTCH[n]	Port Good Octets Transmit Count High	1426
0x003006A0 + 0x8*n, n=0...3	GLPRT_PTC64L[n]	Packets Transmitted [64 Bytes] Count Low	1427
0x003006A4 + 0x8*n, n=0...3	GLPRT_PTC64H[n]	Packets Transmitted [64 Bytes] Count High	1427
0x003006C0 + 0x8*n, n=0...3	GLPRT_PTC127L[n]	Packets Transmitted [65-127 Bytes] Count Low	1427
0x003006C4 + 0x8*n, n=0...3	GLPRT_PTC127H[n]	Packets Transmitted [65-127 Bytes] Count High	1427
0x003006E0 + 0x8*n, n=0...3	GLPRT_PTC255L[n]	Packets Transmitted [128-255 Bytes] Count Low	1428
0x003006E4 + 0x8*n, n=0...3	GLPRT_PTC255H[n]	Packets Transmitted [128-255 Bytes] Count High	1428
0x00300700 + 0x8*n, n=0...3	GLPRT_PTC511L[n]	Packets Transmitted [256-511 Bytes] Count Low	1428
0x00300704 + 0x8*n, n=0...3	GLPRT_PTC511H[n]	Packets Transmitted [256-511 Bytes] Count High	1428
0x00300720 + 0x8*n, n=0...3	GLPRT_PTC1023L[n]	Packets Transmitted [512-1023 Bytes] Count Low	1429

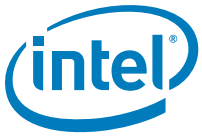


Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x00300724 + 0x8*n, n=0...3	GLPRT_PTC1023H[n]	Packets Transmitted [512-1023 Bytes] Count High	1429
0x00300740 + 0x8*n, n=0...3	GLPRT_PTC1522L[n]	Packets Transmitted [1024-1522 Bytes] Count Low	1429
0x00300744 + 0x8*n, n=0...3	GLPRT_PTC1522H[n]	Packets Transmitted [1024-1522 Bytes] Count High	1429
0x00300760 + 0x8*n, n=0...3	GLPRT_PTC9522L[n]	Packets Transmitted [1523-9522 bytes] Count Low	1430
0x00300764 + 0x8*n, n=0...3	GLPRT_PTC9522H[n]	Packets Transmitted [1523-9522 bytes] Count High	1430
0x00300780 + 0x8*n + 0x20*m, n=0...3, m=0...7	GLPRT_PXONTXC[n,m]	Priority XON Transmitted Count	1430
0x00300880 + 0x8*n + 0x20*m, n=0...3, m=0...7	GLPRT_PXOFFTXC[n,m]	Priority XOFF Transmitted Count	1430
0x00300980 + 0x8*n, n=0...3	GLPRT_LXONTXC[n]	Port Link XON Transmitted Count	1431
0x003009A0 + 0x8*n, n=0...3	GLPRT_LXOFFTXC[n]	Port Link XOFF Transmitted Count	1431
0x003009C0 + 0x8*n, n=0...3	GLPRT_UPTCL[n]	Port Unicast Packets Transmit Count Low	1431
0x003009C4 + 0x8*n, n=0...3	GLPRT_UPTCH[n]	Port Unicast Packets Transmit Count High	1431
0x003009E0 + 0x8*n, n=0...3	GLPRT_MPTCL[n]	Port Multicast Packets Transmit Count Low	1431
0x003009E4 + 0x8*n, n=0...3	GLPRT_MPTCH[n]	Port Multicast Packets Transmit Count High	1432
0x00300A00 + 0x8*n, n=0...3	GLPRT_BPTCL[n]	Port Broadcast Packets Transmit Count Low	1432
0x00300A04 + 0x8*n, n=0...3	GLPRT_BPTCH[n]	Port Broadcast Packets Transmit Count High	1432
0x00300A20 + 0x8*n, n=0...3	GLPRT_TDOLD[n]	Transmit Discard on Link Down	1432
0x00310000 + 0x8*n, n=0...383	GLV_RDPC[n]	VSI Received Discard Packet Count	1432
0x00314000 + 0x8*n, n=0...143	GL_FCOELAST[n]	FCoE Last Error Count	1433
0x00314480 + 0x8*n, n=0...143	GL_FCOEDDPC[n]	FCoE DDP Count	1433
0x00314D80 + 0x8*n, n=0...143	GL_FCOECRC[n]	FCoE CRC Error Count	1433
0x00315200 + 0x8*n, n=0...143	GL_FCOEPRC[n]	FCoE Packets Received Count	1433
0x00318000 + 0x8*n, n=0...143	GL_RXERR1_L[n]	Receive Error Counter 1 Low	1433
0x00318480 + 0x8*n, n=0...143	GL_FCOEDIFEC[n]	FCoE DIF Receive Error Count	1434
0x0031C000 + 0x8*n, n=0...143	GL_RXERR2_L[n]	FCoE Valid DIX Tag to Host Count	1434
0x00320000 + 0x8*n, n=0...143	GL_FCOEDWRCL[n]	FCoE DWord Received Count Low	1434
0x00320004 + 0x8*n, n=0...143	GL_FCOEDWRCH[n]	FCOE DWord Received Count High	1434
0x00324000 + 0x8*n, n=0...143	GL_FCOERPDC[n]	FCoE Rx Packets Dropped Count	1435
0x00328000 + 0x8*n, n=0...383	GLV_GOTCL[n]	VSI Good Octets Transmit Count Low	1435
0x00328004 + 0x8*n, n=0...383	GLV_GOTCH[n]	VSI Good Octets Transmit Count High	1435
0x0032C000 + 0x8*n, n=0...15	GLSW_GOTCL[n]	Switch Good Octets Transmit Count Low	1435
0x0032C004 + 0x8*n, n=0...15	GLSW_GOTCH[n]	Switch Good Octets Transmit Count High	1435
0x00330000 + 0x8*n, n=0...127	GLVEBVL_GOTCL[n]	VEB VLAN Transmit Byte Count Low	1436

Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x00330004 + 0x8*n, n=0...127	GLVEBVL_GOTCH[n]	VEB VLAN Transmit Byte Count High	1436
0x00334000 + 0x8*n + 0x40*m, n=0...7, m=0...15	GLVEBTC_TBCL[n,m]	VEB TC Transmit Byte Count Low	1436
0x00334004 + 0x8*n + 0x40*m, n=0...7, m=0...15	GLVEBTC_TBCH[n,m]	VEB TC Transmit Byte Count High	1436
0x00338000 + 0x8*n + 0x40*m, n=0...7, m=0...15	GLVEBTC_TPCL[n,m]	VEB TC Transmit Packet Count Low	1436
0x00338004 + 0x8*n + 0x40*m, n=0...7, m=0...15	GLVEBTC_TPCH[n,m]	VEB TC Transmit Packet Count High	1437
0x0033C000 + 0x8*n, n=0...383	GLV_UPTCL[n]	VSI Unicast Packets Transmit Count Low	1437
0x0033C004 + 0x8*n, n=0...383	GLV_UPTCH[n]	VSI Unicast Packets Transmit Count High	1437
0x0033CC00 + 0x8*n, n=0...383	GLV_MPTCL[n]	VSI Multicast Packets Transmit Count Low	1437
0x0033CC04 + 0x8*n, n=0...383	GLV_MPTCH[n]	VSI Multicast Packets Transmit Count High	1437
0x0033D800 + 0x8*n, n=0...383	GLV_BPTCL[n]	VSI Broadcast Packets Transmit Count Low	1438
0x0033D804 + 0x8*n, n=0...383	GLV_BPTCH[n]	VSI Broadcast Packets Transmit Count High	1438
0x00340000 + 0x8*n, n=0...15	GLSW_UPTCL[n]	Switch Unicast Packets Transmit Count Low	1438
0x00340004 + 0x8*n, n=0...15	GLSW_UPTCH[n]	Switch Unicast Packets Transmit Count High	1438
0x00340080 + 0x8*n, n=0...15	GLSW_MPTCL[n]	Switch Multicast Packets Transmit Count Low	1438
0x00340084 + 0x8*n, n=0...15	GLSW_MPTCH[n]	Switch Multicast Packets Transmit Count High	1439
0x00340100 + 0x8*n, n=0...15	GLSW_BPTCL[n]	Switch Broadcast Packets Transmit Count Low	1439
0x00340104 + 0x8*n, n=0...15	GLSW_BPTCH[n]	Switch Broadcast Packets Transmit Count High	1439
0x00344000 + 0x4*VSI, VSI=0...383	GLV_TEPC[VSI]	VSI Transmit Error Packet Count	1439
0x00344C00 + 0x8*n, n=0...143	GL_FCOEPTC[n]	FCoE Packets Transmitted Count	1439
0x00348000 + 0x8*n, n=0...15	GLSW_TDPC[n]	Switch Transmit Packets Discarded Count	1439
0x00348080 + 0x8*n, n=0...143	GL_FCOEDWTCL[n]	FCoE DWord Transmitted Count Low	1440
0x00348084 + 0x8*n, n=0...143	GL_FCOEDWTCH[n]	FCoE DWord Transmitted Count High	1440
0x0034C000 + 0x8*n, n=0...143	GL_FCOEDIXEC[n]	FCoE Bad DIX Error from Host Count	1440
0x00350000 + 0x8*n, n=0...143	GL_FCOEDIXVC[n]	FCoE Valid DIX Tag from Host Count	1440
0x00354000 + 0x8*n, n=0...143	GL_FCOEDIFTCL[n]	FCoE DIF Block Transmit Count Low	1440
0x00358000 + 0x8*n, n=0...383	GLV_GORCL[n]	VSI Good Octets Received Count Low	1441
0x00358004 + 0x8*n, n=0...383	GLV_GORCH[n]	VSI Good Octets Received Count High	1441
0x0035C000 + 0x8*n, n=0...15	GLSW_GORCL[n]	Switch Good Octets Received Count Low	1441
0x0035C004 + 0x8*n, n=0...15	GLSW_GORCH[n]	Switch Good Octets Received Count High	1441
0x00360000 + 0x8*n, n=0...127	GLVEBVL_GORCL[n]	VEB VLAN Receive Byte Count Low	1441

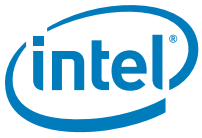


Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x00360004 + 0x8*n, n=0...127	GLVEBVL_GORCH[n]	VEB VLAN Receive Byte Count High	1442
0x00364000 + 0x8*n + 0x40*m, n=0...7, m=0...15	GLVEBTC_RBCL[n,m]	VEB TC Receive Byte Count Low	1442
0x00364004 + 0x8*n + 0x40*m, n=0...7, m=0...15	GLVEBTC_RBCH[n,m]	VEB TC Receive Byte Count High	1442
0x00368000 + 0x8*n + 0x40*m, n=0...7, m=0...15	GLVEBTC_RPCL[n,m]	VEB TC Receive Packet Count Low	1442
0x00368004 + 0x8*n + 0x40*m, n=0...7, m=0...15	GLVEBTC_RPCH[n,m]	VEB TC Receive Packet Count High	1442
0x0036C000 + 0x8*n, n=0...383	GLV_UPRCL[n]	VSI Unicast Packets Received Count Low	1443
0x0036C004 + 0x8*n, n=0...383	GLV_UPRCH[n]	VSI Unicast Packets Received Count High	1443
0x0036CC00 + 0x8*n, n=0...383	GLV_MPRCL[n]	VSI Multicast Packets Received Count Low	1443
0x0036CC04 + 0x8*n, n=0...383	GLV_MPRCH[n]	VSI Multicast Packets Received Count High	1443
0x0036D800 + 0x8*n, n=0...383	GLV_BPRCL[n]	VSI Broadcast Packets Received Count Low	1443
0x0036D804 + 0x8*n, n=0...383	GLV_BPRCH[n]	VSI Broadcast Packets Received Count High	1444
0x0036E400 + 0x8*n, n=0...383	GLV_RUPP[n]	VSI Received Unknown Packet Protocol Count	1444
0x00370000 + 0x8*n, n=0...15	GLSW_UPRCL[n]	Switch Unicast Packets Received Count Low	1444
0x00370004 + 0x8*n, n=0...15	GLSW_UPRCH[n]	Switch Unicast Packets Received Count High	1444
0x00370080 + 0x8*n, n=0...15	GLSW_MPRCL[n]	Switch Multicast Packets Received Count Low	1444
0x00370084 + 0x8*n, n=0...15	GLSW_MPRCH[n]	Switch Multicast Packets Received Count High	1445
0x00370100 + 0x8*n, n=0...15	GLSW_BPRCL[n]	Switch Broadcast Packets Received Count Low	1445
0x00370104 + 0x8*n, n=0...15	GLSW_BPRCH[n]	Switch Broadcast Packets Received Count High	1445
0x00370180 + 0x8*n, n=0...15	GLSW_RUPP[n]	Switch Received Unknown Packet Protocol Count	1445
0x00374000 + 0x8*n, n=0...127	GLVEBVL_UPCL[n]	VEB VLAN Unicast Packet Count Low	1445
0x00374004 + 0x8*n, n=0...127	GLVEBVL_UPCH[n]	VEB VLAN Unicast Packet Count High	1446
0x00374400 + 0x8*n, n=0...127	GLVEBVL_MPCL[n]	VEB VLAN Multicast Packet Count Low	1446
0x00374404 + 0x8*n, n=0...127	GLVEBVL_MPCH[n]	VEB VLAN Multicast Packet Count High	1446
0x00374800 + 0x8*n, n=0...127	GLVEBVL_BPCL[n]	VEB VLAN Broadcast Packet Count Low	1446
0x00374804 + 0x8*n, n=0...127	GLVEBVL_BPCH[n]	VEB VLAN Broadcast Packet Count High	1446
0x00375400 + 0x8*n, n=0...3	GLPRT_TDPC[n]	Port Transmit Packets Discarded Count	1447
PF - LAN Transmit Receive Registers			
0x000442D8	GLLAN_TSOMSK_F	Global TSO TCP Mask First	1447
0x000442DC	GLLAN_TSOMSK_M	Global TSO TCP Mask Middle	1447

Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x000442E0	GLLAN_TSOMSK_L	Global TSO TCP Mask Last	1447
0x00051060	GL_RDPU_CNTRL	Receive Processing Block Control	1448
0x00070000 + 0x400*n + 0x4*VF, n=0...15, VF=0...127	VPLAN_QTABLE[n,VF]	VF PF Queue Mapping Table	1448
0x00074000 + 0x4*VF, VF=0...127	VPLAN_MAPENA[VF]	VF LAN Enablement	1448
0x000E4000 + 0x4*Q, Q=0...1535	QTX_HEAD[Q]	Global Transmit Queue Head	1448
0x000E6500 + 0x4*n, n=0...11	GLLAN_TXPRE_QDIS[n]	Global Transmit Pre Queue Disable	1449
0x00100000 + 0x4*Q, Q=0...1535	QTX_ENA[Q]	Global Transmit Queue Enable	1449
0x00104000 + 0x4*Q, Q=0...1535	QTX_CTL[Q]	Global Transmit Queue Control	1449
0x00108000 + 0x4*Q, Q=0...1535	QTX_TAIL[Q]	Global Transmit Queue Tail	1450
0x00120000 + 0x4*Q, Q=0...1535	QRX_ENA[Q]	Global Receive Queue Enable	1450
0x00128000 + 0x4*Q, Q=0...1535	QRX_TAIL[Q]	Global Receive Queue Tail	1450
0x0012A500	GLLAN_RCTL_0	Global RLAN Control 0	1451
0x001C0400	PFLAN_QALLOC	PF Queue Allocation	1451
0x00200000 + 0x800*n + 0x4*VSI, n=0...7, VSI=0...383	VSILAN_QTABLE[n,VSI]	VSI Receive Queue Mapping Table	1451
0x0020C800 + 0x4*VSI, VSI=0...383	VSILAN_QBASE[VSI]	VSI Queue Control	1452
PF - Rx Filters Registers			
0x001C0000 + 0x4*VF, VF=0...127	VPQF_CTL[VF]	VF Queue Filter Control	1452
0x001C0AC0	PFQF_CTL_0	PF Queue Filter Control 0	1452
0x00206000 + 0x800*n + 0x4*VSI, n=0...3, VSI=0...383	VSIQF_TCREGION[n,VSI]	VSI Receive Traffic Class Queues	1453
0x0020D800 + 0x4*VSI, VSI=0...383	VSIQF_CTL[VSI]	VSI Queue Filter Control	1454
0x00220000 + 0x400*n + 0x4*VF, n=0...15, VF=0...127	VFQF_HLUT[n,VF]	VF Queue Filter Hash LUT	1454
0x00228000 + 0x400*n + 0x4*VF, n=0...12, VF=0...127	VFQF_HKEY[n,VF]	VF Queue Filter Hash Key	1454
0x0022E000 + 0x400*n + 0x4*VF, n=0...7, VF=0...127	VFQF_HREGION[n,VF]	VF Queue Filter Hash Region of Queues	1454
0x00230800 + 0x400*n + 0x4*VF, n=0...1, VF=0...127	VFQF_HENA[n,VF]	VF Queue Filter Hash Enabled Packet Type	1455
0x00240000 + 0x80*n, n=0...127	PFQF_HLUT[n]	PF Queue Filter Hash LUT	1455
0x00244800 + 0x80*n, n=0...12	PFQF_HKEY[n]	PF Queue Filter Hash Key	1456
0x00245900 + 0x80*n, n=0...1	PFQF_HENA[n]	PF Queue Filter Hash Enabled Packet Type	1456
0x00245D80	PFQF_CTL_1	PF Queue Filter Control 1	1456
0x00246280	PFQF_FDALLOC	PF Queue Filter Flow Director Allocation	1456
0x00246380	PFQF_FDSTAT	PF Queue Filter Flow Director Allocation Status	1456
0x00253800 + 0x20*n, n=0...63	PRTQF_FD_FLXINSET[n]	Port Queue Filter Flow Director Input Set	1457
0x00255200 + 0x20*n, n=0...8	PRTQF_FLX_PIT[n]	Port Queue Filter - Flexible Parser Information Table	1457
0x00256E60	PRTQF_CTL_0	Port Queue Filter Control 0	1457

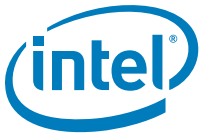
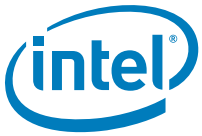


Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x00266800 + 0x4*n, n=0...511	GLQF_PCNT[n]	Global Queue Filter Packet Counter	1458
0x00267E00 + 0x4*n + 0x8*m, n=0...1, m=0...63	GLQF_SWAP[n,m]	Global Queue Filter SWAP Fields	1458
0x00269BA4	GLQF_CTL	Global Queue Filter Control	1458
0x00269BAC	GLQF_FDCNT_0	Global Queue Filter Flow Director Status 0	1459
0x00269D00 + 0x4*n, n=0...63	GLQF_HSYM[n]	Global Queue Filter Symmetric Hash Enablement	1459
0x00270140 + 0x4*n, n=0...12	GLQF_HKEY[n]	Global Queue Filter Hash Key	1459
PF - TimeSync (IEEE 1588) Registers			
0x00085020	PRTTSYN_CTL1	Port Time Sync Control 1	1460
0x00085040 + 0x20*n, n=0...3	PRTTSYN_RXTIME_H[n]	Port Time Sync Receive PTP Packet Time High	1460
0x000850C0 + 0x20*n, n=0...3	PRTTSYN_RXTIME_L[n]	Port Time Sync Receive PTP Packet Time Low	1461
0x00085140	PRTTSYN_STAT_1	Port Time Sync Status 1	1461
0x001E4040	PRTTSYN_INC_L	Port Time Sync Increment Value Low	1461
0x001E4060	PRTTSYN_INC_H	Port Time Sync Increment Value High	1461
0x001E4080 + 0x20*n, n=0...1	PRTTSYN_EVNT_L[n]	Port Time Sync Event Time Low	1462
0x001E40C0 + 0x20*n, n=0...1	PRTTSYN_EVNT_H[n]	Port Time Sync Event Time High	1462
0x001E4100	PRTTSYN_TIME_L	Port Time Sync Time Low	1462
0x001E4120	PRTTSYN_TIME_H	Port Time Sync Time High	1462
0x001E4140 + 0x20*n, n=0...1	PRTTSYN_TGT_L[n]	Port Time Sync Target Time Low	1462
0x001E4180 + 0x20*n, n=0...1	PRTTSYN_TGT_H[n]	Port Time Sync Target Time High	1462
0x001E41C0	PRTTSYN_TXTIME_L	Port Time Sync Transmit Packet Time Low	1463
0x001E41E0	PRTTSYN_TXTIME_H	Port Time Sync Transmit Packet Time High	1463
0x001E4200	PRTTSYN_CTL0	Port Time Sync Control 0	1463
0x001E4220	PRTTSYN_STAT_0	Port Time Sync Status 0	1464
0x001E4240 + 0x20*n, n=0...1	PRTTSYN_CLKO[n]	Port Time Sync Clock Out Duration	1464
0x001E4280	PRTTSYN_ADJ	Port Time Sync Adjustment	1464
0x001E42A0 + 0x20*n, n=0...1	PRTTSYN_AUX_0[n]	Port Time Sync AUX Control 0	1464
0x001E42E0 + 0x20*n, n=0...1	PRTTSYN_AUX_1[n]	Port Time Sync AUX Control 1	1465
PF - FCoE Registers			
0x00269B94	GLFCOE_RCTL	FC Receive Control	1465
PF - Manageability Registers			
0x00083100	GL_FWRESETCNT	Firmware Reset Count	1466
0x00085160 + 0x20*n, n=0...7	PRT_MNG_FTFT_MASK[n]	Flexible TCO Filter Table Registers - Mask	1466
0x00085260	PRT_MNG_FTFT_LENGTH	Flexible TCO Filter Table Registers - Length	1466
0x000852A0 + 0x20*n, n=0...31	PRT_MNG_FTFT_DATA[n]	Flexible TCO Filter Table Registers - Data	1467

Table 11-6. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
0x000B6130	GL_MNG_HWARB_CTRL	Hardware Arbitration Control	1467
0x000B6134	GL_MNG_FWSM	Firmware Semaphore	1467
0x00254200 + 0x20*n, n=0...15	PRT_MNG_MIPAF6[n]	Manageability IPv6 Address Filter	1469
0x00254E00 + 0x20*n, n=0...15	PRT_MNG_MFUTP[n]	Management Flex UDP/TCP Ports	1469
0x00255900 + 0x20*n, n=0...7	PRT_MNG_MAVTV[n]	Management VLAN TAG Value	1469
0x00255D00 + 0x20*n, n=0...7	PRT_MNG_MDEF[n]	Manageability Decision Filters1	1470
0x00255F00 + 0x20*n, n=0...7	PRT_MNG_MDEF_EXT[n]	Manageability Decision Filters	1471
0x00256280 + 0x20*n, n=0...3	PRT_MNG_MIPAF4[n]	Manageability IPv4 Address Filter	1472
0x00256380 + 0x20*n, n=0...3	PRT_MNG_MMAH[n]	Manageability MAC Address High	1472
0x00256480 + 0x20*n, n=0...3	PRT_MNG_MMAL[n]	Manageability MAC Address Low	1472
0x00256580 + 0x20*n, n=0...3	PRT_MNG_MDEFVSI[n]	Management Decision Filters Buffers	1473
0x00256780 + 0x20*n, n=0...3	PRT_MNG_METF[n]	Management Ethernet Type Filters	1473
0x00256A20	PRT_MNG_MANC	Management Control Register	1473
0x00256A60	PRT_MNG_MNGONLY	Management Only Traffic Register	1474
0x00256AA0	PRT_MNG_MSFM	Manageability Special Filters Modifiers	1475



11.2.2 Detailed Register Description - PF BAR0

11.2.2.1 PF - General Registers

11.2.2.1.1 VF Reset Status - VFGEN_RSTAT[VF] (0x00074400 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
VFR_STATE	1:0	00b	RW	The VFR state defines the VFR reset progress as follows: 00b = VFR in progress. 01b = VFR completed. 10b = Reserved. 11b = Reserved. This field is used to communicate the reset progress to the VF with no impact on hardware functionality.
RSVD	31:2	0x0	RSV	Reserved.

11.2.2.1.2 Firmware Status Register - GL_FWSTS (0x00083048; RO)

This register reports the status of the EMP.

Field	Bit(s)	Init.	Type	Description
FWS0B	7:0	0x0	RO	Firmware Status 0 Byte. This bit is RO through the host interface.
RSVD	8	0b	RSV	Reserved.
FWRI	9	1b	RW1C	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	Reserved.
FWS1B	23:16	0x0	RO	Firmware Status 1 Byte. This bit is RO through the host interface.
RESERVED	31:24	0x0	RSV	Reserved.

11.2.2.1.3 PF State - PFGEN_STATE[PF] (0x00088000; RO)

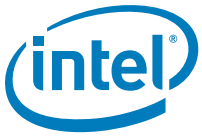
This register defines to software the main characteristics of the PF. It does not have any direct impact on device functionality.

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved
PFFCEN	1	0b	RW	PF FCoE Enable. 0b = FCoE should not be used for this function. 1b = FCoE can be used for this function.
PFLINKEN	2	0b	RW	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	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	Reserved.

11.2.2.1.4 Global GPIO Control - GLGEN_GPIO_CTL[n] (0x00088100 + 0x4*n, n=0...29; RW)

General GPIO Control Registers. These registers control the mode of operation of general purpose I/O (GPIOn) pins. There are 30 GPIO pins that can be configured to different modes and assigned to different ports. There is one register per GPIOn pin. Register GLGEN_GPIO_CTL[n] controls GPIOn pin. This register is initialized only at LAN Power Good preserving the GPIO states across software and PCIe resets.

Field	Bit(s)	Init.	Type	Description
PRT_NUM	1:0	00b	RW	Controls the port number associated with this GPIOn pin. This field is valid when the un-assigned port number bit (*PRT_NUM_NA*) is set to 0b. If a port number is not assigned then this GPIOn pin is a global resource not associated with any specific port.
RESERVED	2	0b	RSV	Reserved.
PRT_NUM_NA	3	0b	RW	Port Number Not Assigned. Set to 1b if this GPIOn pin is not assigned to any port. The *PRT_NUM* field is ignored when this bit is set 1b (used by software/ firmware only).
PIN_DIR	4	0b	RW	Pin Direction. Controls whether this GPIOn pin is configured as an input or output 0b = Input 1b = Output This bit is not affected by software or system reset, only by initial power on or direct software writes. ***Note:*** When GPIO functionality is set to LED, the pin direction is set implicitly to output regardless of this field's value.



Field	Bit(s)	Init.	Type	Description
TRI_CTL	5	0b	RW	Tristate Control. When this GPIOn pin is configured as an output, this field controls whether the pin is driven to tristate or driven high (0b = tristate, 1b = driven high) when *SDP_DATA* is set to 1b. The pin is driven to active low when the *SDP_DATA* is set to 0b. This bit is not affected by software or system reset, only by initial power on or direct software writes. The *SDP_DATA* is set through the GLGEN_GPIO_SET register. ***Note:*** This field is not used when GPIO is set to LED functionality.
OUT_CTL	6	0b	RW	Controls the output state during reset or D3 power state when no WoL or no manageability. 0b = The output transitions to tristate during reset or D3 power state. 1b = The output is driven high during reset or D3 power state. This bit is ignored when this GPIO pin is configured as an input. This bit is not affected by software or system reset, only by initial power on or direct software writes. ***Note:*** This field is not used when GPIO functionality is set to LED.
PIN_FUNC	9:7	000b	RW	Pin Functionality. This field controls the operation mode (or functionality) for this GPIOn pin. 000b = SDP (software definable input or output) 001b = LED (drives external LEDs) 010b = RoL (reset on LAN) 011b = Timesync 0 100b = Timesync 1 All other values are reserved. When this pin is configured in SDP mode, the output value is set through the GLGEN_GPIO_SET register, and the value of the GPIO pin is read through the GLGEN_GPIO_STAT register.
LED_INVRT	10	0b	RW	LED Invert. This field specifies the polarity/inversion of the LED source prior to output or blink control. By default, the output drives the cathode of the LED so when the LED output is 0b the LED is on. 0b = LED output is active low. 1b = LED output is active high. The *LED_INVRT* setting is meaningful only if the *PIN_FUNC* equals to LED (001b).
LED_BLINK	11	0b	RW	LED Blink. This field specifies whether or not to apply blink logic to the (inverted) LED control source prior to the LED output. 0b = Do not blink LED output 1b = Blink LED output The *LED_BLINK* setting is meaningful only if the *PIN_FUNC* is set to LED (001b).
LED_MODE	16:12	0x0	RW	LED Mode. This field specifies the control source for the LED output. LED modes are described in detail in the LED section. The *LED_MODE* setting is meaningful only if the *PIN_FUNC* is set to LED (001b).
INT_MODE	18:17	00b	RW	Interrupt Mode. This field selects the interrupt mode for this GPIOn pin. 00b = No interrupt. 01b = Interrupt on rising edge. 10b = Interrupt on falling edge. 11b = Interrupt on any transition. Set this field to generate interrupts only when *PIN_FUNC* is set to SDP (000b).
OUT_DEFAULT	19	0b	RW	Default value driven on this GPIOn pin when configured as a SDP output. The value is ignored when GPIO pin is configured as a SDP input or when *PIN_FUNC* is not set to SDP (000b). 0b = Output driven low. 1b = Output driven high.

Field	Bit(s)	Init.	Type	Description
PHY_PIN_NAME	25:20	0x0	RW	This field is used to indicate to firmware the name and functionality for which this GPIO pin is being used when connecting to an external PHY/module control/status pin. 0x0 to 0x7 = Control/status pin names when connecting to an SFP+ module. 0x0 = Rx_LOS 0x1 = Mod_ABS 0x2 = RS0/RS1 0x3 = TxDisable 0x4 = TxFault 0x8 to 0xF = Control status pin names when connecting to a QSFP+ module. 0x8 = IntL 0x9 = ResetL 0xA = ModPresL 0xB = LPMode 0xC = ModSelL 0x10 to 0x17 = Control status pin names when connecting to 10GBASE-T PHYs. 0x10 = PhyInt 0x11 = PhyRst 0x12 = TxDisable 0x3F = SDP is not used for PHY/module connectivity. All other values are reserved.
RESERVED	31:26	0x0	RSV	Reserved.

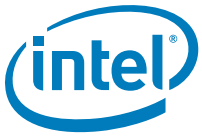
11.2.2.1.5 Global LED Control - GLGEN_LED_CTL (0x00088178; RW)

Global LED Control. Configures the global LED blink mode.

Field	Bit(s)	Init.	Type	Description
GLOBAL_BLINK__MODE	0	0b	RW	Global Blink Mode. This field specifies the blink mode of all LEDs. 0b = Blink at 200 ms on and 200 ms off. 1b = Blink at 83 ms on and 83 ms off.
RESERVED	31:1	0x0	RSV	Reserved.

11.2.2.1.6 Global GPIO Status - GLGEN_GPIO_STAT (0x0008817C; RO)

Field	Bit(s)	Init.	Type	Description
GPIO_VALUE	29:0	0x0	RO	Each bit “n” in this field reflects the value of GPIOn pin (either input or output). For example, Bit[0] reflects the value on GPIO0 pin (1b is high and 0b is low), Bit[1] indicates the value of GPIO1 pin, and so on. See section 3.5 for cross reference of GPIOn to actual pin names.
RESERVED	31:30	00b	RSV	Reserved.



11.2.2.1.7 Global GPIO Transition Status - GLGEN_GPIO_TRANSIT (0x00088180; RW1C)

Field	Bit(s)	Init.	Type	Description
GPIO_TRANSITION	29:0	0x0	RW1C	These register bits are used to latch any transition (low-to-high or high-to-low) on the GPIOn pins (either input or output) since the last time the register bits were cleared. Bit[0] reflects the transition status of GPIO0 pin, Bit[1] indicates the transition status of GPIO1 pin, and so on. See section 3.5 for cross reference of GPIOn to actual pin names. The PF is expected to clear those flags that it cares about by writing 1b to the corresponding bit position. The PF is expected to clear the flags only for the GPIOn pins that it owns.
RESERVED	31:30	00b	RSV	Reserved.

11.2.2.1.8 Global GPIO Set - GLGEN_GPIO_SET (0x00088184; RW)

This Register is used to set the value (0b = Low, 1b = High) of the GPIOn output pins. Attempting to write a value to GPIO input pins has no effect. See Section 3.5 for a cross reference of GPIOn index to actual pin name.

Field	Bit(s)	Init.	Type	Description
GPIO_INDX	4:0	0x0	RW	Defines the GPIO pin number that is driven to *SDP_DATA* value. The *GPIO_INDX* can be set to any value between 0 and 29 decimal (00000b to 11001b). See section 3.5 for a cross reference of GPIOn index to actual pin names. ***Note:*** Software is permitted to drive only GPIO pins that are defined as output SDPs by the GLGEN_GPIO_CTL registers. ***Note:*** Any PFs or EMP can access all GPIOs. It is the PF software responsibility to control only those GPIOs that the PF owns.
SDP_DATA	5	0b	RW	The value in this field is driven to a GPIOn pin pointed by the index *GPIO_INDX*. 0b = Driven low. 1b = Driven high.
DRIVE_SDP	6	0b	RW	Set this flag to 1b to drive the *SDP_DATA* to GPIOn pin pointed by the index *GPIO_INDX*.
RESERVED	31:7	0x0	RSV	Reserved.

11.2.2.1.9 MDI Single Command and Address - GLGEN_MSCA[n] (0x0008818C + 0x4*n, n=0...3; RW)

This register is used for writing the command and address to initiate read/write access over the MDIO interface. This register is used when the MDIOn_SDAn/MDCn_SCLn pins are configured for MDIO operation through the GLGEN_MDIO_I2C_SEL[n] register.

Field	Bit(s)	Init.	Type	Description
MDIADD	15:0	0x0	RW	MDI Address. Address used for new protocol MDI accesses (default = 0000b).
DEVADD	20:16	0x0	RW	Device Type/Register Address. Five bits representing either device type if *STCODE* = 00b or register address if *STCODE* = 01b.
PHYADD	25:21	0x0	RW	PHY Address. The address of the external device.

Field	Bit(s)	Init.	Type	Description
OPCODE	27:26	00b	RW	OpCode. Two bits identifying operation to be performed (default = 00b). 00b = Address cycle (new protocol only). 01b = Write operation. 10b = Read, increment address (new protocol only). 11b = Read operation.
STCODE	29:28	01b	RW	ST Code. Two bits identifying start of frame and old or new protocol (default = 01b). 00b = New protocol 01b = Old protocol 1xb = Illegal
MDICMD	30	0b	RW1C	MDI Command. Perform the MDI operation in this register; cleared when done 0b = MDI ready; operation complete (default). 1b = Perform operation; operation in progress.
MDIINPROGEN	31	0b	RW	MDI In Progress Enable. Generate MDI in progress when operation completes. 0b = MDI ready disable (default). 1b = MDI in progress enable.

11.2.2.1.10 MDI Single Read and Write Data - GLGEN_MSRWD[n] (0x0008819C + 0x4*n, n=0...3; RW)

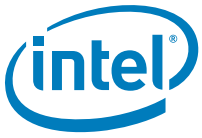
This register is used for reading and writing data to and from the MDIO interface. This register is used when the MDIOn_SDAn/MDCn_SCLn pins are configured for MDIO operation through the GLGEN_MDIO_I2C_SEL[n] register.

Field	Bit(s)	Init.	Type	Description
MDIWRDATA	15:0	0x0	RW	MDI Write Data. Write data For MDI writes to the external device.
MDIRDDATA	31:16	0x0	RO	MDI Read Data. Read data from the external device.

11.2.2.1.11 I^2C Parameters - GLGEN_I2CPARAMS[n] (0x000881AC + 0x4*n, n=0...3; RW)

This register is used to set the parameters for the I^2C access to the 2-wire management interface and to enable bit banging access to the I^2C interface. This register is used when the MDIOn_SDAn/MDCn_SCLn pins are configured for I^2C interface operation through the GLGEN_MDIO_I2C_SEL[n] register.

Field	Bit(s)	Init.	Type	Description
WRITE_TIME	4:0	0x6	RW	Write Time. Defines the delay between a write access and the next access. The value is in ms. A value of 0b is not valid.
READ_TIME	7:5	010b	RW	Read Time. Defines the delay between a read access and the next access. The value is in ms. A value of 0b is not valid.



Field	Bit(s)	Init.	Type	Description
I2CBB_EN	8	0b	RW	I²C Bit Bang Enable. If set, the I2C_CLK and I2C_DATA lines are controlled via the *CLK*, *DATA* and *DATA_OE_N* fields of this register. Otherwise, they are controlled by the hardware machine activated via the GLGEN_I2CCMD[n] register.
CLK	9	1b	RW	I²C CLK out value used to drive the value of I2C_CLK (SCL output to PAD). While in bit bang mode, controls the value driven on the I²C clock pad MDCn_SCLn.
DATA_OUT	10	1b	RW	I²C DATA_OUT value used to drive the value of I²C data (SDA output to PAD). While in bit bang mode and when the DATA_OE_N field is zero, controls the value driven on the I²C data pad MDIOn_SDAn.
DATA_OE_N	11	0b	RW	I²C DATA_OE_N. While in bit bang mode, controls the direction of the I2C_DATA pad MDIOn_SDAn. 0b = Pad is output. 1b = Pad is input.
DATA_IN	12	0b	RO	I²C DATA_IN. Provides the value of I2C_DATA (SDA input from external PAD). This bit is RO. Reflects the value of the I2C_DATA pad MDIOn_SDAn. While in bit bang mode when the *DATA_OE_N* field is zero, this field reflects the value set in the *DATA_OUT* field.
CLK_OE_N	13	0b	RW	I²C Clock Output Enable. While in bit bang mode, controls the direction of the I2C_CLK pad MDCn_SCLn. 0b = Pad is output. 1b = Pad is input.
CLK_IN	14	0b	RO	I²C Clock In Value (SCL input from external PAD). This bit is RO. Reflects the value of the I²C CLK pad MDCn_SCLn. While in bit bang mode when the *CLK_OE_N* field is zero, this field reflects the value set in the *CLK_OUT* field.
CLK_STRETCH_DIS	15	0b	RW	I²C Clock Stretch Disable. 0b = Enable slave clock stretching support in I²C access. 1b = Disable clock stretching support in I²C access.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.1.12 Global Device Timer - GLVFGEN_TIMER (0x000881BC; RW)

Field	Bit(s)	Init.	Type	Description
GTIME	31:0	0x0	RW	GTIME is a free running timer fed by a 1 μs clock.

11.2.2.1.13 Global MDIO or I²C Select - GLGEN_MDIO_I2C_SEL[n] (0x000881C0 + 0x4*n, n=0...3; RW)

This register is used to select between the MDIO interface or I²C interface on the MDIOn_SDAn/ MDCn_SCLn pins. Each pair of these pins can be independently configured for MDIO or I²C interface protocol by using GLGEN_MDIO_I2C_SEL[n] registers.

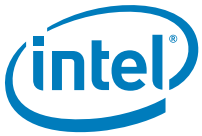
Field	Bit(s)	Init.	Type	Description
MDIO_I2C_SEL	0	0b	RW	The *MDIO_I2C_SEL* bit is used to select between the MDIO interface or I²C interface over the MDIOn_SDAn/MDCn_SCLn pins, where n=0..3. 0b = MDIO interface is selected. 1b = I²C interface is selected.

Field	Bit(s)	Init.	Type	Description
PHY_PORT_NUM	4:1	0x0	RW	Each bit in this field indicates the port number of the external PHY/module accessed through this interface. For MDC/MDIO, one or more PHYs can be managed by the same pair of pins. Hence, more than one bit can be set in this field. In I^2C mode, only a single bit can be set since only a single module might be connected. Bit[0] = Port 0 PHY/Module Bit[1] = Port 1 PHY/Module Bit[2] = Port 2 PHY/Module Bit[3] = Port 3 PHY/Module
PHY0_ADDRESS	9:5	0x0	RW	If this interface is used for connecting to PHY0, this is the address of the PHY device to use. This field is used by firmware.
PHY1_ADDRESS	14:10	0x0	RW	If this interface is used for connecting to PHY1, this is the address of the PHY device to use. This field is used by firmware.
PHY2_ADDRESS	19:15	0x0	RW	If this interface is used for connecting to PHY2, this is the address of the PHY device to use. This field is used by firmware.
PHY3_ADDRESS	24:20	0x0	RW	If this interface is used for connecting to PHY3, this is the address of the PHY device to use. This field is used by firmware.
MDIO_IF_MODE	28:25	0x0	RW	When interface mode is set to MDC/MDIO, this field indicates to firmware the frame structure to use. This field contains a bit per port for scenarios where more than one PHY device is connected through a single interface. Each bit indicates the following options: 0b = Use IEEE Clause 45 Frame Structure. 1b = Use IEEE Clause 22 Frame Structure. Bit definitions: Bit[0] = Port0 *MDIO_IF_MODE* Bit[1] = Port1 *MDIO_IF_MODE* Bit[2] = Port2 *MDIO_IF_MODE* Bit[3] = Port3 *MDIO_IF_MODE*
RSVD	31:29	000b	RSV	Reserved.

11.2.2.1.14 MDIO Control - GLGEN_MDIO_CTRL[n] (0x000881D0 + 0x4*n, n=0...3; RW)

This register is used to control the MDC clock speed and mode when the MDIOn_SDAn/MDCn_SCLn pins are configured for use as a MDIO interface. Each pair of these pins are independently configured as a MDIO or I^2C interface through the GLGEN_MDIO_I2C_SEL[n] registers, where n=0..3.

Field	Bit(s)	Init.	Type	Description
LEGACY_RSVD2	16:0	0x2FFB	RW	Reserved. Do not modify this value.
CONTMDC	17	0b	RW	Continuous MDC. Turn off MDC between MDIO packets. 0b = MDC off between packets (default). 1b = Continuous MDC.
LEGACY_RSVD1	31:18	0x400	RW	Reserved. Do not modify this value.



11.2.2.1.15 I²C Command - GLGEN_I2CCMD[n] (0x000881E0 + 0x4*n, n=0...3; RW)

This register is used to read or write to the configuration registers over the I²C interface when the MDIOn_SDAn/MDCn_SCLn pins are configured for I²C operation. Each pair of these pins are independently configured as a MDIO or I²C interface through the GLGEN_MDIO_I2C_SEL[n] registers, where n=0..3.

Field	Bit(s)	Init.	Type	Description
DATA	15:0	0x0	RW	I²C Data. For a write command, firmware/software places the data bits in this field and hardware shifts them out to the I²C bus. For a read command, hardware reads the bits serially from the I²C bus and places the data in this field so firmware/software can fetch the data from this location. ***Note:*** This field is read in byte order and not in word order. Ordering can be changed by using GLGEN_I2CPARAMS.*I2C_DATA_ORDER* bit.
REGADD	23:16	0x0	RW	I²C Register Address. For example, register 0, 1, 2... 255.
PHYADD	26:24	000b	RW	Device Address Bits 2:0 The actual address used is 1010b (PHYADD[2:0]).
OP	27	0b	RW	Op Code. 0b = I²C write 1b = I²C read
RESET	28	0b	RW1C	Reset Sequence. If set, sends a reset sequence before the actual read or write. This bit is self clearing. A reset sequence is defined as nine consecutive stop conditions.
R	29	0b	RW	Ready Bit. Indicates a read or write operation on the I²C interface has completed. Set to 1b by hardware at the end of the I²C transaction. Reset by a firmware/software write of a new command to this register.
RSVD	30	0b	RSV	Reserved.
E	31	0b	RW	Error. This bit set is to 1b by hardware when it fails to complete an I²C read. Reset by a firmware/software write of a new command. ***Note:*** This bit is valid only when Ready bit R is set to 1b by hardware.

11.2.2.1.16 VM Reset Trigger - VSIGEN_RTRIG[VSI] (0x00090000 + 0x4*VSI, VSI=0...383; RW)

Field	Bit(s)	Init.	Type	Description
VMSWR	0	0b	RW	VM software reset is initiated by setting the VMR bit. At completion of the reset flow, the PF software clears this bit.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.1.17 VM Reset Status - VSIGEN_RSTAT[VSI] (0x00090800 + 0x4*VSI, VSI=0...383; RO)

Field	Bit(s)	Init.	Type	Description
VMRD	0	1b	RO	VM Software Reset Done Indication. This flag is cleared when the VM reset is initiated (by setting the *VMSWR* flag in the matched VSIGEN_RTRIG register or any stronger reset that affects the VM). It is set back to 1b when hardware completes its VM reset sequence.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.1.18 VF Reset Trigger - VPGEN_VFRTRIG[VF] (0x00091800 + 0x4*VF, VF=0...127; RW)

This register affects the VF but exposed only to the parent PF.

Field	Bit(s)	Init.	Type	Description
VFSWR	0	0b	RW	VF software reset is done by the PF setting the *VFSWR* bit. At reset completion, the PF clears this bit.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.1.19 VF Reset Status - VPGEN_VFRSTAT[VF] (0x00091C00 + 0x4*VF, VF=0...127; RO)

This register affects the VF but exposed only to the parent PF.

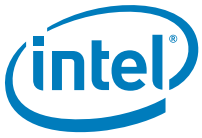
Field	Bit(s)	Init.	Type	Description
VFRD	0	1b	RO	VF Software Reset Done Indication. This flag is cleared when the VF reset is initiated (by setting the *VFSWR* flag in the matched GLGEN_VFRTRIG register or any stronger reset that affects the VF). It is set back to 1b when the hardware completes its VF reset sequence.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.1.20 PF Control - PFGEN_CTRL (0x00092400; RW)

Field	Bit(s)	Init.	Type	Description
PFSWR	0	0b	RW	PF software reset is done by setting the *PFSWR* bit. At reset completion, hardware clears this bit.
RESERVED	31:1	0x0	RSV	Reserved.

11.2.2.1.21 PF Driver Unload - PFGEN_DRUN (0x00092500; RW)

Field	Bit(s)	Init.	Type	Description
DRVUNLD	0	0b	RW	The PF sets this bit to indicate its driver is unloading.
RESERVED	31:1	0x0	RSV	Reserved.



11.2.2.1.22 Global VF Level Reset Status - GLGEN_VFLRSTAT[n] (0x00092600 + 0x4*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
VFLRE	31:0	0x0	RW1C	VFLR Status Indication. Bit 'm' in register 'n' reflects a VFLR event on VF index '32 x n + m', where 'n' is the register index and 'm' is the bit index.

11.2.2.1.23 Global Status - GLGEN_STAT (0x000B612C; RO)

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	00b	RSV	Reserved.
DCBEN	2	0b	RW	Global DCB Enable. Defines if DCB is enabled. Loaded from NVM. Can be overridden by Firmware. 0b = DCB disabled. 1b = DCB enabled.
VTEN	3	0b	RW	Global VT Enable. Defines if the Virtualization offloads are enabled. Loaded from NVM. Can be overridden by Firmware. 0b = VT disabled. 1b = VT enabled.
FCOEN	4	0b	RW	Global FCoE Enable. Defines if FCoE offload is enabled. Loaded from the NVM. Can be overridden by firmware. 0b = FCoE offload disabled. 1b = FCoE offload enabled. Used by firmware only.
EVBEN	5	0b	RW	Global EVB Enable. Defines if the PE structures (s-comp/M-comp) offloads are enabled. 0b = EVB disabled. 1b = EVB enabled.
RESERVED	31:6	0x0	RSV	Reserved.

11.2.2.1.24 General Port Status - PRTGEN_STATUS (0x000B8100; RO)

This register contains general port status information.

Field	Bit(s)	Init.	Type	Description
PORT_VALID	0	0b	RO	Port Valid. Denotes if the respective Ethernet port is enabled. 0b = Port is disabled. 1b = Port is enabled.

Field	Bit(s)	Init.	Type	Description
PORT_ACTIVE	1	0b	RO	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	Reserved.

11.2.2.1.25 General Port Configuration - PRTGEN_CNF (0x000B8120; RO)

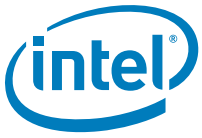
This register contains configuration per Ethernet port loaded from NVM.

Field	Bit(s)	Init.	Type	Description
PORT_DIS	0	1b	RW	Port Disable. Defines if the Ethernet port is enabled from the NVM. Exception: Port 0 is always enabled and cannot be disabled from the NVM. 0b = Enabled 1b = Disabled Defaults: 0b for port 0. 1b for other ports.
ALLOW_PORT_DIS	1	0b	RW	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	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	Reserved.

11.2.2.1.26 General Port Configuration2 - PRTGEN_CNF2 (0x000B8160; RO)

This register contains configuration per Ethernet port loaded from NVM.

Field	Bit(s)	Init.	Type	Description
ACTIVATE_PORT_LINK	0	0b	RW	When this field is set to 0 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. ***Note:*** The PCIe functions associated with the port are not affected by the link loss. ***Note:*** 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.
RSVD	31:1	0x0	RSV	Reserved.



11.2.2.1.27 Global Reset Delay - GLGEN_RSTCTL (0x000B8180; RO)

Field	Bit(s)	Init.	Type	Description
GRSTDEL	5:0	0x8	RW	Global/Core and EMP resets delay defined in 100 ms units. Setting *GRSTDEL* to zero bypasses the delay counter.
RSVD	7:6	00b	RSV	Reserved.
ECC_RST_ENA	8	0b	RW	Graceful GLOBR 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 GLOBR.
RSVD	31:9	0x0	RSV	Reserved.

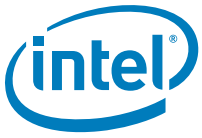
11.2.2.1.28 Global Clock Status - GLGEN_CLKSTAT (0x000B8184; RO)

Field	Bit(s)	Init.	Type	Description
CLKMODE	0	0b	RW	Clock Mode. Controls the operating mode of internal clocks: 0b = Performance mode — Device operates at its highest internal frequencies, independent of the Ethernet link speed. 1b = Power mode — Device adjusts its internal frequencies to match the speed of the Ethernet link.
RESERVED	3:1	000b	RSV	Reserved.
U_CLK_SPEED	5:4	00b	RO	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 MH 11b = Reserved
RESERVED	7:6	00b	RSV	Reserved.
P0_CLK_SPEED	10:8	000b	RO	Debug Feature. Represents the current speed of the Rx clock for MAC 0. 3 bits per port (synchronized to the equivalent MAC clock). Speeds are represented as follows: 000b = 100 Mb/s 001b = 1 GbE 010b = 10 GbE 011b = 40 GbE All other values are reserved.
RESERVED	11	0b	RSV	Reserved.
P1_CLK_SPEED	14:12	000b	RO	Debug Feature. Represents the current speed of the Rx clock for MAC 1. 3 bits per port (synchronized to the equivalent MAC clock). Speeds are represented as follows: 000b = 100 Mb/s 001b = 1 GbE 010b = 10 GbE All other values are reserved.
RESERVED	15	0b	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
P2_CLK_SPEED	18:16	000b	RO	Debug Feature. Represents the current speed of the Rx clock for MAC 2. 3 bits per port (synchronized to the equivalent MAC clock). Speeds are represented as follows: 000b = 100 Mb/s 001b = 1 GbE 010b = 10 GbE 011b = 40 GbE All other values are reserved.
RESERVED	19	0b	RSV	Reserved.
P3_CLK_SPEED	22:20	000b	RO	Debug Feature. Represents the current speed of the Rx clock for MAC 3. 3 bits per port (synchronized to the equivalent MAC clock). Speeds are represented as follows: 000b = 100 MB. 001b = 1 GbE. 010b = 10 GbE. 011b = 40 GbE. All other values are reserved.
RESERVED	31:23	0x0	RSV	Reserved.

11.2.2.1.29 Global Reset Status - GLGEN_RSTAT (0x000B8188; RO)

Field	Bit(s)	Init.	Type	Description
DEVSTATE	1:0	00b	RO	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	Reflects one of the following resets that are/were in progress: 00b = POR 01b = CORER 10b = GLOBR 11b = EMPR
CORERCNT	5:4	00b	RO	Counts the number of initiated core resets since POR. The counter wraps around from 11b to 00b.
GLOBRCNT	7:6	00b	RO	Counts the number of initiated global resets since POR. The counter wraps around from 11b to 00b.
EMPRCNT	9:8	00b	RO	Counts the number of initiated EMP resets since POR. The counter wraps around from 11b to 00b.
TIME_TO_RST	15:10	0x0	RO	The reset time is a down counter, loaded by GLRSTDEL following a GLOBR or CORER or EMPR. When GLOBRTIME reaches a zero value the actual reset is initiated.
RSVD	31:16	0x0	RSV	Reserved.



11.2.2.1.30 Global EMP Reset Enable - GLGEN_RSTENA_EMP (0x000B818C; RO)

This register is accessible only by the EMP.

Field	Bit(s)	Init.	Type	Description
EMP_RST_ENA	0	0b	RW	When *EMP_RST_ENA* is active, setting the *EMPFWR* bit (by the PFs) in the EMPGEN_RSTCTL triggers an EMP reset. When the *EMP_RST_ENA* is cleared, setting the *EMPFWR* bit (by the PFs) in the EMPGEN_RSTCTL has no impact. Firmware is expected to set the *EMP_RST_ENA* flag as one of the very first steps during the internal development phase. During normal operation, this flag always remains cleared.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.1.31 Global Reset Trigger - GLGEN_RTRIG (0x000B8190; RW)

Field	Bit(s)	Init.	Type	Description
CORER	0	0b	RW	Setting the *CORER* triggers a graceful core reset flow.
GLOBR	1	0b	RW	Setting the *GLOBR* triggers a graceful global reset flow.
RSVD	31:2	0x0	RSV	Reserved.

11.2.2.1.32 LAN Port Number - PFGEN_PORTNUM (0x001C0480; RO)

Field	Bit(s)	Init.	Type	Description
PORT_NUM	1:0	00b	RW	Port Number. Indicates the LAN port connected to this function. 00b = Port 0 01b = Port 1 10b = Port 2 11b = Port 3
RSVD	31:2	0x0	RSV	Reserved.

11.2.2.1.33 PCI Function Count - GLGEN_PCIFCNCNT (0x001C0AB4; RO)

Field	Bit(s)	Init.	Type	Description
PCIPFCNT	4:0	0x1	RW	Reports the function number of the highest PCI physical function plus 1 as it is loaded from the NVM. For example, if the NVM enables 2 functions 0 and 7, the value is 8 (7+1). Used to partition the interrupt vectors among supported PFs.
RESERVED	15:5	0x0	RSV	Reserved.
PCIVFCNT	23:16	0x0	RW	Reports the function number of highest PCI virtual function plus 1 for all PFs combined, as it is loaded from the NVM. Used to partition the interrupt vectors among supported VFs.
RESERVED	31:24	0x0	RSV	Reserved.

11.2.2.2 PF - PCIe Registers

11.2.2.2.1 Function Requester ID Information Register - PF_FUNC_RID (0x0009C000; RO)

Field	Bit(s)	Init.	Type	Description
FUNCTION_NUMBER	2:0	000b	RO	Function number assigned to the function based on BIOS/OS enumeration.
DEVICE_NUMBER	7:3	0x0	RO	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	Bus number assigned to the function based on BIOS/OS enumeration.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.2.2 PF PCIe Configuration Indirect Access Address - PF_PCI_CIAA (0x0009C080; RW)

Field	Bit(s)	Init.	Type	Description
ADDRESS	11:0	0x0	RW	The configuration space address to access.
VF_NUM	18:12	0x0	RW	Defines the VF number to access. The VF number is the absolute VF number in the device.
RESERVED	31:19	0x0	RSV	Reserved. Ignore on read. Write 0b.

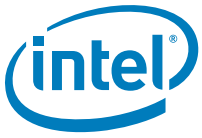
11.2.2.2.3 PCIe Configuration Indirect Access Data - PF_PCI_CIAD (0x0009C100; RW)

Field	Bit(s)	Init.	Type	Description
DATA	31:0	0x0	RW	This register is 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*.

11.2.2.2.4 Function Active and Power State - PFPCI_FACTPS (0x0009C180; RO)

This register is provided for internal firmware use, notifying firmware about function state.

Field	Bit(s)	Init.	Type	Description
FUNC_POWER_STATE	1:0	00b	RO	Power state indication of the function. 00b = Dr 01b = D3 10b = D0a 11b = D0u



Field	Bit(s)	Init.	Type	Description
RESERVED	2	0b	RSV	Reserved.
FUNC_AUX_EN	3	0b	RO	Function Aux Enable. Reflects the Auxiliary Power PM Enable bit from the PCI configuration space.
RESERVED	31:4	0x0	RSV	Reserved.

11.2.2.2.5 PCIe Interrupt Cause - PFPCI_ICAUSE (0x0009C200; RW1C)

Field	Bit(s)	Init.	Type	Description
PCIE_ERR_CAUSE	31:0	0x0	RW1C	Each bit corresponds to a particular error event as described in the section Proprietary Error Reporting.

11.2.2.2.6 PCIe Interrupts Enable - PFPCI_IENA (0x0009C280; RW)

Field	Bit(s)	Init.	Type	Description
PCIE_ERR_EN	31:0	0x0	RW	Each bit corresponds to a particular error event as described in the section Proprietary Error Reporting.

11.2.2.2.7 PCIe VM Pending Index - PFPCI_VMINDEX (0x0009C300; RW)

Field	Bit(s)	Init.	Type	Description
VMINDEX	8:0	0x0	RW	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 383. It can only be set by software to VMs that the PF owns and only to VMs that are not assigned to a VF.
RSVD	31:9	0x0	RSV	Reserved.

11.2.2.2.8 PCIe VM Pending Status - PFPCI_VMPEND (0x0009C380; RO)

Field	Bit(s)	Init.	Type	Description
PENDING	0	0b	RO	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.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.2.9 PCIe Default Revision ID - GLPCI_DREVID (0x0009C480; RO)

Field	Bit(s)	Init.	Type	Description
DEFAULT_REVID	7:0	0x01	RO	Mirroring of default Rev ID prior to an NVM load.
RESERVED	31:8	0x00	RSV	Reserved.

11.2.2.2.10 PCIe Byte Counter High - GLPCI_BYTCTH (0x0009C484; RO)

A byte counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_BW_BCT	31:0	0x0	RO	This register contains the high double word of a 64-bit counter that counts PCIe payload bytes This register gets stuck at its maximum value of 0xFF...F.

11.2.2.2.11 PCIe Byte Counter Low - GLPCI_BYTCTL (0x0009C488; RO)

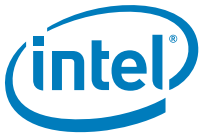
A byte counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_BW_BCT	31:0	0x0	RO	This register contains the high double word of a 64-bit counter that counts PCIe payload bytes This register gets stuck at its maximum value of 0xFF...F.

11.2.2.2.12 PCIe Statistic Control Register #1 - GLPCI_GSCL_1 (0x0009C48C; RW)

This register controls the operation of the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
GIO_COUNT_EN_0	0	0b	RW	Enables PCIe statistic counter number 0.
GIO_COUNT_EN_1	1	0b	RW	Enables PCIe statistic counter number 1.
GIO_COUNT_EN_2	2	0b	RW	Enables PCIe statistic counter number 2.
GIO_COUNT_EN_3	3	0b	RW	Enables PCIe statistic counter number 3.
LBC_ENABLE_0	4	0b	RW	When set, statistics counter 0 operates in leaky bucket mode. When cleared, leaky bucket mode is disabled and the counter is incremented by one for each event.
LBC_ENABLE_1	5	0b	RW	When set, statistics counter 1 operates in leaky bucket mode. When cleared, leaky bucket mode is disabled and the counter is incremented by one for each event.
LBC_ENABLE_2	6	0b	RW	When set, statistics counter 2 operates in leaky bucket mode. When cleared, leaky bucket mode is disabled and the counter is incremented by one for each event.
LBC_ENABLE_3	7	0b	RW	When set, statistics counter 3 operates in leaky bucket mode. When cleared, leaky bucket mode is disabled and the counter is incremented by one for each event.
RESERVED	13:8	0x0	RSV	Reserved
RESERVED	14			Reserved
RESERVED	19:15			Reserved
RESERVED	27:20	0x0	RSV	Reserved.
GIO_64_BIT_EN	28	0b	RW	Enables two 64-bit counters instead of four 32-bit counters.
GIO_COUNT_RESET	29	0b	RW1S	Reset indication of PCIe statistic counters. Reading this bit returns 0b.
GIO_COUNT_STOP	30	0b	RW1S	Stop indication of PCIe statistic counters. Reading this bit returns 0b.



Field	Bit(s)	Init.	Type	Description
GIO_COUNT_START	31	0b	RW1S	Start indication of PCIe statistic counters. Reading this bit returns 0b.

11.2.2.2.13 PCIe Statistic Control Register #2 - GLPCI_GSCL_2 (0x0009C490; RW)

This register defines the events counted by the performance counters.

Field	Bit(s)	Init.	Type	Description
GIO_EVENT_NUM_0	7:0	0x0	RW	Event number that counter 0 counts (GSCN_0).
GIO_EVENT_NUM_1	15:8	0x0	RW	Event number that counter 1 counts (GSCN_1).
GIO_EVENT_NUM_2	23:16	0x0	RW	Event number that counter 2 counts (GSCN_2).
GIO_EVENT_NUM_3	31:24	0x0	RW	Event number that counter 3 counts (GSCN_3).

11.2.2.2.14 PCIe Statistic Control Registers #5...#8 - GLPCI_GSCL_5_8[n] (0x0009C494 + 0x4*n, n=0...3; RW)

These registers control the operation of the leaky bucket counter n.

GSCL_5 corresponds to n=0
GSCL_6 corresponds to n=1
GSCL_7 corresponds to n=2
GSCL_8 corresponds to n=3

Field	Bit(s)	Init.	Type	Description
LBC_THRESHOLD_N	15:0	0x0	RW	Threshold for the leaky bucket counter n.
LBC_TIMER_N	31:16	0x0	RW	Time period between decrementing the value in leaky bucket Counter n. The time period is defined in us units.

11.2.2.2.15 PCIe Statistic Counter Registers #0...#3 - GLPCI_GSCN_0_3[n] (0x0009C4A4 + 0x4*n, n=0...3; RO)

These registers contain the performance counters 0-3.

GSCL_0 corresponds to n=0
GSCL_1 corresponds to n=1
GSCL_2 corresponds to n=2
GSCL_3 corresponds to n=3

Field	Bit(s)	Init.	Type	Description
EVENT_COUNTER	31:0	0x0	RO	A 32-bit event counter. See the section on Performance and Statistics Counters. These registers are stuck at their maximum value of 0xFF...F.

11.2.2.2.16 PCIe Packet Counter - GLPCI_PKTCT (0x0009C4BC; RO)

A packet counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_BW_PCT	31:0	0x0	RO	A 32-bit counter that counts PCIe packets. This register gets stuck at its maximum value of 0xFF...F.

11.2.2.2.17 PCIe PQs Max Used Space - GLPCI_PQ_MAX_USED_SPC (0x0009C4EC; RO)

Field	Bit(s)	Init.	Type	Description
GLPCI_PQ_MAX_USED_SPC_12	7:0	0x0	RO	
GLPCI_PQ_MAX_USED_SPC_13	15:8	0x0	RO	
RSVD	31:16	0x0	RSV	Reserved.

11.2.2.2.18 PCIe Mux Selector for PFB - GLPCI_PM_MUX_PFB (0x0009C4F0; RW)

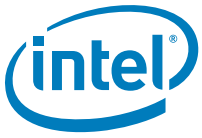
Field	Bit(s)	Init.	Type	Description
PFB_PORT_SEL	4:0	0x0	RW	
RSVD_0	15:5	0x0	RSV	Reserved.
INNER_PORT_SEL	18:16	000b	RW	
RSVD_1	31:19	0x0	RSV	Reserved.

11.2.2.2.19 PCIe Mux Selector for NPQs - GLPCI_PM_MUX_NPQ (0x0009C4F4; RW)

Field	Bit(s)	Init.	Type	Description
NPQ_NUM_PORT_SEL	2:0	000b	RW	
RSVD_0	15:3	0x0	RSV	Reserved.
INNER_NPQ_SEL	20:16	0x0	RW	
RSVD_1	31:21	0x0	RSV	Reserved.

11.2.2.2.20 PCIe Regs Spare Bits 0 - GLPCI_SPARE_BITS_0 (0x0009C4F8; RW)

Field	Bit(s)	Init.	Type	Description
SPARE_BITS	31:0	0x0	RW	



11.2.2.2.21 PCIe Regs Spare Bits 1 - GLPCI_SPARE_BITS_1 (0x0009C4FC; RW)

Field	Bit(s)	Init.	Type	Description
SPARE_BITS	31:0	0x0	RW	

11.2.2.2.22 PCIe VF Flush Done - PFPCI_VF_FLUSH_DONE[VF] (0x0009C600 + 0x4*VF, VF=0...127; RO)

Field	Bit(s)	Init.	Type	Description
FLUSH_DONE	0	0b	RO	
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.2.23 PCIe PF Flush Done - PFPCI_PF_FLUSH_DONE (0x0009C800; RO)

Field	Bit(s)	Init.	Type	Description
FLUSH_DONE	0	0b	RO	
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.2.24 PCIe VM Flush Done - PFPCI_VM_FLUSH_DONE (0x0009C880; RO)

Field	Bit(s)	Init.	Type	Description
FLUSH_DONE	0	0b	RO	
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.2.25 PCIe PF Configuration - PFPCI_CNF (0x000BE000; RO)

Contains per-PF configuration loaded from NVM.

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	00b	RSV	Reserved.
MSI_EN	2	1b	RW	Enables the MSI capability structure for this PCI function. 0b = MSI is disabled. 1b = MSI is enabled.
EXROM_DIS	3	0b	RW	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	I/O BAR Support. 0b = I/O BAR is not supported. 1b = I/O BAR is supported.

Field	Bit(s)	Init.	Type	Description
INT_PIN	6:5	00b	RW	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	Reserved.

11.2.2.2.26 PCIe PF Device ID - PFPCI_DEVID (0x000BE080; RO)

Contains the per-PF Device ID.

Field	Bit(s)	Init.	Type	Description
PF_DEV_ID	15:0	0x154B	RW	Contains the device ID for this PF.
VF_DEV_ID	31:16	0x154B	RW	Contains the device ID for the VFs of this PF.

11.2.2.2.27 PFPCIe Subsystem ID - PFPCI_SUBSYSID (0x000BE100; RO)

Field	Bit(s)	Init.	Type	Description
PF_SUBSYS_ID	15:0	0x0	RW	Subsystem ID for this PF
VF_SUBSYS_ID	31:16	0x0	RW	Subsystem ID for VFs of this PF

11.2.2.2.28 PCIe Functions Configuration 2 - PFPCI_FUNC2 (0x000BE180; RO)

Field	Bit(s)	Init.	Type	Description
RESERVED	31:0	0x0	RSV	Reserved.

11.2.2.2.29 PCIe Functions Configuration - PFPCI_FUNC (0x000BE200; RO)

Field	Bit(s)	Init.	Type	Description
FUNC_DIS	0	1b	RW	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	Allow Function Disable. 0b = Asserting PCI_DIS_N has no effect on this PCI function. 1b = Asserting PCI_DIS_N disables this PCI function.

Field	Bit(s)	Init.	Type	Description
DIS_FUNC_ON_PORT_DIS	2	0b	RW	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	Reserved.

11.2.2.2.30 PCIe Function Status 1 - PFPCI_STATUS1 (0x000BE280; RO)

Field	Bit(s)	Init.	Type	Description
FUNC_VALID	0	0b	RO	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	Reserved.

11.2.2.2.31 PCIe PM - PFPCI_PM (0x000BE300; RW)

Field	Bit(s)	Init.	Type	Description
PME_EN	0	0b	RW	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	Reserved.

11.2.2.2.32 PCIe Storage Class - PFPCI_CLASS (0x000BE400; RO)

Contains per-PF configuration loaded from NVM.

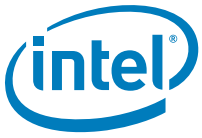
Field	Bit(s)	Init.	Type	Description
STORAGE_CLASS	0	0b	RW	0b = The class code of this function is set to 0x020000 (LAN). 1b = The class code of this function is set to 0x010000 (SCSI).
RESERVED_1	1	0b	RW	Reserved.
PF_IS_LAN	2	0b	RW	0b = SAN function. 1b = LAN function.
RESERVED	31:3	0x0	RSV	Reserved.

11.2.2.2.33 TPH Control Register - GLTPH_CTRL (0x000BE480; RW)

Field	Bit(s)	Init.	Type	Description
RESERVED	8:0	0x0	RSV	Reserved.
DESC_PH	10:9	00b	RW	Descriptor PH. Defines the PH field used when a TPH hint is given for descriptor associated traffic (descriptor fetch, descriptor write back or head write back). The default value should be kept for regular operation
DATA_PH	12:11	10b	RW	Data PH. Defines the PH field used when a TPH hint is given for data associated traffic (Tx data read, Rx data write). The default value should be kept for regular operation
RESERVED	31:13	0x0	RSV	Reserved.

11.2.2.2.34 PCI BAR Control - GLPCI_LBARCTRL (0x000BE484; RO)

Field	Bit(s)	Init.	Type	Description
PREFBAR	0	1b	RW	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.
BAR32	1	0b	RW	BAR 32-bit Enable. 0b = 64-bit BAR addressing mode is selected. 1b = 32-bit BARs are enabled.
RSVD	2	0b	RSV	Reserved.
FLASH_EXPOSE	3	1b	RW	When set, the Flash memory is accessible through the memory BAR.
RESERVED	5:4	00b	RSV	Reserved
FL_SIZE	8:6	111b	RW	Indicates the size of the external Flash as = 64 KB x (2 ** FL_SIZE). The following values are supported: 101b = 2 MB 110b = 4 MB 111b = 8 MB All other values are reserved.
RSVD	10:9	00b	RSV	Reserved.
EXROM_SIZE	13:11	011b	RW	This field indicates the size of the expansion ROM BAR as = 64 KB x (2 ** EXROM_SIZE). 000b = 64 KB 001b = 128 KB 010b = 256 KB 011b = 512 KB 100b = 1 MB 101b = 2 MB 110b = 4 MB 111b = 8 MB Default value is 512 KB.
RSVD	31:14	0x0	RSV	Reserved.



11.2.2.2.35 PCIe Subsystem ID - GLPCI_SUBVENID (0x000BE48C; RO)

Field	Bit(s)	Init.	Type	Description
SUB_VEN_ID	15:0	0x8086	RW	Loaded to the PCI configuration Subsystem Vendor ID register.
RESERVED	31:16	0x0	RSV	Reserved

11.2.2.2.36 PCIe Power Data Register - GLPCI_PWRDATA (0x000BE490; RO)

Field	Bit(s)	Init.	Type	Description
D0_POWER	7:0	0x0	RW	The value in this field is reflected in the PCI Power Management Data register of the LAN functions for D0 power consumption and dissipation (Data_Select = 0 or 4).
COMM_POWER	15:8	0x0	RW	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	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	The value in this field reflects the Data_Scale field in the PCI PMCSR register.
RESERVED	31:26	0x0	RSV	Reserved.

11.2.2.2.37 PCIe Global Config 2 - GLPCI_CNF2 (0x000BE494; RO)

This register contains global status fields of PCIe configuration.

Field	Bit(s)	Init.	Type	Description
RO_DIS	0	0b	RW	Relaxed Ordering Disable. When set to 1b, the device does not request any relaxed ordering transactions. When this bit is cleared, relaxed ordering is specified per request type.
CACHELINE_SIZE	1	0b	RW	Cache Line Size. Determines the system cache line size. 0b = 64 bytes 1b = 128 bytes This field is loaded from the NVM.
MSI_X_PF_N	12:2	0x80	RW	MSI_X PF Table Size. System software reads this field to determine the MSI-X table size N, which is encoded as N-1. This field is loaded from the NVM *MSI_X_N_PF* field and reflects the same field in the PCIe MSI-X configuration (Message Control register).
MSI_X_VF_N	23:13	0x04	RW	MSI_X VF Table Size.System software reads this field to determine the MSI-X table size N for VFs, which is encoded as N-1. This field is loaded from the NVM *MSI_X_N_VF* field and reflects the same field in the PCIe MSI-X configuration (VF MSI-X Control register).
RESERVED	31:24	0x0	RSV	Reserved.

11.2.2.2.38 PCIe Serial Number MAC Address Low - GLPCI_SERL (0x000BE498; RO)

Field	Bit(s)	Init.	Type	Description
SER_NUM_L	31:0	0x0	RW	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 4th 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_SAL1|PFPM_SAL0] words of the EMP Settings Module.

11.2.2.2.39 PCIe Serial Number MAC Address High - GLPCI_SERH (0x000BE49C; RO)

Field	Bit(s)	Init.	Type	Description
SER_NUM_H	15:0	0x0	RW	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 5th 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	Reserved.

11.2.2.2.40 PCIe Capabilities Control - GLPCI_CAPCTRL (0x000BE4A4; RW)

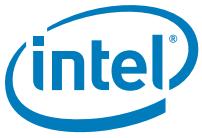
Determines PCIe capabilities supported by the device, and that SW is allowed to enable or disable.

Field	Bit(s)	Init.	Type	Description
VPD_EN	0	0b	RW	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	Reserved.

11.2.2.2.41 PCIe Capabilities Support - GLPCI_CAPSUP (0x000BE4A8; RO)

Determines PCIe capabilities supported by the device.

Field	Bit(s)	Init.	Type	Description
PCIE_VER	0	1b	RW	Determines the PCIe capability version. 0b = Capability version: 0x1 1b = Capability version: 0x2
RESERVED	1	0b	RSV	Reserved.
LTR_EN	2	0b	RW	A value of 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	A value of 1b indicates support for the PCIe TPH requester capability.
ARI_EN	4	1b	RW	A value of 1b indicates support for PCIe ARI capability.
IOV_EN	5	1b	RW	A value of 1b indicates support for PCIe SR-IOV capability.
ACS_EN	6	1b	RW	A value of 1b indicates support for PCIe ACS capability.



Field	Bit(s)	Init.	Type	Description
SEC_EN	7	1b	RW	A value of 1b indicates support for the secondary PCI express extended capability.
RESERVED	15:8	0x0	RSV	Reserved.
ECRC_GEN_EN	16	1b	RW	Loaded into the ECRC. Generation capable bit of the PCIe configuration registers.
ECRC_CHK_EN	17	1b	RW	Loaded into the ECRC Check capable bit of the PCIe configuration registers.
IDO_EN	18	1b	RW	Enables ID-based ordering (IDO).
MSI_MASK	19	1b	RW	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	Enables access to CSRs via the PCI configuration space. See the section on Configuration Access to Internal Registers and Memories.
RESERVED	29:21	0x0	RSV	Reserved.
LOAD_SUBSYS_ID	30	0b	RW	Load Subsystem IDs. When set to 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	Load Device ID. When set to 1b, indicates that the device loads its PCI device ID's from the NVM.

11.2.2.2.42 PCIe Link Capabilities - GLPCI_LINKCAP (0x000BE4AC; RO)

Determines PCIe Link capabilities supported by the device.

Field	Bit(s)	Init.	Type	Description
LINK_SPEEDS_VECTOR	5:0	0x0	RW	Supported Link Speeds Vector. Loaded to the Link Capabilities 2 register in the PCIe capability values: Bit[0] = 5.0 GT/s. Bit[1] = 8.0 GT/s. Bits[5:2] = Reserved. ***Note:*** 2.5 GT/s is always supported.
MAX_PAYLOAD	8:6	100b	RW	Max Payload Size Supported. Loaded to the PCIe Device Capabilities register. Supported values are 000b to 100b (128 bytes to 2 KB). Otherwise, keep the hardware default.
MAX_LINK_WIDTH	12:9	0x7	RW	Max Link Width. Loaded to the PCIe Link Capabilities register Values are: 0001b = Limit max link width to x1. 0011b = Limit max link width to x4. 0100b = Limit max link width to x8. 0111b = Do not limit max link width. Negotiate to the max width supported by the link. All other values are reserved.
RESERVED	31:13	0x0	RSV	Reserved.

11.2.2.2.43 PCIe PM Support - GLPCI_PMSUP (0x000BE4B0; RO)

This register contains parameters that define PCIe power management support.

Field	Bit(s)	Init.	Type	Description
ASPM_SUP	1:0	11b	RW	Active state link PM support is loaded to the PCIe Link Capabilities register.
L0S_EXIT_LAT	4:2	101b	RW	L0s Exit Latency. Loaded to L0s Exit Latency field in the PCIe Link Capabilities register.
L1_EXIT_LAT	7:5	100b	RW	L1 Exit Latency. Loaded to L1 Exit Latency field in the PCIe Link Capabilities register.
L0S_ACC_LAT	10:8	011b	RW	Loaded to the Endpoint L0s Acceptable Latency field in the PCIe Device Capabilities register.
L1_ACC_LAT	13:11	110b	RW	Loaded to the Endpoint L1 Acceptable Latency field in the PCIe Device Capabilities register.
SLOT_CLK	14	1b	RW	Slot Clock Configuration. Loaded to the PCIe Link Status register.
OBFF_SUP	16:15	00b	RW	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	Reserved.

11.2.2.2.44 PCIe Revision ID - GLPCI_REVID (0x000BE4B4; RO)

This register is shared by all physical functions.

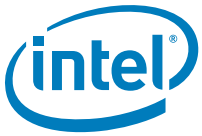
Field	Bit(s)	Init.	Type	Description
NVM_REVID	7:0	0x0	RW	Value of the Rev ID loaded from the NVM.
RESERVED	31:8	0x0	RSV	Reserved.

11.2.2.2.45 PCIe VF Capabilities Support - GLPCI_VFSUP (0x000BE4B8; RO)

Field	Bit(s)	Init.	Type	Description
VF_PREFETCH	0	1b	RW	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.
VR_BAR_TYPE	1	1b	RW	VF BAR Type. 0b = VF BARs advertise 32-bit size. 1b = VF BARs advertise 64-bit size.
RESERVED	31:2	0x0	RSV	Reserved.

11.2.2.2.46 PCIe Global Config - GLPCI_CNF (0x000BE4C0; RO)

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	00b	RSV	Reserved.
WAKE_PIN_EN	2	0b	RW	When set to 1b, enables the use of the WAKE pin for a PME event in all power states.
RESERVED	31:3	0x0	RSV	Reserved.



11.2.2.2.47 PCIe Upper Address - GLPCI_UPADD (0x000BE4F8; RW)

This register is used to block PCIe master accesses above some address.

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
ADDRESS	31:1	0x0	RW	Address. Bits [31:1] correspond to bits [63:33] in the PCIe address space, respectively.

11.2.2.2.48 PCIe Errors Reported - GLPCI_PCIERR (0x000BE4FC; RO)

This register indicates which PCIe errors are reported to device SW.

Field	Bit(s)	Init.	Type	Description
PCIE_ERR_REP	31:0	0x0	RW	Each bit corresponds to a particular error event as described in the section Proprietary Error Reporting.

11.2.2.2.49 PCIe Vendor ID - GLPCI_VENDORID (0x000BE518; RO)

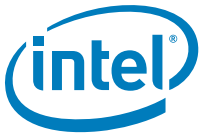
Field	Bit(s)	Init.	Type	Description
VENDORID	15:0	0x8086	RW	Contains the Vendor ID exposed in offset 0x0 in the config space of all functions. A value of 0xFFFF is ignored.
RESERVED	31:16	0x0	RSV	Reserved

11.2.2.3 PF - MAC Registers

11.2.2.3.1 PCS_XAUI_SWAP_A - PRTMAC_PCS_XAUI_SWAP_A (0x0008C480; RO)

Enables lane swapping, when working in XAUI mode. This register performs mapping between Group-A, on board, lanes 0-3 to internal PCS/MAC lanes.

Field	Bit(s)	Init.	Type	Description
SWAP_TX_LANE3	1:0	11b	RW	Select physical lane number to output MAC Lane3. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_TX_LANE2	3:2	10b	RW	Select physical lane number to output MAC Lane2. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_TX_LANE1	5:4	01b	RW	Select physical lane number to output MAC Lane1. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_TX_LANE0	7:6	00b	RW	Select physical lane number to output MAC Lane0. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_RX_LANE3	9:8	11b	RW	Select physical lane number from which to receive data into MAC Lane3. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_RX_LANE2	11:10	10b	RW	Select physical lane number from which to receive data into MAC Lane2. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_RX_LANE1	13:12	01b	RW	Select physical lane number from which to receive data into MAC Lane1. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_RX_LANE0	15:14	00b	RW	Select physical lane number from which to receive data into MAC Lane0. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
RESERVED	31:16	0x0	RSV	Reserved.



11.2.2.3.2 PCS_XAUI_SWAP_B - PRTMAC_PCS_XAUI_SWAP_B (0x0008C484; RO)

Enables lane swapping, when working in XAUI mode. This register performs mapping between Group-B, on board, lanes 0-3 to internal PCS/MAC lanes.

Field	Bit(s)	Init.	Type	Description
SWAP_TX_LANE3	1:0	11b	RW	Select physical lane number to output MAC Lane3. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_TX_LANE2	3:2	10b	RW	Select physical lane number to output MAC Lane2. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_TX_LANE1	5:4	01b	RW	Select physical lane number to output MAC Lane1. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_TX_LANE0	7:6	00b	RW	Select physical lane number to output MAC Lane0. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_RX_LANE3	9:8	11b	RW	Select physical lane number from which to receive data into MAC Lane3. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_RX_LANE2	11:10	10b	RW	Select physical lane number from which to receive data into MAC Lane2. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_RX_LANE1	13:12	01b	RW	Select physical lane number from which to receive data into MAC Lane1. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
SWAP_RX_LANE0	15:14	00b	RW	Select physical lane number from which to receive data into MAC Lane0. 00b = Lane0 01b = Lane1 10b = Lane2 11b = Lane3
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.3.3 Port MAC Address Low - PRTGL_SAL (0x001E2120; RO)

This register contains the lower bits of the NVM pre-loaded 48-bit per-PF Ethernet MAC Address. All 32 bits are valid.

Field	Bit(s)	Init.	Type	Description
FC_SAL	31:0	0x0	RW	Station address low used for flow control packet processing. The lower 32 bits of the 48-bit Ethernet MAC address. ***Note:*** This field is defined in Big Endian (LS byte of SAL is first on the wire).

11.2.2.3.4 Port MAC Address High - PRTGL_SAH (0x001E2140; RO)

This register contains the upper bits of the NVM pre-loaded 48-bit per-PF Ethernet MAC Address. The complete address is {PFPM_SAH, PFPM_SAL}. PFPM_SAH.*AV* determines whether this address is valid.

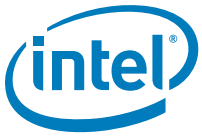
Field	Bit(s)	Init.	Type	Description
FC_SAH	15:0	0x0	RW	Station address high used for flow control packet processing. The upper 16 bits of the 48-bit Ethernet MAC address. ***Note:*** This field is defined in Big Endian (MS byte of SAH is last on the wire).
MFS	31:16	0x2600	RW	This field defines the maximum receive frame size in bytes from MAC addresses up to and inclusive of the CRC. Frames received that are larger than this value might be dropped based on the SBP configuration. ***Note:*** This configuration has no effect on maximum transmit frame size.

11.2.2.3.5 HSEC CONTROL Receive PFC ENABLE - PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE (0x001E30C0; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_RX_PAUSE_ENABLE	8:0	0x0	RW	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	Reserved.

11.2.2.3.6 HSEC CONTROL Transmit PAUSE_ENABLE - PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE (0x001E30D0; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_TX_PAUSE_ENABLE	8:0	0x0	RW	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	Reserved.



11.2.2.3.7 HSEC CONTROL Receive ENABLE_GCP - PRTMAC_HSEC_CTL_RX_ENABLE_GCP (0x001E30E0; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_RX_ENABLE_GCP	0	1b	RW	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	Reserved.

11.2.2.3.8 HSEC CONTROL Receive PAUSE_DA_UCAST_PART1 - PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1 (0x001E3110; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_RX_PAUSE_DA_UCAST_PART1	31:0	0x0	RW	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.

11.2.2.3.9 HSEC CONTROL Receive PAUSE_DA_UCAST_PART2 - PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2 (0x001E3120; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_RX_PAUSE_DA_UCAST_PART2	15:0	0x0	RW	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	Reserved.

11.2.2.3.10 HSEC CONTROL Receive PAUSE_SA_PART1 - PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART1 (0x001E3140; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_RX_PAUSE_SA_PART1	31:0	0x0	RW	Source address for pause processing.

11.2.2.3.11 HSEC CONTROL Receive PAUSE_SA_PART2 - PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART2 (0x001E3150; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_RX_PAUSE_SA_PART2	15:0	0x0	RW	Source address for pause processing.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.3.12 HSEC CONTROL Receive ENABLE_GPP - PRTMAC_HSEC_CTL_RX_ENABLE_GPP (0x001E3260; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_RX_ENABLE_GPP	0	0b	RW	A value of 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	Reserved.

11.2.2.3.13 HSEC CONTROL Receive ENABLE_PPP - PRTMAC_HSEC_CTL_RX_ENABLE_PPP (0x001E32E0; RO)

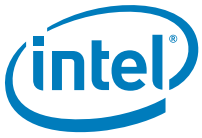
Field	Bit(s)	Init.	Type	Description
HSEC_CTL_RX_ENABLE_PPP	0	0b	RW	A value of 1b enables priority pause packet processing.
RESERVED	31:1	0x0	RSV	Reserved.

11.2.2.3.14 HSEC CONTROL Receive FORWARD_CONTROL - PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL (0x001E3360; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_RX_FORWARD_CONTROL	0	0b	RW	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	Reserved.

11.2.2.3.15 HSEC CONTROL Transmit PAUSE_QUANTA - PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n] (0x001E3370 + 0x10*n, n=0...8; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_TX_PAUSE_QUANTA	15:0	0xFFFF	RW	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	Reserved.



11.2.2.3.16 HSEC CONTROL Transmit PAUSE_REFRESH_TIMER - PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n] (0x001E3400 + 0x10*n, n=0...8; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_TX_PAUSE_REFRESH_TIMER	15:0	0x0	RW	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	Reserved.

11.2.2.3.17 HSEC CONTROL Transmit SA_GPP_PART1 - PRTMAC_HSEC_CTL_TX_SA_PART1 (0x001E34B0; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_TX_SA_PART1	31:0	0x0	RW	Source address for transmitting pause packets.

11.2.2.3.18 HSEC CONTROL Transmit SA_GPP_PART2 - PRTMAC_HSEC_CTL_TX_SA_PART2 (0x001E34C0; RO)

Field	Bit(s)	Init.	Type	Description
HSEC_CTL_TX_SA_PART2	15:0	0x0	RW	Source address for transmitting pause packets.
RESERVED	31:16	0x0	RSV	Reserved.

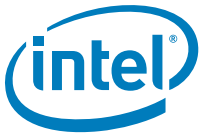
11.2.2.4 PF - Power Management Registers

11.2.2.4.1 General Control - PRTPM_GC (0x000B8140; RO)

Field	Bit(s)	Init.	Type	Description
EMP_LINK_ON	0	0b	RW	EMP Link On. 0b = This Ethernet port is not required for EMP functionality. 1b = This Ethernet port is required to be up for EMP functionality.
MNG_VETO	1	0b	RW	Manageability Veto for link LPLU and reset. Access read/write by manageability, read only to the host. Impact on LPLU: 0b = No specific constraints on link from manageability. 1b = Hold off any low-power link mode changes. This is done to avoid link loss and interrupting manageability activity. Impact on reset: Reset impact is global for the device and is received as a logical OR between the per port bits. When at least one of the per port bits is set, PCI reset triggers internal CORER which does not impact the MAC and PHY interfaces. When all bits are cleared, PCI reset triggers internal GLOBR which initialize also the MAC and PHY interfaces.
RATD	2	0b	RW	Restarts Auto Negotiation on Transition to Dx This bit enables the functionality to restart KX/KX4/KR Backplane Auto Negotiation on transitions to Dx(Dr/D3). 0b = Does not restart auto negotiation when all port's functions moves to the Dx state. 1b = Restarts auto negotiation to reach a low-power link mode (1G link) when all port's functions transitions to the Dx state.
LCDMP	3	0b	RW	Lowest Common Denominator (LCD) on Dx(Dr/D3) without main power. 0b = No specific action. 1b = Move to lowest common denominator link speed when main power is removed. When RATD bit is also set to 1b, it causes the link mode to auto negotiate to the lowest common denominator (if enabled) when the main-power (MAIN_PWR_OK) is removed.
RESERVED	30:4	0x0	RSV	Reserved.
LPLU_ASSERTED	31	0b	RW	LPLU asserted. This bit is set/cleared by FW according to the LPLU enable/disable state set by the Set PHY Config AQC received from the PF(s) attached to the port.

11.2.2.4.2 Energy Efficient Ethernet (EEE) STATUS - PRTPM_EEE_STAT (0x001E4320; RO)

Field	Bit(s)	Init.	Type	Description
RESERVED	28:0	0x0	RSV	Reserved. Write 0b. Ignore on read.
EEE_NEG	29	0b	RO	EEE support Negotiated on link. 0b = EEE operation not supported on link. 1b = EEE operation supported on link.
RX_LPI_STATUS	30	0b	RO	RX Link in LPI state. 0b = RX in Active state. 1b = RX in LPI state.
TX_LPI_STATUS	31	0b	RO	TX Link in LPI state. 0b = TX in Active state. 1b = TX in LPI state.



11.2.2.4.3 Energy Efficient Ethernet (EEE) Register - PRTPM_EEER (0x001E4360; RO)

Field	Bit(s)	Init.	Type	Description
TW_SYSTEM	15:0	0x0	RW	Time expressed in microseconds that no data is 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 SW 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	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, the device does 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, the device initiates entry into TX EEE LPI link state only if EEE support at the link speed was negotiated during Auto-negotiation or forced by SW to enable EEE on non AN protocols.
RESERVED	31:17	0x0	RSV	Reserved. Write 0. Ignore on read.

11.2.2.4.4 Energy Efficient Ethernet (EEE) Control - PRTPM_EEEC (0x001E4380; RO)

Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0x0	RSV	Reserved. Write 0. Ignore on read.
TW_WAKE_MIN	21:16	0xA	RW	Minimum time (expressed in 1 ms) 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, the device does not move out of TX LPI after timer has expired.
RESERVED	23:22	00b	RSV	Reserved.
TX_LU_LPI_DLY	25:24	11b	RW	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	0x2	RW	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, the device does not enter TX LPI and continue normal operation. 2. Minimum configuration should be 0x1.

11.2.2.4.5 EEE RX LPI Count - PRTPM_RLPIC (0x001E43A0; RO)

This register 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.

Field	Bit(s)	Init.	Type	Description
ERLPIC	31:0	0x0	RO	Number of EEE RX LPI events.

11.2.2.4.6 EEE TX LPI Count - PRTPM_TLPIC (0x001E43C0; RO)

This register 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.

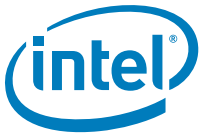
Field	Bit(s)	Init.	Type	Description
ETLPIC	31:0	0x0	RO	Number of EEE TX LPI events.

11.2.2.4.7 EEE TX Control - PRTPM_EEETXC (0x001E43E0; RO)

Field	Bit(s)	Init.	Type	Description
TW_PHY	15:0	0x1	RW	Tw_phy value is set by FW 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	Reserved

11.2.2.4.8 EEE TX Control - PRTPM_EEEFWD (0x001E4400; RO)

Field	Bit(s)	Init.	Type	Description
RESERVED	30:0	0x0	RSV	Reserved.
EEE_FW_CONFIG_DONE	31	0b	RW	Set by FW to indicate FW configuration of the EEE parameters after link establishment is done, cleared by HW when link is down



11.2.2.5 PF - Wake-Up and Proxying Registers

11.2.2.5.1 Flexible Host Filter Table Length - PFPM_FHFT_LENGTH[n] (0x0006A000 + 0x80*n, n=0...7; RW)

Field	Bit(s)	Init.	Type	Description
LENGTH	7:0	0x0	RW	This field contains the length of the filter defining the number of bytes from the beginning of the packet compared by this filter. If actual packet length is less than length specified by this field, the filter fails. Otherwise, it depends on the result of actual byte comparison. The value should not be greater than 128.
RESERVED	31:8	0x0	RSV	Reserved.

11.2.2.5.2 Wake Up Control Register - PFPM_WUC (0x0006B200; RW)

The *PME_En* and *PME_Status* bits of this register are reset when LAN_PWR_GOOD is 0b. When AUX_PWR = 0b, these register bits also reset by de-asserting PE_RST_N and during a D3 to D0 transition.

For each PF only the WUC register placed in the first WoL and Proxying register set is valid. The values programmed to the WUC registers in higher numbered WoL and Proxying register sets is ignored.

Field	Bit(s)	Init.	Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
EN_APM_D0	5	0b	RW	Enable APM wake also on D0. 0b = Enable wake only when function is in D3/Dr 1b = Enable wake also in D0
RESERVED	31:6	0x0	RSV	Reserved.

11.2.2.5.3 Wake Up Filter Control Register - PFPM_WUFC (0x0006B400; RW)

This register is used to enable each of the pre-defined and flexible 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.

Note: If the *NoTCO* bit is set, any packet that passes the manageability packet filtering does not cause a wake-up event.

Field	Bit(s)	Init.	Type	Description
LNKC	0	0b	RW	Link Status Change wake-up enable.
MAG	1	0b	RW	Magic Packet wake-up enable.
RESERVED	2	0b	RSV	Reserved.
MNG	3	0b	RW	MNG wake-up enable.
RESERVED	30:4	0x0	RSV	Reserved.
FW_RST_WK	31	0b	RW	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.

11.2.2.5.4 Wake Up Status Register - PFPM_WUS (0x0006B600; 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 RST# is asserted. It is only cleared when LAN_PWR_GOOD is deasserted, 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.

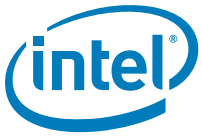
Field	Bit(s)	Init.	Type	Description
LNKC	0	0b	RW1C	Link Status Changed.
MAG	1	0b	RW1C	Magic Packet Received.
PME_STATUS	2	0b	RW1C	PME_Status. This bit is set when device receives a wake-up event. It is the same as the PME_Status bit in the Power Management Control / Status Register (PMCSR). Writing a 1b to this bit clears also 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	MNG wake-up status.
RESERVED	30:4	0x0	RSV	Reserved.
FW_RST_WK	31	0b	RW1C	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.

11.2.2.5.5 Flexible Host Filter Header Removal - PRTPM_FHFHR (0x0006C000; RO)

The Flexible Host Filter Header Removal (FHFHR) register defines the packet headers that need to be removed before doing a comparison via the Flex Filter Tables (FHFT registers).

Notes:
1. This register is reset by LAN_PWR_GOOD only.
2. Register is per-Port.
3. Register should be read by the PF driver prior to FHFT programing.

Field	Bit(s)	Init.	Type	Description
UNICAST	0	1b	RW	0b = Compare also Unicast-TAG information in Flex filters. 1b = Remove Unicast-TAG field before doing comparison in Flex filters.
MULTICAST	1	1b	RW	0b = Compare also Multicast-TAG information in Flex filters. 1b = Remove Multicast-TAG field before doing comparison in Flex filters.
RESERVED	31:2	0x0	RSV	Reserved.



11.2.2.5.6 WU on MNG Control - GLPM_WUMC (0x0006C800; RO)

Field	Bit(s)	Init.	Type	Description
NOTCO	0	0b	RW	Ignore management only packets for wake up. 0b = Ignore management only packets for wake up- a packet that meet the criteria defined in the MNGONLY register (i.e., are intended only for the BMC and not the Host) does not wake the host. 1b = Wake the host on any WUFC matched packet. This bit impacts the WU decision only when all PF functions are in D3 power state.
RESERVED	15:1	0x0	RSV	Reserved.
MNG_WU_PF	31:16	0x0	RW	*MNG_WU_PF* EMP can set a bit in this field to indicate MNG initiated WU event (bit per PF)

11.2.2.5.7 APM Control Register - PFPM_APM (0x000B8080; RW)

Field	Bit(s)	Init.	Type	Description
APME	0	0b	RW	Advance Power Management Enable. If set to 1b, APM Wakeup is enabled. ***Note:*** Bit is reset on Power on reset (LAN_PWR_GOOD) only.
RESERVED	31:1	0x0	RSV	Reserved.

11.2.2.5.8 MAC Address Low - PRTPM_SAL[n] (0x001E4440 + 0x20*n, n=0...3; RO)

Contains the NVM-loaded lower bits of the 48-bit Ethernet MAC Address. All 32 bits are valid.

Field	Bit(s)	Init.	Type	Description
PFPM_SAL	31:0	0x0	RW	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).

11.2.2.5.9 MAC Address High - PRTPM_SAH[n] (0x001E44C0 + 0x20*n, n=0...3; RO)

Contains the upper bits of the NVM pre-loaded 48-bit Ethernet MAC Address. The complete address is {PFPM_SAH, PFPM_SAL}.

Field	Bit(s)	Init.	Type	Description
PFPM_SAH	15:0	0x0	RW	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	Reserved.
PF_NUM	29:26	0x0	RW	PF number to be used for reporting the waking PF, value is written by FW.
MC_MAG_EN	30	0b	RW	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* or in PFPM_APM.*APME*.
AV	31	0b	RW	Address Valid. If the NVM is present, the Station Address is assigned by FW after loading from the NVM, and its Address Valid field is set to 1b.

11.2.2.6 PF - NVM Registers

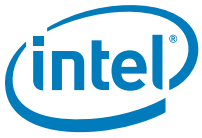
11.2.2.6.1 Unit Load Status - GLNVM_ULD (0x000B6008; RO)

This register provides indications on the completion of loading the Shadow RAM and Alternate Module into the device units.

Field	Bit(s)	Init.	Type	Description
CONF_PCIR_DONE	0	0b	RW	0b = Indicates that the units affected by the respective NVM module still need to be initialized. 1b = Indicates that the units affected by the module are ready.
CONF_PCIRTL_DONE	1	0b	RW	0b = Indicates that the units affected by the respective NVM module still need to be initialized. 1b = Indicates that the units affected by the module are ready.
CONF_LCB_DONE	2	0b	RW	0b = Indicates that the units affected by the respective NVM module still need to be initialized. 1b = Indicates that the units affected by the module are ready.
CONF_CORE_DONE	3	0b	RW	0b = Indicates that the units affected by the respective NVM module still need to be initialized. 1b = Indicates that the units affected by the module are ready.
CONF_GLOBAL_DONE	4	0b	RW	0b = Indicates that the units affected by the respective NVM module still need to be initialized. 1b = Indicates that the units affected by the module are ready.
CONF_POR_DONE	5	0b	RW	0b = Indicates that the units affected by the respective NVM module still need to be initialized. 1b = Indicates that the units affected by the module are ready.
CONF_PCIE_ANA_DONE	6	0b	RW	0b = Indicates that the units affected by the respective NVM module still need to be initialized. 1b = Indicates that the units affected by the module are ready.
CONF_PHY_ANA_DONE	7	0b	RW	0b = Indicates that the units affected by the respective NVM module still need to be initialized. 1b = Indicates that the units affected by the module are ready.
CONF_EMP_DONE	8	0b	RW	0b = Indicates that the units affected by the respective NVM module still need to be initialized. 1b = Indicates that the units affected by the module are ready.
CONF_PCIALT_DONE	9	0b	RW	0b = Indicates that the units affected by the respective NVM module still need to be initialized. 1b = Indicates that the units affected by the module are ready.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.6.2 Protected CSR List - GLNVM_PROTCSR[n] (0x000B6010 + 0x4*n, n=0...59; RO)

Field	Bit(s)	Init.	Type	Description
ADDR_BLOCK	23:0	0xFFFFFF	RW	CSR Blocked Address. 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	Reserved.



11.2.2.6.3 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	Description
NVM_PRES	0	0b	RO	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	Reserved.
SR_SIZE	7:5	110b	RO	Shadow RAM Size. Defines the size of the internal shadow RAM. This is equal to the size of the internal shadow RAM in power of 2 KB units. Initial value is 110b = 64 KB.
BANK1VAL	8	0b	RW	Basic Bank 1 Valid. 0b = Indicates that the content of basic banks 0 is valid. 1b = Indicates that the content of the basic bank 1 of the Flash device is valid. Meaningful only when *NVM_PRES* bit is read as 1b. It is written by hardware once at power-up, and then toggled only by EMP.
RESERVED	31:9	0x0	RSV	Reserved.

11.2.2.6.4 Flash ID Register - GLNVM_FLASHID (0x000B6104; RO)

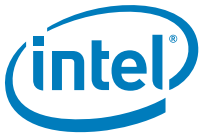
Field	Bit(s)	Init.	Type	Description
FLASHID	23:0	0x0	RO	Flash ID. It is formed by the 3-bytes JEDEC-ID of the device. Byte 0 = Bits[7:0] — The first byte read from the flash after the READ JEDEC-ID Opcode (0x9F) was issued to it. It contains the Manufacturer's ID. Byte 1 = Bits[15:8] — The second byte read from the flash after the READ JEDEC-ID Opcode (0x9F) was issued to it. It contains the Memory Type. Byte 2 = Bits[23:16] — The third byte read from the flash after the READ JEDEC-ID Opcode (0x9F) was issued to it. It contains the Memory Capacity.
RESERVED	30:24	0x0	RSV	Reserved.
FLEEP_PERF	31	0b	RW	FLEEP block Performance Enhancement. This bit enables 4-byte transaction to SPI flash when host reads from flash through the Expansion ROM BAR interface. 0b = Only single-byte transactions take place. 1b = 4-byte transactions are enabled. This bit can changed any time between transactions.

11.2.2.6.5 Flash Access Register - GLNVM_FLA (0x000B6108; RW)

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.

Field	Bit(s)	Init.	Type	Description
FL_SCK	0	0b	RW	Clock input to the Flash. When *FL_GNT* is set to 1b, the *FL_SCK* output signal is mapped to this bit and provides the serial clock input to the Flash. Software clocks the Flash via toggling this bit with successive writes. ***Note:*** This bit is operational to software only when in the blank flash programming mode. It is operational to EMP in any mode.

Field	Bit(s)	Init.	Type	Description
FL_CE	1	0b	RW	Chip select input to the Flash. When *FL_GNT* is set to 1b, the *FL_CE* output signal is mapped to the chip select of the Flash device. Software enables the Flash by writing a 0b to this bit. ***Note:*** This bit is operational to software only when in the blank flash programming mode. It is operational to EMP in any mode.
FL_SI	2	0b	RW	Data input to the Flash. When *FL_GNT* is set to 1b, the *FL_SI* output signal is mapped directly to this bit. Software provides data input to the Flash via writes to this bit. ***Note:*** This bit is operational to software only when in the blank flash programming mode. It is operational to EMP in any mode.
FL_SO	3	0b	RW	Data output bit from the Flash. The *FL_SO* output signal is mapped directly to this bit in the register and contains the Flash serial data output. This bit is read-only from a software perspective. Note that writes to this bit have no effect. ***Note:*** This bit is operational to software only when in the blank flash programming mode. It is operational to EMP in any mode.
FL_REQ	4	0b	RW	Request Flash Access. Software must write a 1b to this bit to get direct Flash access. It has access when *FL_GNT* is set to 1b. When software completes the access, it must then write a 0b. ***Note:*** This bit is operational to software only when in the blank flash programming mode. It is operational to EMP in any mode.
FL_GNT	5	0b	RO	Grant Flash Access. When this bit is set to 1b, software can access the Flash using the *FL_SCK*, *FL_CE*, *FL_SI*, and *FL_SO* bits. ***Note:*** This bit is operational to software only when in the blank flash programming mode. It is operational to EMP in any mode.
LOCKED	6	1b	RW	Normal NVM Programming Mode. When set to 1b, the device is in the normal NVM programming mode. When set to 0b, the device is in the blank flash programming mode. The bit can be cleared by EMP or by hardware.
RESERVED	17:7	0x0	RSV	Reserved
FL_SADDR	28:18	0x0	RW	Flash Sector Erase Address. Determines which 4 KB sector is erased when *FL_SER* command is used. This address is expressed in sector index units, starting from sector index 0. ***Note:*** This field is operational to software only when in the blank flash programming mode. It is operational to EMP in any mode.
FL_SER	29	0b	RSV	Flash Sector Erase Command. This bit is auto-cleared and reads as 0b. The 4 KB sector index to be erased is determined by *FL_SADDR* field. ***Note:*** This field is operational to software only when in the blank flash programming mode. It is operational to EMP in any mode.
FL_BUSY	30	0b	RO	Flash Busy. This bit is set to 1b while a write or an erase to the Flash is in progress. While this bit is cleared (reads as 0b), software/EMP can access to write a new byte to the Flash. ***Note:*** This bit is operational to software only when in the blank flash programming mode. It is operational to EMP in any mode.
FL_DER	31	0b	RW	Flash Device Erase Command. This bit is auto-cleared and reads as 0b. The entire Flash device is erased. ***Note:*** This field is operational to software only when in the blank flash programming mode. It is operational to EMP in any mode.



11.2.2.6.6 Shadow RAM Control Register - GLNVM_SRCTL (0x000B6110; RW)

This register cannot be loaded from NVM via one of the CSR format modules.

Field	Bit(s)	Init.	Type	Description
SRBUSY	0	0b	RO	Shadow RAM Busy. This bit indicates that the shadow RAM is busy and could not be accessed.
RESERVED	13:1	0x0	RSV	Reserved.
ADDR	28:14	0x0	RW	Address. This field is written by host along with *START* bit and the *WRITE* bit to indicate which shadow RAM word address to read or write.
WRITE	29	0b	RW	Write. This bit signals the shadow RAM if the current operation is read or write. 0b = Read 1b = Write
START	30	0b	RW	Start. Writing a 1b to this bit causes the read or write operation of the shadow RAM according to the write bit. This bit is self-cleared by the hardware.
DONE	31	1b	RW	Transaction Done. This bit is cleared after the *START* bit and *WRITE* bit are set by host, and is set back again when the shadow RAM write or read transaction completes.

11.2.2.6.7 Shadow RAM Read/Write Data - GLNVM_SRDATA (0x000B6114; RW)

Field	Bit(s)	Init.	Type	Description
WRDATA	15:0	0x0	RW	Write Data. Data to be written to the shadow RAM.
RDDATA	31:16	0x0	RO	Read Data. Data returned by the hardware from the shadow RAM read.

11.2.2.7 PF - Analyzer Registers

11.2.2.7.1 L2 Tag - Enable - PRT_L2TAGSEN (0x001C0B20; RW)

Field	Bit(s)	Init.	Type	Description
ENABLE	7:0	0x0	RW	Defines the L2 tags expected on this port.
RESERVED	31:8	0x0	RSV	Reserved.

11.2.2.8 PF - Switch Registers

11.2.2.8.1 Switching Table Default Action Enable Bitmap - GL_SWR_DEF_ACT_EN[n] (0x0026CFB8 + 0x4*n, n=0...1; RO)

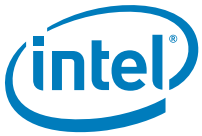
Switching table default action enable bitmap corresponding to GL_SWR_DEF_ACT.

Field	Bit(s)	Init.	Type	Description
DEF_ACT_EN_BITMAP	31:0	0x0	RW	Switching Table Default Action Enabling Bitmap.

11.2.2.8.2 Switching Table Default Action - GL_SWR_DEF_ACT[n] (0x00270200 + 0x4*n, n=0...35; RO)

For each switching table that does not hit for a packet, the default action takes place in case its corresponding enable bit is set in GL_SWR_DEF_ACT_EN register.

Field	Bit(s)	Init.	Type	Description
DEF_ACTION	31:0	0x0	RW	32-bit Switching Action per table.



11.2.2.9 PF - Interrupt Registers

11.2.2.9.1 VF Interrupt Throttling for Interrupt N - VFINT_ITRN[n,INTVF] (0x00020000 + 0x800*n + 0x4*INTVF, n=0...2, INTVF=0...511; RW)

Register index 'n' relates to interrupt 'n+1', while interrupt zero is controlled by the VFINT_ITR0 register.

Field	Bit(s)	Init.	Type	Description
INTERVAL	11:0	0x0	RW	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 also be programmed by setting the *INTERVAL* field in the matched xxINT_DYN_CTLx register.
RSVD	31:12	0x0	RSV	Reserved.

11.2.2.9.2 VF Interrupt N Dynamic Control - VFINT_DYN_CTLN[INTVF] (0x00024800 + 0x4*INTVF, INTVF=0...511; RW)

Register index 'n' relates to interrupt 'n+1', while interrupt zero is controlled by the VFINT_DYN_CTL0 register.

Field	Bit(s)	Init.	Type	Description
INTENA	0	0b	RW	Interrupt Enable. 0b = Interrupt disabled. 1b = Interrupt enabled. Refer to auto-clear policy in the “Interrupt Enablement” section. This bit is meaningful only if the *INTENA_MSK* flag in this register is not set.
CLEARPBA	1	0b	RW1C	Setting this bit clears the matched PBA bit. This bit is auto-cleared by hardware.
SWINT_TRIG	2	0b	RW1C	Trigger SW Interrupt. When this bit is set, a SW interrupt is triggered. This bit is auto-cleared by hardware.
ITR_INDX	4:3	00b	RW1C	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 hardware.
INTERVAL	16:5	0x0	RW1C	The interval for the ITR defined by the *ITR_INDX* field in this register. This field is auto-cleared by hardware.
RSVD	23:17	0x0	RSV	Reserved.
SW_ITR_INDX_ENA	24	0b	RW1C	Enables the programming of the *SW_ITR_INDX* field in this register. This flag is auto-cleared by hardware.

Field	Bit(s)	Init.	Type	Description
SW_ITR_INDX	26:25	00b	RW	ITR Index of the SW interrupt: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR When programming this field, the *SW_ITR_INDX_ENA* flag in this register should be set as well.
RSVD	30:27	0x0	RSV	Reserved.
INTENA_MSK	31	0b	RW1C	When this bit is set, the *INTENA* setting does not impact the device setting. This bit is auto-cleared by hardware.

11.2.2.9.3 Protected VF Interrupt N Linked List - VPINT_LNKLSTN[INTVF] (0x00025000 + 0x4*INTVF, INTVF=0...511; RW)

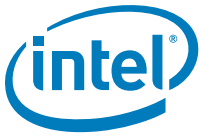
Register index 'n' relates to interrupt 'n+1', while interrupt zero is controlled by the VPINT_LNKLST0 register.

Field	Bit(s)	Init.	Type	Description
FIRSTQ_INDX	10:0	0x0	RW	First Queue Index in the MSI-X cause list. Transmit and receive queue indexes are within the PF space. CEQ indexes are within the function's space. Setting the index to 0x7FF points to an empty linked list (might be useful for “other cause” interrupt).
FIRSTQ_TYPE	12:11	00b	RW	First Queue Type. It can be one of the following: 00b = Receive Queues 01b = Transmit Queues 10b = PE Completion Event Queues 11b = Reserved
RSVD	31:13	0x0	RSV	Reserved.

11.2.2.9.4 Protected VF Interrupt N Rate Limit - VPINT_RATEN[INTVF] (0x00025800 + 0x4*INTVF, INTVF=0...511; RW)

These registers affect the VF but exposed only to the parent PF. Register index 'n' relates to interrupt 'n+1', while interrupt zero is controlled by the VFINT_RATE0 register.

Field	Bit(s)	Init.	Type	Description
INTERVAL	5:0	0x0	RW	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. For accurate rate limit, the *INTERVAL* must be smaller than 0x3C (up to 236 μs).
INTRL_ENA	6	0b	RW	Enable Interrupt Rate Limit on this interrupt vector.
RSVD	31:7	0x0	RSV	Reserved.



11.2.2.9.5 VF PE Completion Event Queue Interrupt Cause Control - VPINT_CEQCTL[INTVF] (0x00026800 + 0x4*INTVF, INTVF=0...511; RW)

Field	Bit(s)	Init.	Type	Description
MSIX_INDX	7:0	0x0	RW	MSI-X vector index within the function space. Software should set the *MSIX_INDX* field to values in the range of allocated interrupt vectors to the function.
RSVD	10:8	000b	RSV	Reserved.
ITR_INDX	12:11	00b	RW	ITR Index of the interrupt cause: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR
MSIX0_INDX	15:13	000b	RW	The index of the "QUEUE_x" bits in the ICR0 register ('x' = 0,...7), which is set as a result of an event on the queue. This field is relevant only if *MSIX_INDX* equals zero.
NEXTQ_INDX	26:16	0x0	RW	Next Queue Index in the MSI-X cause list. Transmit and receive queue indexes are within the PF space. CEQ indexes are within the function's space. Setting the index to 0x7FF is a NULL pointer indicating the end of the linked list.
NEXTQ_TYPE	28:27	00b	RW	Next Queue Type. It can be one of the following: 00b = Receive Queues 01b = Transmit Queues 10b = PE Completion Event Queues 11b = Reserved
RSVD	29	0b	RSV	Reserved.
CAUSE_ENA	30	0b	RW	Enable interrupt by this queue. When this bit is cleared, interrupts are not generated by the queue. The queue remains in the interrupt linked list and is processed at ITR expiration.
INTEVENT	31	0b	RO	Interrupt Event indication. Triggered by the specific event on the queue, and cleared when the interrupt is acknowledged by the interrupt signal logic

11.2.2.9.6 VF Interrupt Throttling for Interrupt Zero - VFINT_ITR0[n,VF] (0x00028000 + 0x400*n + 0x4*VF, n=0...2, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
INTERVAL	11:0	0x0	RW	ITR 'n' interval, where 'n' is the register index = 0,1,2 for the three ITRs per interrupt. 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 also be programmed by setting the *INTERVAL* field in the matched xxINT_DYN_CTLx register.
RSVD	31:12	0x0	RSV	Reserved.

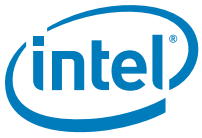
11.2.2.9.7 VF Interrupt Zero Static Control - VFINT_STAT_CTL0[VF] (0x0002A000 + 0x4*VF, VF=0...127; RW)

In case of MSI or Legacy INTA mode of operation, Interrupt zero is the only valid interrupt.

Field	Bit(s)	Init.	Type	Description
RSVD	1:0	00b	RSV	Reserved.
OTHER_ITR_INDX	3:2	00b	RW	ITR Index of the "other" interrupt causes: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR
RSVD	31:4	0x0	RSV	Reserved.

11.2.2.9.8 VF Interrupt Zero Dynamic Control - VFINT_DYN_CTL0[VF] (0x0002A400 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
INTENA	0	0b	RW	Interrupt Enable. 0b = Interrupt disabled. 1b = Interrupt enabled. Refer to the auto-clear policy in the "Interrupt Enablement" section. This bit is meaningful only if *INTENA_MSK* flag in this register is not set.
CLEARPBA	1	0b	RW1C	Setting this bit clears the matched PBA bit. This bit is auto-cleared by hardware.
SWINT_TRIG	2	0b	RW1C	Trigger SW Interrupt. When the bit is set, a SW interrupt is triggered. This bit is auto-cleared by hardware.
ITR_INDX	4:3	00b	RW1C	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 hardware:
INTERVAL	16:5	0x0	RW1C	The interval for the ITR defined by the *ITR_INDX* field in this register. This field is auto-cleared by hardware.
RSVD	23:17	0x0	RSV	Reserved.
SW_ITR_INDX_ENA	24	0b	RW1C	Enables the programming of the *SW_ITR_INDX* field in this register. This flag is auto-cleared by hardware.
SW_ITR_INDX	26:25	00b	RW	ITR Index of the SW interrupt: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR When programming this field, the *SW_ITR_INDX_ENA* flag in this register should be set as well.
RSVD	30:27	0x0	RSV	Reserved.
INTENA_MSK	31	0b	RW1C	When this bit is set, the *INTENA* setting does not impact the device setting. This bit is auto-cleared by hardware.



11.2.2.9.9 Protected VF Interrupt Zero Linked List - VPINT_LNKLST0[VF] (0x0002A800 + 0x4*VF, VF=0...127; RW)

In case of MSI or Legacy INTA mode of operation, Interrupt zero is the only valid interrupt.

Field	Bit(s)	Init.	Type	Description
FIRSTQ_INDX	10:0	0x0	RW	First Queue Index in the MSI-X cause list. Transmit and receive queue indexes are within the PF space. CEQ indexes are within the function's space. Setting the index to 0x7FF points to an empty linked list (might be useful for “other cause” interrupt).
FIRSTQ_TYPE	12:11	00b	RW	First Queue Type. It can be one of the following: 00b = Receive Queues 01b = Transmit Queues 10b = PE Completion Event Queues 11b = Reserved
RSVD	31:13	0x0	RSV	Reserved.

11.2.2.9.10 Protected VF Interrupt Zero Rate Limit - VPINT_RATE0[VF] (0x0002AC00 + 0x4*VF, VF=0...127; RW)

This register affects the VF, but is exposed only to the parent PF.

Field	Bit(s)	Init.	Type	Description
INTERVAL	5:0	0x0	RW	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. For accurate rate limit, the *INTERVAL* must be smaller than 0x3C (up to 236 μs).
INTRL_ENA	6	0b	RW	Enable Interrupt Rate Limit on this interrupt vector.
RSVD	31:7	0x0	RSV	Reserved.

11.2.2.9.11 VF PE Asynchronous Event Queue Interrupt Cause Control - VPINT_AEQCTL[VF] (0x0002B800 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
MSIX_INDX	7:0	0x0	RW	MSI-X vector index within the function space. Software should set the *MSIX_INDX* field to values in the range of allocated interrupt vectors to the function.
RSVD	10:8	000b	RSV	Reserved.
ITR_INDX	12:11	00b	RW	ITR Index of the interrupt cause: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR
MSIX0_INDX	15:13	000b	RW	The index of the “QUEUE_x” bits in the ICR0 register ('x' = 0,...7), which is set as a result of an event on the queue. This field is relevant only if *MSIX_INDX* equals zero.
RSVD	29:16	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
CAUSE_ENA	30	0b	RW	Enable interrupt by this queue. When this bit is cleared, interrupts are not generated by the queue. The queue remains in the interrupt linked list and is processed at ITR expiration
RSVD	31	0b	RSV	Reserved.

11.2.2.9.12 VF Interrupt Zero Cause - VFINT_ICR0[VF] (0x0002BC00 + 0x4*VF, VF=0...127; RCW)

Field	Bit(s)	Init.	Type	Description
INTEVENT	0	0b	RCW	Interrupt Event indication. This bit is set on assertion of any causes for this interrupt, and cleared when the interrupt is asserted to the Rate Limit logic.
QUEUE_0	1	0b	RCW	Queue 0 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_1	2	0b	RCW	Queue 1 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_2	3	0b	RCW	Queue 2 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_3	4	0b	RCW	Queue 3 interrupt for LAN Transmit and Receive queues and PE CEQs.
RSVD	29:5	0x0	RSV	Reserved.
ADMINQ	30	0b	RCW	Send/Receive Admin queue interrupt indication.
SWINT	31	0b	RCW	Software Interrupt indication.

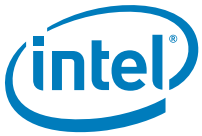
11.2.2.9.13 VF Interrupt Zero Cause Enablement - VFINT_ICR0_ENA[VF] (0x0002C000 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
RSVD	29:0	0x0	RSV	Reserved.
ADMINQ	30	0b	RW	Enable this interrupt at '1'.
RSVD	31	0b	RW	Reserved.

11.2.2.9.14 PF Interrupt Throttling for Interrupt N - PFINT_ITRN[n,INTPF] (0x00030000 + 0x800*n + 0x4*INTPF, n=0...2, INTPF=0...511; RW)

Register index 'n' relates to interrupt 'n+1', while interrupt zero is controlled by the PFINT_ITR0 register. In case of MSI or Legacy INTA, only interrupt zero is valid, so none of these registers impact the device functionality.

Field	Bit(s)	Init.	Type	Description
INTERVAL	11:0	0x0	RW	ITR 'n' interval, where 'n' is the register index = 0,1,2 for the three ITRs per interrupt. 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 also be programmed by setting the *INTERVAL* field in the matched xxINT_DYN_CTLx register.
RSVD	31:12	0x0	RSV	Reserved.



11.2.2.9.15 PF Interrupt N Dynamic Control - PFINT_DYN_CTLN[INTPF] (0x00034800 + 0x4*INTPF, INTPF=0...511; RW)

Register index 'n' relates to interrupt 'n+1', while interrupt zero is controlled by the PFINT_DYN_CTL0 register. In case of MSI or Legacy INTA, only interrupt zero is valid, so none of these registers impact the device functionality.

Field	Bit(s)	Init.	Type	Description
INTENA	0	0b	RW	Interrupt Enable. 0b = Interrupt disabled. 1b = Interrupt enabled. Refer to the auto-clear policy in the "Interrupt Enablement" section. This bit is meaningful only if the *INTENA_MSK* flag in this register is not set.
CLEARPBA	1	0b	RW1C	Setting this bit clears the matched PBA bit. This bit is auto-cleared by hardware.
SWINT_TRIG	2	0b	RW1C	Trigger SW Interrupt. When this bit is set, a SW interrupt is triggered. This bit is auto-cleared by hardware.
ITR_INDX	4:3	00b	RW1C	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 hardware.
INTERVAL	16:5	0x0	RW1C	The interval for the ITR defined by the *ITR_INDX* field in this register. This field is auto-cleared by hardware.
RSVD	23:17	0x0	RSV	Reserved.
SW_ITR_INDX_ENA	24	0b	RW1C	Enables the programming of the *SW_ITR_INDX* field in this register. This flag is auto-cleared by hardware.
SW_ITR_INDX	26:25	00b	RW	ITR Index of the SW interrupt: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR When programming this field, the *SW_ITR_INDX_ENA* flag in this register should be set as well.
RSVD	30:27	0x0	RSV	Reserved
INTENA_MSK	31	0b	RW1C	When the *INTENA_MSK* bit is set, the *INTENA* setting does not impact the device setting. This bit is auto-cleared by hardware.

11.2.2.9.16 PF Interrupt N Linked List - PFINT_LNKLSTN[INTPF] (0x00035000 + 0x4*INTPF, INTPF=0...511; RW)

Register index 'n' relates to interrupt 'n+1', while interrupt zero is controlled by the PFINT_LNKLST0 register. In case of MSI or Legacy INTA, only interrupt zero is valid, so none of these registers impact the device functionality.

Field	Bit(s)	Init.	Type	Description
FIRSTQ_INDX	10:0	0x0	RW	First Queue Index in the MSI-X cause list. Transmit and receive queue indexes are within the PF space. CEQ indexes are within the function's space. Setting the index to 0x7FF points to an empty linked list (might be useful for “other cause” interrupt).
FIRSTQ_TYPE	12:11	00b	RW	First Queue Type. It can be one of the following: 00b = Receive Queues 01b = Transmit Queues 10b = PE Completion Event Queues 11b = Reserved
RSVD	31:13	0x0	RSV	Reserved.

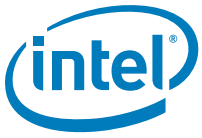
11.2.2.9.17 PF Interrupt N Rate Limit - PFINT_RATEN[INTPF] (0x00035800 + 0x4*INTPF, INTPF=0...511; RW)

Register index 'n' relates to interrupt 'n+1', while interrupt zero is controlled by the PFINT_RATE0 register. In case of MSI or Legacy INTA, only interrupt zero is valid, so none of these registers impact the device functionality.

Field	Bit(s)	Init.	Type	Description
INTERVAL	5:0	0x0	RW	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. For accurate rate limit, the *INTERVAL* must be smaller than 0x3C (up to 236 μs).
INTRL_ENA	6	0b	RW	Enable Interrupt Rate Limit on this interrupt vector.
RSVD	31:7	0x0	RSV	Reserved.

11.2.2.9.18 PF PE Completion Event Queue Interrupt Cause Control - PFINT_CEQCTL[INTPF] (0x00036800 + 0x4*INTPF, INTPF=0...511; RW)

Field	Bit(s)	Init.	Type	Description
MSIX_INDX	7:0	0x0	RW	MSI-X vector index within the function space. Software should set the *MSIX_INDX* field to values in the range of allocated interrupt vectors to the function.
RSVD	10:8	000b	RSV	Reserved.
ITR_INDX	12:11	00b	RW	ITR Index of the interrupt cause: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR



Field	Bit(s)	Init.	Type	Description
MSIX0_INDX	15:13	000b	RW	The index of the "QUEUE_x" bits in the ICR0 register ('x' = 0,...7), which is set as a result of an event on the queue. This field is relevant only if *MSIX_INDX* equals zero.
NEXTQ_INDX	26:16	0x0	RW	Next Queue Index in the MSI-X cause list. Transmit and receive queue indexes are within the PF space. CEQ indexes are within the function's space. Setting the index to 0x7FF is a NULL pointer indicating the end of the linked list.
NEXTQ_TYPE	28:27	00b	RW	Next Queue Type. It can be one of the following: 00b = Receive Queues 01b = Transmit Queues 10b = PE Completion Event Queues 11b = Reserved
RSVD	29	0b	RSV	Reserved.
CAUSE_ENA	30	0b	RW	Enable interrupt by this queue. When this bit is cleared, interrupts are not generated by the queue. The queue remains in the interrupt linked list and is processed at ITR expiration.
INTEVENT	31	0b	RO	Interrupt Event indication. Triggered by the specific event on the queue, and cleared when the interrupt is acknowledged by the interrupt signal logic

11.2.2.9.19 PF Interrupt Throttling for Interrupt Zero - PFINT_ITR0[n] (0x00038000 + 0x80*n, n=0...2; RW)

Field	Bit(s)	Init.	Type	Description
INTERVAL	11:0	0x0	RW	ITR 'n' interval, where 'n' is the register index = 0,1,2 for the three ITRs per interrupt. 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 also be programmed also by setting the *INTERVAL* field in the matched xxINT_DYN_CTLx register.
RSVD	31:12	0x0	RSV	Reserved.

11.2.2.9.20 PF Interrupt Zero Static Control - PFINT_STAT_CTL0 (0x00038400; RW)

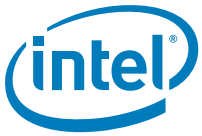
In case of MSI or Legacy INTA mode of operation, interrupt zero is the only valid interrupt.

Field	Bit(s)	Init.	Type	Description
RSVD	1:0	00b	RSV	Reserved
OTHER_ITR_INDX	3:2	00b	RW	ITR Index of the "other" interrupt causes: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR
RSVD	31:4	0x0	RSV	Reserved.

11.2.2.9.21 PF Interrupt Zero Dynamic Control - PFINT_DYN_CTL0 (0x00038480; RW)

In case of MSI or Legacy INTA mode of operation, interrupt zero is the only valid interrupt.

Field	Bit(s)	Init.	Type	Description
INTENA	0	0b	RW	Interrupt Enable. 0b = Interrupt disabled. 1b = Interrupt enabled. Refer to the auto-clear policy in the "Interrupt Enablement" section. This bit is meaningful only if the *INTENA_MSK* flag in this register is not set.
CLEARPBA	1	0b	RW1C	Setting this bit clears the matched PBA bit. This bit is auto-cleared by hardware.
SWINT_TRIG	2	0b	RW1C	Trigger SW Interrupt. When this bit is set, a SW interrupt is triggered. This bit is auto-cleared by hardware.
ITR_INDX	4:3	00b	RW1C	Defines the ITR Index to be updated, as follows: 00b = ITR0 01b = ITR1 10b = ITR 11b = No ITR Update This field is auto-cleared by hardware:
INTERVAL	16:5	0x0	RW1C	The interval for the ITR defined by the *ITR_INDX* field in this register. This field is auto-cleared by hardware.
RSVD	23:17	0x0	RSV	Reserved.
SW_ITR_INDX_ENA	24	0b	RW1C	Enables the programming of the *SW_ITR_INDX* field in this register. This flag is auto-cleared by hardware.
SW_ITR_INDX	26:25	00b	RW	ITR Index of the SW interrupt: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR When programming this field, the *SW_ITR_INDX_ENA* flag in this register should be set as well.
RSVD	30:27	0x0	RSV	Reserved.
INTENA_MSK	31	0b	RW1C	When the *INTENA_MSK* bit is set, the *INTENA* setting does not impact the device setting. This bit is auto-cleared by hardware.



11.2.2.9.22 PF Interrupt Zero Linked List - PFINT_LNKLST0 (0x00038500; RW)

In case of MSI or Legacy INTA mode of operation, interrupt zero is the only valid interrupt.

Field	Bit(s)	Init.	Type	Description
FIRSTQ_INDX	10:0	0x0	RW	First Queue Index in the MSI-X cause list. Transmit and receive queue indexes are within the PF space. CEQ indexes are within the function's space. Setting the index to 0x7FF points to an empty linked list (might be useful for “other cause” interrupt).
FIRSTQ_TYPE	12:11	00b	RW	First Queue Type. It can be one of the following: 00b = Receive Queues 01b = Transmit Queues 10b = PE Completion Event Queues 11b = Reserved
RSVD	31:13	0x0	RSV	Reserved.

11.2.2.9.23 PF Interrupt Zero Rate Limit - PFINT_RATE0 (0x00038580; RW)

Field	Bit(s)	Init.	Type	Description
INTERVAL	5:0	0x0	RW	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. For accurate rate limit, the *INTERVAL* must be smaller than 0x3C (up to 236 μs).
INTRL_ENA	6	0b	RW	Enable Interrupt Rate Limit on this interrupt vector.
RSVD	31:7	0x0	RSV	Reserved.

11.2.2.9.24 PF PE Asynchronous Event Queue Interrupt Cause Control - PFINT_AEQCTL (0x00038700; RW)

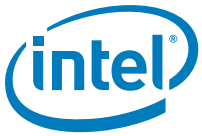
Field	Bit(s)	Init.	Type	Description
MSIX_INDX	7:0	0x0	RW	MSI-X vector index within the function space. Software should set the *MSIX_INDX* field to values in the range of allocated interrupt vectors to the function.
RSVD	10:8	000b	RSV	Reserved.
ITR_INDX	12:11	00b	RW	ITR Index of the interrupt cause: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR
MSIX0_INDX	15:13	000b	RW	The index of the “QUEUE_x” bits in the ICR0 register ('x' = 0,...7), which is set as a result of an event on the queue. This field is relevant only if *MSIX_INDX* equals zero.
RSVD	29:16	0x0	RSV	Reserved.
CAUSE_ENA	30	0b	RW	Enable interrupt by this queue. When this bit is cleared, interrupts are not generated by the queue. The queue remains in the interrupt linked list and is processed at ITR expiration
RSVD	31	0b	RSV	Reserved.

11.2.2.9.25 PF Interrupt Zero Cause - PFINT_ICR0 (0x00038780; RCW)

Field	Bit(s)	Init.	Type	Description
INTEVENT	0	0b	RCW	Interrupt Event indication. This bit is set on assertion of any causes for this interrupt, and cleared when the interrupt is asserted to the Rate Limit logic.
QUEUE_0	1	0b	RCW	Queue 0 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_1	2	0b	RCW	Queue 1 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_2	3	0b	RCW	Queue 2 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_3	4	0b	RCW	Queue 3 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_4	5	0b	RCW	Queue 4 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_5	6	0b	RCW	Queue 5 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_6	7	0b	RCW	Queue 6 i.nterrupt for LAN Transmit and Receive queues and PE CEQs
QUEUE_7	8	0b	RCW	Queue 7 interrupt for LAN Transmit and Receive queues and PE CEQs.
RSVD	15:9	0x0	RSV	Reserved.
ECC_ERR	16	0b	RCW	Unrecoverable ECC Error. This bit is set when an unrecoverable error is detected in one of the device memories.
RSVD	18:17	00b	RSV	Reserved.
MAL_DETECT	19	0b	RCW	Malicious programming detected.
GRST	20	0b	RCW	Global Resets Requested (CORER, GLOBR or EMPR).
PCI_EXCEPTION	21	0b	RCW	The PCI exception is activated by one of the events described in section “3.1.5.7 Proprietary Error Reporting”. The events are captured in the PFPCI_ICAUSE register.
GPIO	22	0b	RCW	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.
TIMESYNC	23	0b	RCW	Any of the TimeSync interrupt causes as described in the interrupts section of the “TimeSync (IEEE1588 and 802.1AS)” section.
RESERVED	25:24	00b	RSV	Reserved.
HMC_ERR	26	0b	RCW	HMC error as indicated in the PFHMC_ERRORINFO and PFHMC_ERRORDATA registers.
RSVD	28:27	00b	RSV	Reserved
VFLR	29	0b	RCW	VFLR was initiated by one of the VFs of the PF. The PF should read the GLGEN_VFLRSTAT for an indication for the VF that generated the VFLR.
ADMINQ	30	0b	RCW	Send/Receive Admin queue interrupt indication.
SWINT	31	0b	RCW	Software Interrupt indication.

11.2.2.9.26 PF Interrupt Zero Cause Enablement - PFINT_ICR0_ENA (0x00038800; RW)

Field	Bit(s)	Init.	Type	Description
RSVD	15:0	0x0	RSV	Reserved.
ECC_ERR	16	0b	RW	Enable this interrupt at 1b.



Field	Bit(s)	Init.	Type	Description
RSVD	18:17	00b	RSV	Reserved.
MAL_DETECT	19	0b	RW	Enable this interrupt at 1b.
GRST	20	0b	RW	Enable this interrupt at 1b.
PCI_EXCEPTION	21	0b	RW	Enable this interrupt at 1b.
GPIO	22	0b	RW	Enable this interrupt at 1b.
TIMESYNC	23	0b	RW	Enable this interrupt at 1b.
RESERVED	25:24	00b	RSV	Reserved.
HMC_ERR	26	0b	RW	Enable this interrupt at 1b.
RSVD	28:27	00b	RSV	Reserved.
VFLR	29	0b	RW	Enable this interrupt at 1b.
ADMINQ	30	0b	RW	Enable this interrupt at 1b.
RSVD	31	0b	RW	Reserved.

11.2.2.9.27 Receive Queue Interrupt Cause Control - QINT_RQCTL[Q] (0x0003A000 + 0x4*Q, Q=0...1535; RW)

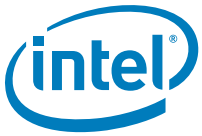
Field	Bit(s)	Init.	Type	Description
MSIX_INDX	7:0	0x0	RW	MSI-X vector index within the function space. Software should set the *MSIX_INDX* field to values in the range of allocated interrupt vectors to the function.
RSVD	10:8	000b	RSV	Reserved
ITR_INDX	12:11	00b	RW	ITR Index of the interrupt cause: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR
MSIX0_INDX	15:13	000b	RW	The index of the “QUEUE_x” bits in the ICR0 register ('x' = 0,...7)s which is set as a result of an event on the queue. This field is relevant only if *MSIX_INDX* equals zero.
NEXTQ_INDX	26:16	0x0	RW	Next Queue Index in the MSI-X cause list. Transmit and receive queue indexes are within the PF space. CEQ indexes are within the function's space. Setting the index to 0x7FF is a NULL pointer indicating the end of the linked list.
NEXTQ_TYPE	28:27	00b	RW	Next Queue Type. It can be one of the following: 00b = Receive Queues 01b = Transmit Queues 10b = PE Completion Event Queues 11b = Reserved
RSVD	29	0b	RSV	Reserved.
CAUSE_ENA	30	0b	RW	Enable interrupt by this queue. When this bit is cleared, interrupts are not generated by the queue. The queue remains in the interrupt linked list and is processed at ITR expiration
INTEVENT	31	0b	RO	Interrupt Event indication. Triggered by the specific event on the queue, and cleared when the interrupt is acknowledged by the interrupt signal logic

11.2.2.9.28 Transmit Queue Interrupt Cause Control - QINT_TQCTL[Q] (0x0003C000 + 0x4*Q, Q=0...1535; RW)

Field	Bit(s)	Init.	Type	Description
MSIX_INDX	7:0	0x0	RW	MSI-X vector index within the function space. Software should set the *MSIX_INDX* field to values in the range of allocated interrupt vectors to the function.
RSVD	10:8	000b	RSV	Reserved.
ITR_INDX	12:11	00b	RW	ITR Index of the interrupt cause: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR
MSIX0_INDX	15:13	000b	RW	The index of the “QUEUE_x” bits in the ICR0 register ('x' = 0,...7), which is set as a result of an event on the queue. This field is relevant only if *MSIX_INDX* equals zero.
NEXTQ_INDX	26:16	0x0	RW	Next Queue Index in the MSI-X cause list. Transmit and receive queue indexes are within the PF space. CEQ indexes are within the function's space. Setting the index to 0x7FF is a NULL pointer indicating the end of the linked list.
NEXTQ_TYPE	28:27	00b	RW	Next Queue Type. It can be one of the following: 00b = Receive Queues 01b = Transmit Queues 10b = PE Completion Event Queues 11b = Reserved
RSVD	29	0b	RSV	Reserved.
CAUSE_ENA	30	0b	RW	Enable interrupt by this queue. When this bit is cleared, interrupts are not generated by the queue. The queue remains in the interrupt linked list and is processed at ITR expiration
INTEVENT	31	0b	RO	Interrupt Event indication. Triggered by the specific event on the queue, and cleared when the interrupt is acknowledged by the interrupt signal logic

11.2.2.9.29 LAN Port MDIO Number - PFGEN_PORTMDIO_NUM (0x0003F100; RO)

Field	Bit(s)	Init.	Type	Description
PORT_NUM	1:0	00b	RW	Port Number. Indicates the LAN port connected to this function. 00b = Port 0 01b = Port 1 10b = Port 2 11b = Port 3 This field must be identical to the PFGEN_PORTNUM register.
RSVD	31:2	0x0	RSV	Reserved.

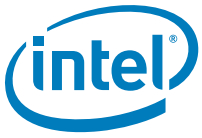


11.2.2.9.30 PF General Purpose IO Interrupt Enablement - PFINT_GPIO_ENA (0x00088080; RW)

Field	Bit(s)	Init.	Type	Description
GPIO0_ENA	0	0b	RW	Enable interrupt on GPIO 0.
GPIO1_ENA	1	0b	RW	Enable interrupt on GPIO 1.
GPIO2_ENA	2	0b	RW	Enable interrupt on GPIO 2.
GPIO3_ENA	3	0b	RW	Enable interrupt on GPIO 3.
GPIO4_ENA	4	0b	RW	Enable interrupt on GPIO 4.
GPIO5_ENA	5	0b	RW	Enable interrupt on GPIO 5.
GPIO6_ENA	6	0b	RW	Enable interrupt on GPIO 6.
GPIO7_ENA	7	0b	RW	Enable interrupt on GPIO 7.
GPIO8_ENA	8	0b	RW	Enable interrupt on GPIO 8.
GPIO9_ENA	9	0b	RW	Enable interrupt on GPIO 9.
GPIO10_ENA	10	0b	RW	Enable interrupt on GPIO 10.
GPIO11_ENA	11	0b	RW	Enable interrupt on GPIO 11.
GPIO12_ENA	12	0b	RW	Enable interrupt on GPIO 12.
GPIO13_ENA	13	0b	RW	Enable interrupt on GPIO 13.
GPIO14_ENA	14	0b	RW	Enable interrupt on GPIO 14.
GPIO15_ENA	15	0b	RW	Enable interrupt on GPIO 15.
GPIO16_ENA	16	0b	RW	Enable interrupt on GPIO 16.
GPIO17_ENA	17	0b	RW	Enable interrupt on GPIO 17.
GPIO18_ENA	18	0b	RW	Enable interrupt on GPIO 18.
GPIO19_ENA	19	0b	RW	Enable interrupt on GPIO 19.
GPIO20_ENA	20	0b	RW	Enable interrupt on GPIO 20.
GPIO21_ENA	21	0b	RW	Enable interrupt on GPIO 21.
GPIO22_ENA	22	0b	RW	Enable interrupt on GPIO 22.
GPIO23_ENA	23	0b	RW	Enable interrupt on GPIO 23.
GPIO24_ENA	24	0b	RW	Enable interrupt on GPIO 24.
GPIO25_ENA	25	0b	RW	Enable interrupt on GPIO 25.
GPIO26_ENA	26	0b	RW	Enable interrupt on GPIO 26.
GPIO27_ENA	27	0b	RW	Enable interrupt on GPIO 27.
GPIO28_ENA	28	0b	RW	Enable interrupt on GPIO 28.
GPIO29_ENA	29	0b	RW	Enable interrupt on GPIO 29.
RSVD	31:30	00b	RSV	Reserved.

11.2.2.9.31 EMP General Purpose IO Interrupt Enablement - EMPINT_GPIO_ENA (0x00088188; RO)

Field	Bit(s)	Init.	Type	Description
GPIO0_ENA	0	0b	RW	Enable interrupt on GPIO 0.
GPIO1_ENA	1	0b	RW	Enable interrupt on GPIO 1.
GPIO2_ENA	2	0b	RW	Enable interrupt on GPIO 2.
GPIO3_ENA	3	0b	RW	Enable interrupt on GPIO 3.
GPIO4_ENA	4	0b	RW	Enable interrupt on GPIO 4.
GPIO5_ENA	5	0b	RW	Enable interrupt on GPIO 5.
GPIO6_ENA	6	0b	RW	Enable interrupt on GPIO 6.
GPIO7_ENA	7	0b	RW	Enable interrupt on GPIO 7.
GPIO8_ENA	8	0b	RW	Enable interrupt on GPIO 8.
GPIO9_ENA	9	0b	RW	Enable interrupt on GPIO 9.
GPIO10_ENA	10	0b	RW	Enable interrupt on GPIO 10.
GPIO11_ENA	11	0b	RW	Enable interrupt on GPIO 11.
GPIO12_ENA	12	0b	RW	Enable interrupt on GPIO 12.
GPIO13_ENA	13	0b	RW	Enable interrupt on GPIO 13.
GPIO14_ENA	14	0b	RW	Enable interrupt on GPIO 14.
GPIO15_ENA	15	0b	RW	Enable interrupt on GPIO 15.
GPIO16_ENA	16	0b	RW	Enable interrupt on GPIO 16.
GPIO17_ENA	17	0b	RW	Enable interrupt on GPIO 17.
GPIO18_ENA	18	0b	RW	Enable interrupt on GPIO 18.
GPIO19_ENA	19	0b	RW	Enable interrupt on GPIO 19.
GPIO20_ENA	20	0b	RW	Enable interrupt on GPIO 20.
GPIO21_ENA	21	0b	RW	Enable interrupt on GPIO 21.
GPIO22_ENA	22	0b	RW	Enable interrupt on GPIO 22.
GPIO23_ENA	23	0b	RW	Enable interrupt on GPIO 23.
GPIO24_ENA	24	0b	RW	Enable interrupt on GPIO 24.
GPIO25_ENA	25	0b	RW	Enable interrupt on GPIO 25.
GPIO26_ENA	26	0b	RW	Enable interrupt on GPIO 26.
GPIO27_ENA	27	0b	RW	Enable interrupt on GPIO 27.
GPIO28_ENA	28	0b	RW	Enable interrupt on GPIO 28.
GPIO29_ENA	29	0b	RW	Enable interrupt on GPIO 29.
RSVD	31:30	00b	RSV	Reserved.



11.2.2.10 PF - Virtualization PF Registers

11.2.2.10.1 Malicious Driver Detected on TX - VP_MDET_TX[VF] (0x000E6000 + 0x4*VF, VF=0...127; RW1C)

This register records a malicious event detected on the TX queues. Once read, driver must write 0xFFFF to clear.

Field	Bit(s)	Init.	Type	Description
VALID	0	0b	RW1C	A malicious event has been detected on this function.
RESERVED	31:1	0x0	RSV	Reserved.

11.2.2.10.2 Malicious Driver Detected on TX - PF_MDET_TX (0x000E6400; RW1C)

This register records a malicious event detected on the TX queues. Once read, driver must write 0xFFFF to clear.

Field	Bit(s)	Init.	Type	Description
VALID	0	0b	RW1C	A malicious event has been detected on this function.
RESERVED	31:1	0x0	RSV	Reserved.

11.2.2.10.3 Malicious Driver TX Event Details - GL_MDET_TX (0x000E6480; RW1C)

This register records the details of the first TX event detected.

Field	Bit(s)	Init.	Type	Description
QUEUE	11:0	0x0	RW1C	Absolute queue ID on which the event was detected.
VF_NUM	20:12	0x0	RW1C	Absolute VF number on which the event was detected.
PF_NUM	24:21	0x0	RW1C	PF / parent PF number on which the event was detected.
EVENT	29:35	0x0	RW1C	ID of the event that has been recorded see “Malicious Driver - TX descriptor checks” table.
RSVD	30	0b	RSV	Reserved.
VALID	31	0b	RW1C	Indicates that an event has been captured.

11.2.2.10.4 Malicious Driver Detected on RX - VP_MDET_RX[VF] (0x0012A000 + 0x4*VF, VF=0...127; 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	Description
VALID	0	0b	RW1C	A malicious event has been detected on this function.
RESERVED	31:1	0x0	RSV	Reserved.

11.2.2.10.5 Malicious Driver Detected on RX - PF_MDET_RX (0x0012A400; 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	Description
VALID	0	0b	RW1C	A malicious event has been detected on this function.
RESERVED	31:1	0x0	RSV	Reserved.

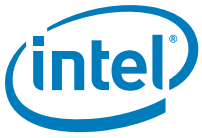
11.2.2.10.6 Malicious Driver RX Event Details - GL_MDET_RX (0x0012A510; RW1C)

This register records the details of the first RX event detected.

Field	Bit(s)	Init.	Type	Description
FUNCTION	7:0	0x0	RW1C	The function that triggered the event.
EVENT	16:8	0x0	RW1C	ID of the event that has been recorded.
QUEUE	30:17	0x0	RW1C	Queue Id on which the event was detected.
VALID	31	0b	RW1C	Indicates that an event has been captured.

11.2.2.10.7 PF Resources Allocation - PF_VT_PFALLOC (0x001C0500; RO)

Field	Bit(s)	Init.	Type	Description
FIRSTVF	7:0	0x0	RW	The first VF allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-127
LASTVF	15:8	0x0	RW	The last VF allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-127
RESERVED	30:16	0x0	RSV	Reserved.
VALID	31	0b	RW	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.



11.2.2.11 PF - DCB Registers

11.2.2.11.1 Port DCB General Control - PRTDCB_GENC (0x00083000; RW)

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	00b	RW	Reserved.
NUMTC	5:2	0x1	RW	Number of Traffic Classes (TCs) for the port. This field is set consistently with the settings made in the Tx-scheduler.
FCOEUP	8:6	011b	RW	Defines the 802.1p UP field used for FCoE traffic over the link.
FCOEUP_VALID	9	1b	RW	Validity bit for the FCoEUP field. 0b = FCoE is not used over this port, and therefore RPB settings shall use the same Max Frame Size for all TCs. 1b = Means that the FCOEUP field contents is valid.
RESERVED	15:10	0x0	RSV	Reserved.
PFCLDA	31:16	0x079D	RW	PFC Link Delay Allowance. Expressed in 16 bytes units. Default value assumes 9.5 KB Jumbo frames over a 10 GbE link with a 10GBASE-T PHY and 100-meter Cat6 cable (no optimization done for lower link speeds). For a 40 GbE link, the number is 0x1E70.

11.2.2.11.2 Port DCB General Status - PRTDCB_GENS (0x00083020; RO)

Field	Bit(s)	Init.	Type	Description
DCBX_STATUS	2:0	000b	RW	DCBX status. 000b = NOT_STARTED 001b = IN_PROGRESS 010b = DONE 011b = MULTIPLE_PEERS 100b = Reserved 101b = Reserved 110b = Reserved 111b = DISABLED
RESERVED	31:3	0x0	RSV	Reserved.

11.2.2.11.3 Global DCB General Control - GLDCB_GENC (0x00083044; RW)

Field	Bit(s)	Init.	Type	Description
PCIRTT	15:0	0x009C	RW	PCIe Round Trip Time. Expressed in 16-byte units. Default is 2 µs PCIe round trip time assuming 10 GbE links (no optimization done for lower link speeds). For a 40 GbE link, the number is 0x0270.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.11.4 DCB Transmit ETS Control for TPB - PRTDCB_TETSC_TPB (0x00098060; RW)

Field	Bit(s)	Init.	Type	Description
RESERVED	7:0	0x0	RSV	Reserved.
LLTC	15:8	0x0	RW	8-bit wide bitmap with a 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 TC for transmit. 1b = TC is defined to be a Low Latency TC for transmit.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.11.5 DCB Transmit Frame Monitoring Status per TC - PRTDCB_TCMSTC[n] (0x000A0040 + 0x20*n, n=0...7; RO)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	Description
MSTC	19:0	0x0	RW	Monitoring Status of the TC Number of bytes in transit from the host to the TPB (TPB waiting list included) for TC n, where n is the index of the register in the array.
RESERVED	31:20	0x0	RSV	Reserved.

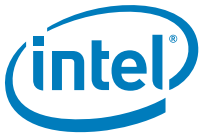
11.2.2.11.6 DCB Transmit Data Pipe Monitor Control - PRTDCB_TDPMC (0x000A0180; RW)

Field	Bit(s)	Init.	Type	Description
RESERVED	31:0	0x0	RSV	Reserved.

11.2.2.11.7 DCB Transmit Command Waiting Status per TC - PRTDCB_TCWSTC[n] (0x000A2040 + 0x20*n, n=0...7; RO)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	Description
MSTC	19:0	0x0	RW	Monitoring Status of the TC Number of commands 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	31:20	0x0	RSV	Reserved.



11.2.2.11.8 DCB Transmit Command Pipe Monitor Control - PRTDCB_TCPMC (0x000A21A0; RW)

Field	Bit(s)	Init.	Type	Description
CPM	12:0	0x098	RW	Depth of the per Port Monitor applied over the Tx Command Pipe Expressed in commands units.
LLTC	20:13	0x0	RW	0b = TC is bulk. 1b = TC is LL.
RESERVED2	31:21	0x0	RSV	Reserved.

11.2.2.11.9 DCB Transmit ETS Control for TCB - PRTDCB_TETSC_TCB (0x000AE060; RW)

Field	Bit(s)	Init.	Type	Description
RESERVED	7:0	0x0	RSV	Reserved.
LLTC	15:8	0x0	RW	8-bit wide bitmap with a 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 TC for transmit. 1b = TC is defined to be a Low Latency TC for transmit.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.11.10 DCB Receive ETS per TC Control - PRTDCB_RETSTCC[n] (0x00122180 + 0x20*n, n=0...7; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	Description
RESERVED	29:0	0x0	RSV	Reserved.
UPINTC_MODE	30	0b	RW	Rx-ETS operating mode when several UPs are attached to TC n, where n is the register index in the array. 0b = Strict Priority (SP) mode. 1b = Round Robin (RR) mode.
ETSTC	31	0b	RW	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.

11.2.2.11.11 DCB Receive per Port Pipe Monitor Control - PRTDCB_RPPMC (0x001223A0; RW)

Meaningless when DCBRSPMC.*RPM_MODE* is not set to 00b.

Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0x0	RSV	Reserved.
RX_FIFO_SIZE	23:16	0x08	RW	Rx Command FIFO Size. The number of Rx commands per TC that can be accumulated in RCB before sending back pressure to RCU.
RESERVED	31:24	0x0	RSV	Reserved.

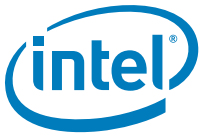
11.2.2.11.12 DCB Receive ETS Control - PRTDCB_RETSC (0x001223E0; RW)

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
NON_ETS_MODE	1	0b	RW	Rx Non-ETS operating mode 0b = Strict Priority (SP) mode. 1b = Round Robin (RR) mode.
RESERVED	7:2	0x0	RSV	Reserved.
LLTC	15:8	0x0	RW	8-bit wide bitmap with a 1-bit entry per each TC. Each entry controls whether the TC is considered to have low latency needs for the receive path. 0b = TC is defined to be a Bulk TC for receive. 1b = TC is defined to be a Low Latency TC for receive.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.11.13 DCB Receive per UP PFC Timer Queue - PRTDCB_RUPTQ[n] (0x00122400 + 0x20*n, n=0...7; RO)

One register per UP. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	Description
RXQNUM	13:0	0x0	RO	Rx Queue Number. This register returns the Rx Queue number (which had not enough available Rx descriptors) that caused the PFCTIMER of the Port/UP 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 and Rx queue registers. The value is meaningful only when the corresponding bit in PFDCB_RUPTI is set.
RESERVED	31:14	0x0	RSV	Reserved



11.2.2.11.14 DCB Receive per UP PFC Timer Indication - GLDCB_RUPTI (0x00122618; RO)

Field	Bit(s)	Init.	Type	Description
PFCTIMEOUT_UP	31:0	0x0	RW1C	PFC Time-out UPs. Bitmap where a bit set to 1b means that the PFC Timer corresponding to the Port/UP has timed out. Bits 0-7 are for port 0, UP 0 to 7. Bits 8-15 are for port 1, UP 0 to 7. Bits 16-23 are for port 2, UP 0 to 7. Bits 24-31 are for port 3, UP 0 to 7. Writing a bit with 1b restarts the corresponding PFC Timer.

11.2.2.11.15 DCB TC to PFC Mapping - PRTDCB_TC2PFC (0x001C0980; RW)

Field	Bit(s)	Init.	Type	Description
TC2PFC	7:0	0x0	RW	Bitmap that controls the use of Priority Flow Control (PFC) per each TC. Bit n set 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. 1b = TC n uses PFC in Rx and Tx. The TC is referred as a no-drop TC.
RESERVED	31:8	0x0	RSV	Reserved.

11.2.2.11.16 DCB Receive UP to TC Mapping for RCB - PRTDCB_RUP2TC (0x001C09A0; RW)

24-bit wide bitmap with a 3-bit entry per each UP. Each entry controls the mapping of a UP to a 3-bits 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.

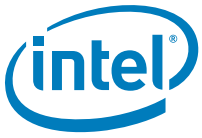
Field	Bit(s)	Init.	Type	Description
UP0TC	2:0	000b	RW	TC index to which UP 0 is mapped.
UP1TC	5:3	000b	RW	TC index to which UP 1 is mapped.
UP2TC	8:6	000b	RW	TC index to which UP 2 is mapped.
UP3TC	11:9	000b	RW	TC index to which UP 3 is mapped.
UP4TC	14:12	000b	RW	TC index to which UP 4 is mapped.
UP5TC	17:15	000b	RW	TC index to which UP 5 is mapped.
UP6TC	20:18	000b	RW	TC index to which UP 6 is mapped.
UP7TC	23:21	000b	RW	TC index to which UP 7 is mapped.
RESERVED	31:24	0x0	RSV	Reserved.

11.2.2.11.17 DCB Receive UP in PPRS - PRTDCB_RUP (0x001C0B00; RW)

Field	Bit(s)	Init.	Type	Description
NOVLANUP	2:0	000b	RW	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	Reserved.

11.2.2.11.18 MAC Flow Control Register - PRTDCB_MFLCN (0x001E2400; RW)

Field	Bit(s)	Init.	Type	Description
PMCF	0	0b	RW	Pass MAC Control Frames. Filter out unrecognized pause (flow control op code does not match) and other control frames. 0b = Filter unrecognized pause frames. 1b = Pass/forward unrecognized pause frames.
DPF	1	0b	RW	Discard Pause Frame. 0b = Pause frames are sent to the host. 1b = Pause frames are discarded when *RFCE* or *RPFCM* is set to 1b. Setting this bit to 1b has no effect otherwise (if both *RFCE* and *RPFCM* are set to 0b).
RPFCM	2	0b	RW	Receive Priority Flow Control Mode Indicates that the device responds to the reception of Priority Flow Control packets. If auto-negotiation is enabled, this bit should be set by software to the negotiated flow control value. This bit is set as a logical "OR" over the *RPFCE*[7:0] bitmap. It is useful to control forwarding of PFC frames to host, if required. ***Note:*** Receive Priority Flow Control and Receive Link Flow Control are mutually exclusive and user should not configure both of them to be enabled at the same time. ***Note:*** This bit should not be set if bit 3 is set.
RFCE	3	0b	RW	Receive Link Flow Control Enable Indicates that the device responds to the reception of Link Flow Control packets. If auto-negotiation is enabled, this bit should be set by software to the negotiated flow control value. ***Note:*** This bit should not be set if bit 2 is set.
RPFCE	11:4	0x0	RW	Receive Priority Flow Control Enable bitmap. When bit n is: 0b = Priority Flow Control indications received for UPn are ignored. 1b = Upon reception of Priority Flow Control packets for UPn, the device stops transmit over the TC to which UPn is mapped in Tx.
RESERVED	31:12	0x0	RSV	Reserved.



11.2.2.11.19 Transmit Flow Control Status - PRTDCB_TFCS (0x001E4560; RO)

Field	Bit(s)	Init.	Type	Description
TXOFF	0	0b	RO	Transmission Paused. Pause state indication of the transmit function when symmetrical link flow control is enabled.
RESERVED	7:1	0x0	RSV	Reserved.
TXOFF0	8	0b	RO	TC 0 Transmission Paused. Pause state indication of the TC 0 when priority flow control is enabled.
TXOFF1	9	0b	RO	TC 1 Transmission Paused. Pause state indication of the TC 1 when priority flow control is enabled.
TXOFF2	10	0b	RO	TC 2 Transmission Paused. Pause state indication of the TC 2 when priority flow control is enabled.
TXOFF3	11	0b	RO	TC 3 Transmission Paused. Pause state indication of the TC 3 when priority flow control is enabled.
TXOFF4	12	0b	RO	TC 4 Transmission Paused. Pause state indication of the TC 4 when priority flow control is enabled.
TXOFF5	13	0b	RO	TC 5 Transmission Paused. Pause state indication of the TC 5 when priority flow control is enabled.
TXOFF6	14	0b	RO	TC 6 Transmission Paused. Pause state indication of the TC 6 when priority flow control is enabled.
TXOFF7	15	0b	RO	TC 7 Transmission Paused. Pause state indication of the TC 7 when priority flow control is enabled.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.11.20 Flow Control Transmit Timer Value n - PRTDCB_FCTTVN[n] (0x001E4580 + 0x20*n, n=0...3; RW)

Field	Bit(s)	Init.	Type	Description
TTV_2N	15:0	0xFFFF	RW	Transmit Timer Value 2n. Timer value included in XOFF frames as Timer (2n). The same value is set to User Priorities attached to the same TC, as defined in the 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	Transmit Timer Value 2n+1. Timer value included in XOFF frames as Timer (2n+1). The same value is set to User Priorities attached to the same TC, as defined in PRTDCB_RUP2TC register.

11.2.2.11.21 Flow Control Refresh Threshold Value - PRTDCB_FCRTV (0x001E4600; RW)

Field	Bit(s)	Init.	Type	Description
FC_REFRESH_TH	15:0	0x7FFF	RW	Flow Control Refresh Threshold. 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	Reserved.

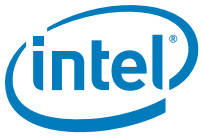
11.2.2.11.22 Flow Control Configuration - PRTDCB_FCCFG (0x001E4640; RW)

Field	Bit(s)	Init.	Type	Description
RESERVED	2:0	000b	RSV	Reserved.
TFCE	4:3	00b	RW	Transmit Flow Control Enable. Indicate that the XL710 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	Reserved.

11.2.2.11.23 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	Description
PFCTIMER	13:0	0x0	RW	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	Reserved.



11.2.2.12 PF - Receive Packet Buffer Registers

11.2.2.12.1 RPB Dedicated Pool High Watermark - PRTRPB_DHW[n] (0x000AC100 + 0x20*n, n=0...7; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	Description
DHW_TCN	19:0	0x0	RW	Dedicated Pool High Watermark for TC n. It is expressed in bytes.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.2 RPB Dedicated Pool Low Watermark - PRTRPB_DLW[n] (0x000AC220 + 0x20*n, n=0...7; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	Description
DLW_TCN	19:0	0x0	RW	Dedicated Pool Low Watermark for TC n. It is expressed in bytes.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.3 RPB Dedicated Pool Size - PRTRPB_DPS[n] (0x000AC320 + 0x20*n, n=0...7; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	Description
DPS_TCN	19:0	0x0	RW	Dedicated Pool Size for TC n. It is expressed in bytes.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.4 RPB Shared Pool High Threshold - PRTRPB_SHT[n] (0x000AC480 + 0x20*n, n=0...7; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	Description
SHT_TCN	19:0	0x3C800	RW	Shared Pool High Threshold for TC n. Expressed in bytes.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.5 RPB Shared Pool High Watermark - PRTRPB_SHW (0x000AC580; RW)

Field	Bit(s)	Init.	Type	Description
SHW	19:0	0x3C800	RW	Shared Pool High Watermark. Expressed in bytes.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.6 RPB Shared Pool Low Threshold - PRTRPB_SLT[n] (0x000AC5A0 + 0x20*n, n=0...7; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	Description
SLT_TCN	19:0	0x0	RW	Shared Pool Low Threshold for TC n. Expressed in bytes.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.7 RPB Shared Pool Low Watermark - PRTRPB_SLW (0x000AC6A0; RW)

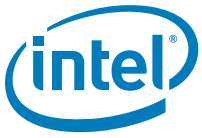
Field	Bit(s)	Init.	Type	Description
SLW	19:0	0x0	RW	Shared Pool Low Watermark. Expressed in bytes.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.8 RPB Shared Pool Size - PRTRPB_SPS (0x000AC7C0; RW)

Field	Bit(s)	Init.	Type	Description
SPS	19:0	0x3C800	RW	Shared Pool Size. Number of bytes allocated to the shared pool of the port.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.9 RPB Dedicated Pool Size for Single Shared Buffer State - GLRPB_DPSS (0x000AC828; RW)

Field	Bit(s)	Init.	Type	Description
DPS_TCN	19:0	0x0	RW	Dedicated Pool Size for the Single shared buffer state, for all TCs of all ports. Expressed in bytes.
RESERVED	31:20	0x0	RSV	Reserved.



11.2.2.12.10 RPB Global High Watermark - GLRPB_GHW (0x000AC830; RW)

Field	Bit(s)	Init.	Type	Description
GHW	19:0	0xF2000	RW	Global High Watermark. Expressed in bytes.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.11 RPB Global Low Watermark - GLRPB_GLW (0x000AC834; RW)

Field	Bit(s)	Init.	Type	Description
GLW	19:0	0x0	RW	Global Low Watermark. Expressed in bytes.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.12 RPB Packet High Watermark - GLRPB_PHW (0x000AC844; RW)

Field	Bit(s)	Init.	Type	Description
PHW	19:0	0x1246	RW	Packet High Watermark. Relative to the total number of packets stored in the RPB.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.12.13 RPB Packet Low Watermark - GLRPB_PLW (0x000AC848; RW)

Field	Bit(s)	Init.	Type	Description
PLW	19:0	0x0846	RW	Packet Low Watermark. Relative to the total number of packets stored in the RPB.
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.13 PF - Transmit Scheduler Registers

11.2.2.13.1 Transmit Scheduler Quanta Register - GLSCD_QUANTA (0x000B2080; RO)

This register is used to set a value of the Transmit Scheduler Quanta. Quanta controls amount of data that can be transmitted during single scheduling cycle.

Field	Bit(s)	Init.	Type	Description
TSCDQUANTA	2:0	0x2	RW	This field is used to set a quanta shift value. Quanta is calculated by (1024 << shift). Depending on the setting of the shift value, quanta can be set to 1 KB, 2 KB, 4 KB, 8 KB, and 16 KB values. Shift values of 4-7 are reserved, and should not be used.
RSVD	31:3	0x0	RSV	Reserved.

11.2.2.14 PF - Host Memory Cache Registers

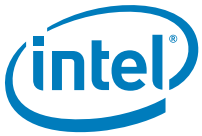
11.2.2.14.1 Private Memory Space Segment Descriptor Command - PFHMC_SDCMD (0x000C0000; RW)

This register is used to access the Host Memory Cache's segment table. The HMC's segment table is partitioned per PCI function. However, each PCIe PF is able access any segment table entry.

For read operations, PFHMC_SDCMD must be written with *PMSDWR* set to zero, and *PMSDIDX* must be set to the segment table index to be read. After the write operation 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 one and PMSDIDX set to the segment table index to be written.

Field	Bit(s)	Init.	Type	Description
PMSDIDX	11:0	0x0	RW	Relative Index of the HMC Segment Descriptor to be read or written. The actual index to be used to access the Segment Table is (*PMSDBASE* + *PMSDIDX*), where *PMSDBASE* is from the GLHMC_SDPART register associated with this function. On write operations, if (*PMSDIDX* >= GLHMC_SDPART.*PMSDSIZE*), the write is dropped.
RESERVED	30:12	0x0	RSV	Reserved.
PMSDWR	31	0b	RW	0b = Read operations. 1b = Write operations.



11.2.2.14.2 Private Memory Space Segment Descriptor Data Low - PFHMC_SDDATALOW (0x000C0100; 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	Description
PMSDVALID	0	0b	RW	Valid bit of an HMC segment descriptor table entry.
PMSDTYPE	1	0b	RW	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	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	Bits[31:12] of an HMC segment descriptor table entry.

11.2.2.14.3 Private Memory Space Segment Descriptor Data High - PFHMC_SDDATAHIGH (0x000C0200; 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	Description
PMSDDATAHIGH	31:0	0x0	RW	Most significant 32 bits of a segment descriptor.

11.2.2.14.4 Private Memory Space Page Descriptor Invalidate - PFHMC_PDINV (0x000C0300; RW)

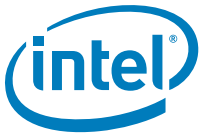
This register is used to invalidate cached HMC Page Descriptors that have been set to the invalid state by software.

Field	Bit(s)	Init.	Type	Description
PMSDIDX	11:0	0x0	RW	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 GLHMC_SDPART register associated with this function. On write operations, if (*PMSDIDX* >= GLHMC_SDPART.*PMSDSIZE*), the write is dropped.
RESERVED	15:12	0x0	RSV	Reserved.
PMPDIDX	24:16	0x0	RW	Index of the Page Descriptor within the Page Descriptor Page indicated by *PMSDIDX*.
RESERVED	31:25	0x0	RSV	Reserved.

11.2.2.14.5 Host Memory Cache Error Information Register - PFHMC_ERRORINFO (0x000C0400; RW)

This register reports the errors detected by the host memory cache. Errors reported through this register also may trigger interrupts through the TBD bit in the TBD register.

Field	Bit(s)	Init.	Type	Description
PMF_INDEX	4:0	0x0	RW	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	Reserved.
PMF_ISVF	7	0b	RW	Private Memory Function Is VF. 0b = The Private Memory Function reported in *PMF_INDEX* is associated with a PF.W 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	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 — The PMF Index programmed in the GLHMC_VFPMFMAP register is < 16 or > 47. This is a firmware error. 2 = Invalid PF for a Protocol Engine enabled VF in GLHMC_VFPMFTABLE — 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 = Invalid LAN Queue Index or FCoE VF Index — 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 = Object Index from transaction (or calculated absolute object index for FCoE objects) 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 = 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 = 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. The PFHMC_ERRORDATA register can be read to determine the following: • The LAN Queue index or FCoE VF index associated with error type 3. • The HMC object index associated with error types 4 and 12. • The HMC function relative *SD_Index* and *PD_Index* associated with error types 5 through 9.
RESERVED	15:12	0x0	RSV	Reserved.
HMC_OBJECT_TYPE	20:16	0x0	RW	Specifies the object type associated with the error.
RESERVED	30:21	0x0	RSV	Reserved.



Field	Bit(s)	Init.	Type	Description
ERROR_DETECTED	31	0b	RW	Error Detected. This field is set to 1b when a new error is detected by the HMC. No subsequent errors are recorded until this field is written with a value of 0b. A write of a 0b to this register clears the error and allows a subsequent error to be reported. Writes of 1b to this field are ignored.

11.2.2.14.6 Host Memory Cache Error Data Register - PFHMC_ERRORDATA (0x000C0500; RO)

This register reports the Private Memory Address or HMC object index related to an error detected by the Host Memory Cache.

Field	Bit(s)	Init.	Type	Description
HMC_ERROR_DATA	29:0	0x0	RO	Error Data. 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. • 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, and HMC_ERROR_DATA[29:28] should be zero. • When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 4 or 12, HMC_ERROR_DATA[27:0] reports the object index associated with the error, and HMC_ERROR_DATA[29:28] should be zero. • When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 5 thru 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 thru 9. • When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 0 thru 2, or 10 thru 11, HMC_ERROR_DATA[29:0] is not valid, and should be zero.
RESERVED	31:30	00b	RSV	Reserved.

11.2.2.14.7 Private Memory Segment Table Partitioning Registers - GLHMC_SDPART[n] (0x000C0800 + 0x4*n, n=0...15; RO)

This register is used to partition the shared Host Memory Cache segment table.

Field	Bit(s)	Init.	Type	Description
PMSDBASE	11:0	0x0	RW	Base segment table index for the function n.
RSVD	15:12	0x0	RSV	Reserved.
PMSDSIZE	28:16	0x0	RW	Number of valid segment table entries for the function n.
RSVD	31:29	000b	RSV	Reserved.

11.2.2.14.8 Private Memory Physical Function Table - GLHMC_PFASSIGN[n] (0x000C0C00 + 0x4*n, n=0...15; RO)

This register is used to allow Protocol Engine HMC resource sharing between PCIe Physical Functions. This enables load balancing capabilities for Protocol Engine functionality between PFs of a single device assigned to the same team.

Note: This register is initialized and managed by Protocol Engine firmware.

Field	Bit(s)	Init.	Type	Description
PMFCNPFASSIGN	3:0	0x0	RW	Private Memory Function Physical Function Assignment. This field is used to assign each PF to an HMC Private Memory Function. The default assignment is for each PCI Function to be assigned to the HMC Private Memory Function that matches the PF index. In cases where RDMA load balancing is desired, the PFs of a team must be assigned to the same HMC Private Memory Function. Only Protocol Engine objects use this table. LAN and FCoE objects all use the HMC Private Memory Function that matches the PF index. If PF 0 and 2 are part of a team and want to use Protocol Engine resources for load balancing, then *PMFCNPFASSIGN*[0] and *PMFCNPFASSIGN*[2] should be set to 0 and software should populate the Protocol Engine HMC backing pages using PF0.
RSVD	31:4	0x0	RSV	Reserved

11.2.2.14.9 Private Memory LAN TX Object Size - GLHMC_LANTXOBJSZ (0x000C2004; RO)

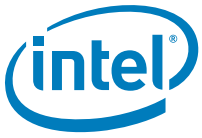
This register is used to calculate the amount of memory to allocate for Host Memory Cache LAN TX Queue objects. The value in this register is decoded such that the value = log2(ObjSize).

Field	Bit(s)	Init.	Type	Description
PMLANTXOBJSZ	3:0	0x7	RO	0x7 = 128 bytes.
RSVD	31:4	0x0	RSV	Reserved.

11.2.2.14.10 Private Memory LAN Queue Maximum - GLHMC_LANQMAX (0x000C2008; RO)

This register reports the maximum number of LAN Queues that any PCI function can utilize. The actual number of queues that a driver can use may be limited by NVM settings.

Field	Bit(s)	Init.	Type	Description
PMLANQMAX	10:0	0x600	RO	Private Memory LAN Queue Maximum. Reports the maximum number of LAN transmit or receive queue resources supported by the HMC. A given PCI function may be restricted to a smaller value by the NVRAM settings and the total number of enabled PCI functions.
RSVD	31:11	0x0	RSV	Reserved.



11.2.2.14.11 Private Memory LAN RX Object Size - GLHMC_LANRXOBJSZ (0x000C200C; RO)

This register is used to calculate the amount of memory to allocate for Host Memory Cache LAN RX Queue objects. The value in this register is decoded such that the value = log2(ObjSize).

Field	Bit(s)	Init.	Type	Description
PMLANRXOBJSZ	3:0	0x5	RO	0x5 = 32 bytes.
RSVD	31:4	0x0	RSV	Reserved.

11.2.2.14.12 Private Memory FCoE DDP Object Size - GLHMC_FCOEDDPOBJSZ (0x000C2010; RO)

This register is used to calculate the amount of memory to allocate for Host Memory Cache FCoE DDP Context objects.

Field	Bit(s)	Init.	Type	Description
PMFCOEDDPOBJSZ	3:0	0x6	RO	0x6 = 64 bytes.
RSVD	31:4	0x0	RSV	Reserved.

11.2.2.14.13 Private Memory FCoE Resource Maximum - GLHMC_FCOEMAX (0x000C2014; RO)

This register reports the maximum number of FCoE HMC context objects that are allocated per PCI PF or VF.

Field	Bit(s)	Init.	Type	Description
PMFCOEMAX	12:0	0x1000	RO	Private Memory FCoE Resource Maximum. Reports the maximum number of FCoE resources supported by the Host Memory Cache.
RSVD	31:13	0x0	RSV	Reserved.

11.2.2.14.14 Private Memory FCoE Filter Object Size - GLHMC_FCOEFOBJSZ (0x000C2018; RO)

This register is used to calculate the amount of memory to allocate for Host Memory Cache FCoE Filter objects.

Field	Bit(s)	Init.	Type	Description
PMFCOEFOBJSZ	3:0	0x6	RO	0x6 = 64 bytes.
RSVD	31:4	0x0	RSV	Reserved.

11.2.2.14.15 Private Memory FSI Multicast Group Object Size - GLHMC_FSIMCOBJSZ (0x000C205C; RO)

This register is used to calculate the amount of memory to allocate for Host Memory Cache FSI Multi-cast Group objects.

Field	Bit(s)	Init.	Type	Description
PMFSIMCOBJSZ	3:0	0x6	RO	0x6 = 64 bytes.
RSVD	31:4	0x0	RSV	Reserved.

11.2.2.14.16 Private Memory FSI Multicast Group Max - GLHMC_FSIMCMAX (0x000C2060; RO)

This register reports the maximum number of FSI Multicast Group Entries supported by the Host Memory Cache.

Field	Bit(s)	Init.	Type	Description
PMFSIMCMAX	13:0	0x2000	RO	
RSVD	31:14	0x0	RSV	Reserved.

11.2.2.14.17 Private Memory FSI Address Vector Object Size - GLHMC_FSIAVOBJSZ (0x000C2064; RO)

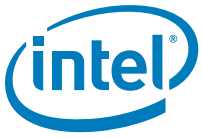
This register is used to calculate the amount of memory to allocate for Host Memory Cache FSI Address Vector objects.

Field	Bit(s)	Init.	Type	Description
PMFSIAVOBJSZ	3:0	0x5	RO	0x5 = 32 bytes.
RSVD	31:4	0x0	RSV	

11.2.2.14.18 Private Memory FSI Address Vector Max - GLHMC_FSIAVMAX (0x000C2068; RO)

This register reports the maximum number of FSI Address Vectors supported by the Host Memory Cache.

Field	Bit(s)	Init.	Type	Description
PMFSIAVMAX	16:0	0x10000	RO	
RSVD	31:17	0x0	RSV	



11.2.2.14.19 Private Memory FCoE Filter Resource Maximum - GLHMC_FCOEFMAX (0x000C20D0; RO)

This register reports the maximum number of FCoE Filter HMC context objects that are allocated per PCI PF or VF.

Field	Bit(s)	Init.	Type	Description
PMFCOEFMAX	15:0	0x9000	RO	Private Memory FCoE Filter Resource Maximum. This field reports the maximum number of FCoE filter resources supported by the Host Memory Cache per PCI function. The total number of FCoE filter resources allocated to a PCI function via the GLHMC_FCOEFCNT register must include this value multiplied by the number of VFs that will support FCoE plus 1 for the PF.
RSVD	31:16	0x0	RSV	Reserved.

11.2.2.14.20 FPM FSI Address Vector Base - GLHMC_FSIAVBASE[n] (0x000C5600 + 0x4*n, n=0...15; RW)

This register reports the Function Private Memory space base address for the FSI Address Vector 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.

Field	Bit(s)	Init.	Type	Description
FPMFSIAVBASE	23:0	0x0	RW	
RSVD	31:24	0x0	RSV	Reserved.

11.2.2.14.21 FPM FSI Address Vector Object Count - GLHMC_FSIAVCNT[n] (0x000C5700 + 0x4*n, n=0...15; RW)

This register is 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.

Field	Bit(s)	Init.	Type	Description
FPMFSIAVCNT	28:0	0x0	RW	
RSVD	31:29	0x0	RW	Reserved.

11.2.2.14.22 FPM FSI Multicast Group Base - GLHMC_FSIMCBASE[n] (0x000C6000 + 0x4*n, n=0...15; RW)

This register reports the Function Private Memory space base address for the FSI Multicast Group 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.

Field	Bit(s)	Init.	Type	Description
FPMFSIMCBASE	23:0	0x0	RW	
RSVD	31:24	0x0	RSV	Reserved.

11.2.2.14.23 FPM FSI Multicast Group Object Count - GLHMC_FSIMCCNT[n] (0x000C6100 + 0x4*n, n=0...15; RW)

This register is 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.

Field	Bit(s)	Init.	Type	Description
FPMFSIMCSZ	28:0	0x0	RW	
RSVD	31:29	0x0	RSV	Reserved.

11.2.2.14.24 FPM LAN TX Queue Base - GLHMC_LANTXBASE[n] (0x000C6200 + 0x4*n, n=0...15; RW)

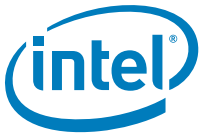
This register reports the Function Private Memory space base address for the LAN Transmit Queue objects in 512-byte increments. In other words, the value in this register must be multiplied by 512 to get the actual address out of the 8 GB FPM address space.

Field	Bit(s)	Init.	Type	Description
FPMLANTXBASE	23:0	0x0	RW	
RSVD	31:24	0x0	RW	Reserved.

11.2.2.14.25 FPM LAN TX Queue Object Count - GLHMC_LANTXCNT[n] (0x000C6300 + 0x4*n, n=0...15; RW)

This register is used to set the Function Private Memory space size for the LAN TX Queue objects.

Field	Bit(s)	Init.	Type	Description
FPMLANTXCNT	10:0	0x0	RW	
RSVD	31:11	0x0	RSV	Reserved.



11.2.2.14.26 FPM LAN RX Queue Base - GLHMC_LANRXBASE[n] (0x000C6400 + 0x4*n, n=0...15; RW)

This register reports the Function Private Memory space base address for the LAN Receive Queue 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.

Field	Bit(s)	Init.	Type	Description
FPMLANRXBASE	23:0	0x0	RW	
RSVD	31:24	0x0	RSV	Reserved.

11.2.2.14.27 FPM LAN RX Queue Object Count - GLHMC_LANRXCNT[n] (0x000C6500 + 0x4*n, n=0...15; RW)

This register is used to set the Function Private Memory space size for the LAN RX Queue objects.

Field	Bit(s)	Init.	Type	Description
FPMLANRXCNT	10:0	0x0	RW	
RSVD	31:11	0x0	RSV	Reserved.

11.2.2.14.28 FPM FCoE DDP Base - GLHMC_FCOEDDPBASE[n] (0x000C6600 + 0x4*n, n=0...15; RW)

This register reports the Function Private Memory space base address for the FCoE DDP 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.

Field	Bit(s)	Init.	Type	Description
FPMFCOEDDPBASE	23:0	0x0	RW	
RSVD	31:24	0x0	RSV	Reserved.

11.2.2.14.29 FPM FCoE DDP Object Count - GLHMC_FCOEDDPCNT[n] (0x000C6700 + 0x4*n, n=0...15; RW)

This register is used to set the Function Private Memory space size for the FCoE DDP objects.

Field	Bit(s)	Init.	Type	Description
FPMFCOEDDPCNT	19:0	0x0	RW	
RSVD	31:20	0x0	RSV	Reserved.

11.2.2.14.30 FPM FCoE Filters Base - GLHMC_FCOEFBASE[n] (0x000C6800 + 0x4*n, n=0...15; RW)

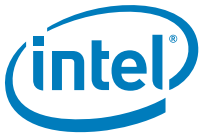
This register reports the Function Private Memory space base address for the FCoE Filters objects in 512-byte increments. In other words, the value in this register must be multiplied by 512 to get the actual address out of the 8 GB FPM address space.

Field	Bit(s)	Init.	Type	Description
FPMFCOEFBASE	23:0	0x0	RW	
RSVD	31:24	0x0	RSV	Reserved.

11.2.2.14.31 FPM FCoE Filters Object Count - GLHMC_FCOEFCNT[n] (0x000C6900 + 0x4*n, n=0...15; RW)

This register is used to set the Function Private Memory space size for the FCoE Filters objects.

Field	Bit(s)	Init.	Type	Description
FPMFCOEFCNT	22:0	0x0	RW	
RSVD	31:23	0x0	RSV	Reserved.



11.2.2.15 PF - Context Manager Registers

11.2.2.15.1 CMLAN Error Info - PFCM_LAN_ERRINFO (0x0010C000; RO)

This register reports errors for internal CM clients.

Field	Bit(s)	Init.	Type	Description
ERROR_VALID	0	0b	RO	Indicates the information in ERRINFO and ERRDATA is valid. Writing a 0 to this bit has no affect. Writing a 1 to this bit clears the contents of ERRINFO and ERRDATA.
RESERVED	3:1	000b	RSV	Reserved.
ERROR_INST	6:4	000b	RO	Indicates the internal unit that reported the error. 001b = DBL 010b = RLU 011b = RLS All other values are reserved.
RESERVED	7	0b	RSV	Reserved.
DBL_ERROR_CNT	15:8	0x0	RO	Number of DBL errors reported.
RLU_ERROR_CNT	23:16	0x0	RO	Number of RLU errors reported.
RLS_ERROR_CNT	31:24	0x0	RO	Number of RLS errors reported.

11.2.2.15.2 CMLAN Error Data - PFCM_LAN_ERRDATA (0x0010C080; RO)

This register reports errors for internal CM clients.

Field	Bit(s)	Init.	Type	Description
ERROR_CODE	3:0	0x0	RO	Error code. 0001b = PMAT Error — Other 0010b = PMAT Error — Dummy completion 0100b = Context CRC Error All other values are reserved.
Q_TYPE	6:4	000b	RO	
RESERVED	7	0b	RSV	Reserved.
Q_NUM	19:8	0x0	RO	
RESERVED	31:20	0x0	RSV	Reserved.

11.2.2.15.3 CMLAN Context Data Registers - PFCM_LANCTXDATA[n] (0x0010C100 + 0x80*n, n=0...3; RW)

32-bit portion of the 128-bit context sub-line data. This register is indexed by PF index (n) and by context word index. Word index 0 is the least significant word and line 3 is the most significant word.

Field	Bit(s)	Init.	Type	Description
DATA	31:0	0x0	RW	

11.2.2.15.4 CMLAN Context Control Register - PFCM_LANCTXCTL (0x0010C300; RW)

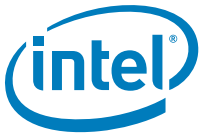
This register provides an interface into the context cache for pre-boot context initialization.

Field	Bit(s)	Init.	Type	Description
QUEUE_NUM	11:0	0x0	RW	Specifies the LAN Queue index to be loaded or invalidated. This number is an absolute queue number.
SUB_LINE	14:12	000b	RW	Specifies the 16-byte sub-line to be accessed for write operations.
QUEUE_TYPE	16:15	00b	RW	Specifies the queue type: 00b = LAN RX Context 01b = LAN TX Context 10b = FCoE DDP RX Context 11b = Reserved
OP_CODE	18:17	00b	RW	Specifies the operation type: 00b = Read 01b = Write 10b = Invalidate 11b = Reserved
RSVD	31:19	0x0	RSV	Reserved.

11.2.2.15.5 CMLAN Context Status Register - PFCM_LANCTXSTAT (0x0010C380; RO)

This register provides completion status for operations initiated via the CMLAN context control register.

Field	Bit(s)	Init.	Type	Description
CTX_DONE	0	0b	RO	Context Operation Done. 0b = A previous LAN context operation that was initiated by a write to the PFCM_LANCTXCTL register is in progress, or no LAN context operation has been issued. 1b = A previous LAN context operation that was initiated by a write to the PFCM_LANCTXCTL register has completed.
CTX_MISS	1	0b	RO	Context Queue Number Not Present. This field reports if the queue was in the context cache before the operation specified in the PFCM_LANCTXCGL.*OP_CODE*. 0b = The requested queue number was resident in the context cache. 1b = The requested queue number was not resident in the context cache.
RSVD	31:2	0x0	RSV	Reserved.



11.2.2.15.6 CMPE VF Error Info - VFCM_PE_ERRINFO[VF] (0x00138400 + 0x4*VF, VF=0...127; RO)

This register reports errors for internal CM clients.

Field	Bit(s)	Init.	Type	Description
ERROR_VALID	0	0b	RO	Indicates the information in ERRINFO and ERRDATA is valid. Writing a 0 to this bit has no affect. Writing a 1 to this bit clears the contents of ERRINFO and ERRDATA.
RESERVED	3:1	000b	RSV	Reserved.
ERROR_INST	6:4	000b	RO	Indicates the internal unit that reported the error. 001b = DBL 010b = RLU 011b = RLS All other values are reserved.
RESERVED	7	0b	RSV	Reserved.
DBL_ERROR_CNT	15:8	0x0	RO	Number of DBL errors reported.
RLU_ERROR_CNT	23:16	0x0	RO	Number of RLU errors reported.
RLS_ERROR_CNT	31:24	0x0	RO	Number of RLS errors reported.

11.2.2.15.7 CMPE VF Error Data - VFCM_PE_ERRDATA[VF] (0x00138800 + 0x4*VF, VF=0...127; RO)

This register reports errors for internal CM clients.

Field	Bit(s)	Init.	Type	Description
ERROR_CODE	3:0	0x0	RO	Error code. 0001b = PMAT Error — Other 0010b = PMAT Error — Dummy completion 0100b = Context CRC Error All other values are reserved.
Q_TYPE	6:4	000b	RO	
RESERVED	7	0b	RSV	Reserved.
Q_NUM	25:8	0x0	RO	
RESERVED	31:26	0x0	RSV	Reserved.

11.2.2.16 PF - Admin Queue

11.2.2.16.1 PF Admin Transmit Queue Base Address Low - PF_ATQBAL (0x00080000; RW)

Field	Bit(s)	Init.	Type	Description
ATQBAL	31:0	0x0	RW	Transmit descriptor base address low. Must be 64-byte aligned.

11.2.2.16.2 Global Admin Transmit Queue Base Address Low - GL_ATQBAL (0x00080040; RW)

Field	Bit(s)	Init.	Type	Description
ATQBAL	31:0	0x0	RW	Transmit descriptor base address low. Must be 64-byte aligned.

11.2.2.16.3 PF Admin Receive Queue Base Address Low - PF_ARQBAL (0x00080080; RW)

Field	Bit(s)	Init.	Type	Description
ARQBAL	31:0	0x0	RW	Receive descriptor base address low. Must be 64-byte aligned.

11.2.2.16.4 Global Admin Receive Queue Base Address Low - GL_ARQBAL (0x000800C0; RW)

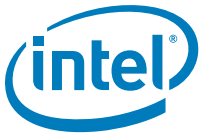
Field	Bit(s)	Init.	Type	Description
ARQBAL	31:0	0x0	RW	Receive descriptor base address low. Must be 64-byte aligned.

11.2.2.16.5 PF Admin Transmit Queue Base Address High - PF_ATQBAH (0x00080100; RW)

Field	Bit(s)	Init.	Type	Description
ATQBAH	31:0	0x0	RW	Transmit descriptor base address high.

11.2.2.16.6 Global Admin Transmit Queue Base Address High - GL_ATQBAH (0x00080140; RW)

Field	Bit(s)	Init.	Type	Description
ATQBAH	31:0	0x0	RW	Transmit descriptor base address high.



11.2.2.16.7 PF Admin Receive Queue Base Address High - PF_ARQBAH (0x00080180; RW)

Field	Bit(s)	Init.	Type	Description
ARQBAH	31:0	0x0	RW	Receive descriptor base address high.

11.2.2.16.8 Global Admin Receive Queue Base Address High - GL_ARQBAH (0x000801C0; RW)

Field	Bit(s)	Init.	Type	Description
ARQBAH	31:0	0x0	RW	Receive descriptor base address high.

11.2.2.16.9 PF Admin Transmit Queue Length - PF_ATQLEN (0x00080200; RW)

Field	Bit(s)	Init.	Type	Description
ATQLEN	9:0	0x0	RW	Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	Reserved.
ATQVFE	28	0b	RW	VF Error. Set by FW on a PF queue when one of its VFs has an admin queue error.
ATQOVFL	29	0b	RW	Overflow Error. Set by FW when a message is lost because there is no room on the queue.
ATQCRIT	30	0b	RW	Critical Error. Set by FW when a critical error is detected on this queue.
ATQENABLE	31	0b	RW	Enable bit. Set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is cleared by PFR.

11.2.2.16.10 Global Admin Transmit Queue Length - GL_ATQLEN (0x00080240; RW)

Field	Bit(s)	Init.	Type	Description
ATQLEN	9:0	0x0	RW	Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	Reserved.
ATQVFE	28	0b	RW	VF Error. Set by FW on a PF queue when one of its VFs has an admin queue error.
ATQOVFL	29	0b	RW	Overflow Error. Set by FW when a message is lost because there is no room on the queue.
ATQCRIT	30	0b	RW	Critical Error. Set by FW when a critical error is detected on this queue.
ATQENABLE	31	0b	RW	Enable bit. Set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is cleared by CORER.

11.2.2.16.11 PF Admin Receive Queue Length - PF_ARQLEN (0x00080280; RW)

Field	Bit(s)	Init.	Type	Description
ARQLEN	9:0	0x0	RW	Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	Reserved.
ARQVFE	28	0b	RW	VF Error. Set by FW on a PF queue when one of its VFs has an admin queue error.
ARQOVFL	29	0b	RW	Overflow Error. Set by FW when a message is lost because there is no room on the queue.
ARQCRIT	30	0b	RW	Critical Error. Set by FW when a critical error is detected on this queue.
ARQENABLE	31	0b	RW	Enable bit. Set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is cleared by PFR.

11.2.2.16.12 PF Admin Transmit Head - PF_ATQH (0x00080300; RW)

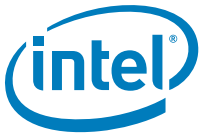
Field	Bit(s)	Init.	Type	Description
ATQH	9:0	0x0	RW	Transmit queue head pointer. At queue initialization, software clears the head pointer. During nominal operation, the Firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.16.13 Global Admin Transmit Head - GL_ATQH (0x00080340; RW)

Field	Bit(s)	Init.	Type	Description
ATQH	9:0	0x0	RW	Transmit queue head pointer. At queue initialization, software clears the head pointer. During nominal operation, the Firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.16.14 PF Admin Receive Queue Head - PF_ARQH (0x00080380; RW)

Field	Bit(s)	Init.	Type	Description
ARQH	9:0	0x0	RW	Receive queue head pointer. At queue initialization, software clears the head pointer. During nominal operation, the Firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	Reserved.



11.2.2.16.15 Global Admin Receive Queue Head - GL_ARQH (0x000803C0; RW)

Field	Bit(s)	Init.	Type	Description
ARQH	9:0	0x0	RW	Receive queue head pointer. At queue initialization, software clears the head pointer. During nominal operation, the Firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.16.16 PF Admin Transmit Tail - PF_ATQT (0x00080400; RW)

Field	Bit(s)	Init.	Type	Description
ATQT	9:0	0x0	RW	Transmit queue tail. Incremented to indicate that there are new valid descriptors on the ring. SW may only write to this register once both transmit and receive queues are properly initialized,. Clears to zero at queue initialization.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.16.17 Global Admin Transmit Tail - GL_ATQT (0x00080440; RW)

Field	Bit(s)	Init.	Type	Description
ATQT	9:0	0x0	RW	Transmit queue tail. Incremented to indicate that there are new valid descriptors on the ring. SW may only write to this register once both transmit and receive queues are properly initialized. Clears to zero at queue initialization.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.16.18 PF Admin Receive Queue Tail - PF_ARQT (0x00080480; RW)

Field	Bit(s)	Init.	Type	Description
ARQT	9:0	0x0	RW	Receive queue tail. Incremented to indicate that there are new valid descriptors on the ring. SW may only write to this register once the queue is fully configured. Clears to zero at queue initialization.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.16.19 Global Admin Receive Queue Tail - GL_ARQT (0x000804C0; RW)

Field	Bit(s)	Init.	Type	Description
ARQT	9:0	0x0	RW	Receive queue tail. Incremented to indicate that there are new valid descriptors on the ring. SW may only write to this register once the queue is fully configured. Clears to zero at queue initialization.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.16.20 VF Admin Transmit Queue Base Address Low - VF_ATQBAL[VF] (0x00080800 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
ATQBAL	31:0	0x0	RW	Transmit descriptor base address low. Must be 64-byte aligned.

11.2.2.16.21 VF Admin Receive Queue Base Address Low - VF_ARQBAL[VF] (0x00080C00 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
ARQBAL	31:0	0x0	RW	Receive descriptor base address low. Must be 64-byte aligned.

11.2.2.16.22 VF Admin Transmit Queue Base Address High - VF_ATQBAH[VF] (0x00081000 + 0x4*VF, VF=0...127; RW)

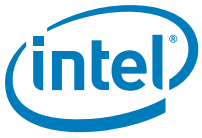
Field	Bit(s)	Init.	Type	Description
ATQBAH	31:0	0x0	RW	Transmit descriptor base address high.

11.2.2.16.23 VF Admin Receive Queue Base Address High - VF_ARQBAH[VF] (0x00081400 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
ARQBAH	31:0	0x0	RW	Receive descriptor base address high.

11.2.2.16.24 VF Admin Transmit Queue Length - VF_ATQLEN[VF] (0x00081800 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
ATQLEN	9:0	0x0	RW	Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	Reserved.
ATQVFE	28	0b	RW	VF Error. Set by FW on a PF queue when one of its VFs has an admin queue error.
ATQOVFL	29	0b	RW	Overflow Error. Set by FW when a message is lost because there is no room on the queue.
ATQCRIT	30	0b	RW	Critical Error. This bit is set by FW when a critical error is detected on this queue.
ATQENABLE	31	0b	RW	Enable bit. Set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is cleared by VFR.



11.2.2.16.25 VF Admin Receive Queue Length - VF_ARQLEN[VF] (0x00081C00 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
ARQLEN	9:0	0x0	RW	Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	Reserved.
ARQVFE	28	0b	RW	VF Error. Set by FW on a PF queue when one of its VFs has an admin queue error.
ARQOVFL	29	0b	RW	Overflow Error. Set by FW when a message is lost because there is no room on the queue.
ARQCRIT	30	0b	RW	Critical Error. This bit is set by FW when a critical error is detected on this queue.
ARQENABLE	31	0b	RW	Enable bit. Set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is cleared by PFR.

11.2.2.16.26 VF Admin Transmit Head - VF_ATQH[VF] (0x00082000 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
ATQH	9:0	0x0	RW	Transmit queue head pointer. At queue initialization, software clears the head pointer. During nominal operation, the Firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.16.27 VF Admin Receive Queue Head - VF_ARQH[VF] (0x00082400 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
ARQH	9:0	0x0	RW	Receive queue head pointer. At queue initialization, software clears the head pointer. During nominal operation, the Firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.16.28 VF Admin Transmit Tail - VF_ATQT[VF] (0x00082800 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
ATQT	9:0	0x0	RW	Transmit queue tail. Incremented to indicate that there are new valid descriptors on the ring. SW may only write to this register once both transmit and receive queues are properly initialized. Clears to zero at queue initialization.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.16.29 VF Admin Receive Queue Tail - VF_ARQT[VF] (0x00082C00 + 0x4*VF, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
ARQT	9:0	0x0	RW	Receive queue tail. Incremented to indicate that there are new valid descriptors on the ring. SW may only write to this register once the queue is fully configured. Clears to zero at queue initialization.
RESERVED	31:10	0x0	RSV	Reserved.

11.2.2.17 PF - Statistics Registers

11.2.2.17.1 Port Good Octets Received Count Low - GLPRT_GORCL[n] (0x00300000 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
GORCL	31:0	0x0	RW1C	Good Octets Received Count Low. Counts 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.

11.2.2.17.2 Port Good Octets Received Count High - GLPRT_GORCH[n] (0x00300004 + 0x8*n, n=0...3; RW1C)

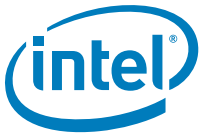
Field	Bit(s)	Init.	Type	Description
GORCH	15:0	0x0	RW1C	Good Octets Received Count High. Counts number of bytes received by this port. Higher 16 bits. The low and high registers are part of a 64-bit register, and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.3 Port MAC Local Fault Count - GLPRT_MLFC[n] (0x00300020 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
MLFC	31:0	0x0	RW1C	MAC Local Fault Count. Number of faults in the local MAC.

11.2.2.17.4 Port MAC Remote Fault Count - GLPRT_MRFC[n] (0x00300040 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
MRFC	31:0	0x0	RW1C	MAC Remote Fault Count. Number of faults in the remote MAC.



11.2.2.17.5 Port CRC Error Count - GLPRT_CRCERRS[n] (0x00300080 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
CRCERRS	31:0	0x0	RW1C	CRC Error Count. Counts the number of receive packets with CRC errors. This includes packets that are also counted by other error registers (refer to section 7.11.3 for details).

11.2.2.17.6 Receive Length Error Count - GLPRT_RLEC[n] (0x003000A0 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
RLEC	31:0	0x0	RW1C	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.

11.2.2.17.7 Port Illegal Byte Error Count - GLPRT_ILLERRC[n] (0x003000E0 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
ILLERRC	31:0	0x0	RW1C	Illegal Byte Error Packet Count. Counts the number of receive packets with illegal bytes errors (i.e., there is an illegal symbol in the packet).

11.2.2.17.8 Receive Undersize Count - GLPRT_RUC[n] (0x00300100 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
RUC	31:0	0x0	RW1C	Receive Undersize Count. 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.

11.2.2.17.9 Receive Oversize Count - GLPRT_ROC[n] (0x00300120 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
ROC	31:0	0x0	RW1C	Receive Oversize Count. 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.

11.2.2.17.10 Port Link XON Received Count - GLPRT_LXONRXC[n] (0x00300140 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
LXONRXCNT	31:0	0x0	RW1C	Link XON Received Count. Number of XON packets received.

11.2.2.17.11 Port Link XOFF Received Count - GLPRT_LXOFFRXC[n] (0x00300160 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
LXOFFRXCNT	31:0	0x0	RW1C	Link XOFF Received Count. Number of XOFF packets received.

11.2.2.17.12 Priority XON Received Count - GLPRT_PXONRXC[n,m] (0x00300180 + 0x8*n + 0x20*m, n=0...3, m=0...7; RW1C)

Port Priority FC XON received count (array of 8 per port).

Field	Bit(s)	Init.	Type	Description
PRPXONRXCNT	31:0	0x0	RW1C	Priority XON Received Count. Number of XON packets received.

11.2.2.17.13 Priority XOFF Received Count - GLPRT_PXOFFRXC[n,m] (0x00300280 + 0x8*n + 0x20*m, n=0...3, m=0...7; RW1C)

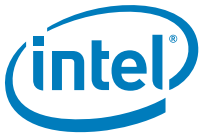
Port Priority FC XOFF received count (array of 8 per port).

Field	Bit(s)	Init.	Type	Description
PRPXOFFRXCNT	31:0	0x0	RW1C	Priority XOFF Received Count. Number of XOFF packets received.

11.2.2.17.14 Priority XON to XOFF Count - GLPRT_RXON2OFFCNT[n,m] (0x00300380 + 0x8*n + 0x20*m, n=0...3, m=0...7; RW1C)

Counts the number of times the Priority egress was paused (array of 8 per port).

Field	Bit(s)	Init.	Type	Description
PRRXON2OFFCNT	31:0	0x0	RW1C	Priority XON to XOFF Count. Number of times transmitter transitioned from XON to XOFF.



11.2.2.17.15 Packets Received [64 Bytes] Count Low - GLPRT_PRC64L[n] (0x00300480 + 0x8*n, n=0...3; RW1C)

Port packets received (64 bytes) count low. Combined with GLPRT_PRC64H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC64L	31:0	0x0	RW1C	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.

11.2.2.17.16 Packets Received [64 Bytes] Count High - GLPRT_PRC64H[n] (0x00300484 + 0x8*n, n=0...3; RW1C)

Port packets received (64 bytes) count high. Combined with GLPRT_PRC64L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC64H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.17 Packets Received [65-127 Bytes] Count Low - GLPRT_PRC127L[n] (0x003004A0 + 0x8*n, n=0...3; RW1C)

Port packets received (64 to 127 bytes) count low. Combined with GLPRT_PRC127H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC127L	31:0	0x0	RW1C	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.

11.2.2.17.18 Packets Received [65-127 Bytes] Count High - GLPRT_PRC127H[n] (0x003004A4 + 0x8*n, n=0...3; RW1C)

Port packets received (64 to 127 bytes) count low. Combined with GLPRT_PRC127L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC127H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.19 Packets Received [128-255 Bytes] Count Low - GLPRT_PRC255L[n] (0x003004C0 + 0x8*n, n=0...3; RW1C)

Port packets received (128 to 255 bytes) count low. Combined with GLPRT_PRC255H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC255L	31:0	0x0	RW1C	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.

11.2.2.17.20 Packets Received [128-255 Bytes] Count High - GLPRT_PRC255H[n] (0x003004C4 + 0x8*n, n=0...3; RW1C)

Port packets received (128 to 255 bytes) count low. Combined with GLPRT_PRC255L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRTPRC255H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.21 Packets Received [256-511 Bytes] Count Low - GLPRT_PRC511L[n] (0x003004E0 + 0x8*n, n=0...3; RW1C)

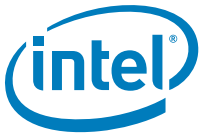
Port packets received (256 to 511 bytes) count low. Combined with GLPRT_PRC511H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC511L	31:0	0x0	RW1C	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.

11.2.2.17.22 Packets Received [256-511 Bytes] Count High - GLPRT_PRC511H[n] (0x003004E4 + 0x8*n, n=0...3; RW1C)

Port packets received (256 to 511 bytes) count low. Combined with GLPRT_PRC511L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC511H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.



11.2.2.17.23 Packets Received [512-1023 Bytes] Count Low - GLPRT_PRC1023L[n] (0x00300500 + 0x8*n, n=0...3; RW1C)

Port packets received (512 to 1023 bytes) count low. Combined with GLPRT_PRC1023H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC1023L	31:0	0x0	RW1C	Packets Received Count (512-1023 Bytes) Low. Number of packets received that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). 0The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

11.2.2.17.24 Packets Received [512-1023 Bytes] Count High - GLPRT_PRC1023H[n] (0x00300504 + 0x8*n, n=0...3; RW1C)

Port packets received (512 to 1023 bytes) count low. Combined with GLPRT_PRC1023L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC1023H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.25 Packets Received [1024-1522 Bytes] Count Low - GLPRT_PRC1522L[n] (0x00300520 + 0x8*n, n=0...3; RW1C)

Port packets received (1024 to 1522 bytes) count low. Combined with GLPRT_PRC1522H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC1522L	31:0	0x0	RW1C	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.

11.2.2.17.26 Packets Received [1024-1522 Bytes] Count High - GLPRT_PRC1522H[n] (0x00300524 + 0x8*n, n=0...3; RW1C)

Port packets received (1024 to 1522 bytes) count low. Combined with GLPRT_PRC1522L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC1522H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.27 Packets Received [1523-9522 Bytes] Count Low - GLPRT_PRC9522L[n] (0x00300540 + 0x8*n, n=0...3; RW1C)

Port packets received (1523 to 9522 bytes) count low. Combined with GLPRT_PRC9522H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PRC9522L	31:0	0x0	RW1C	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.

11.2.2.17.28 Packets Received [1523-9522 Bytes] Count High - GLPRT_PRC9522H[n] (0x00300544 + 0x8*n, n=0...3; RW1C)

Port packets received (1523 to 9522 bytes) count low. Combined with GLPRT_PRC9522L to form a 64-bit register.

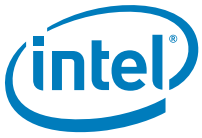
Field	Bit(s)	Init.	Type	Description
PRC9522H	15:0	0x0	RW1C	Packets Received Count (1523-9522 Bytes) High. Number of packets received that are 1523-9522 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.29 Receive Fragment Count - GLPRT_RFC[n] (0x00300560 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
RFC	31:0	0x0	RW1C	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.

11.2.2.17.30 Receive Jabber Count - GLPRT_RJC[n] (0x00300580 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
RJC	31:0	0x0	RW1C	Receive Jabber Count (number of receive jabber errors). Counts the number of received packets that passed address filtering, and are greater than maximum size and have bad CRC (this is slightly different from the Receive Oversize Count register). The packets length is counted from <Destination Address> through <CRC>, inclusively.



11.2.2.17.31 Port Unicast Packets Received Count Low - GLPRT_UPRCL[n] (0x003005A0 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
UPRCL	31:0	0x0	RW1C	Unicast Packets Received Count Low. Counts number of unicast packets received 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.

11.2.2.17.32 Port Unicast Packets Received Count High - GLPRT_UPRCH[n] (0x003005A4 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
UPRCH	15:0	0x0	RW1C	Unicast Packets Received Count High. Counts number of unicast packets received by this VSI. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.33 Port Multicast Packets Received Count Low - GLPRT_MPRCL[n] (0x003005C0 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
MPRCL	31:0	0x0	RW1C	Multicast Packets Received Count Low. Counts number of multicast packets received 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.

11.2.2.17.34 Port Multicast Packets Received Count High - GLPRT_MPRCH[n] (0x003005C4 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
MPRCH	15:0	0x0	RW1C	Multicast Packets Received Count High. Counts number of multicast packets received by this VSI. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.35 Port Broadcast Packets Received Count Low - GLPRT_BPRCL[n] (0x003005E0 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
BPRCL	31:0	0x0	RW1C	Broadcast Packets Received Count Low. Counts number of broadcast packets received by this port. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

11.2.2.17.36 Port Broadcast Packets Received Count High - GLPRT_BPRCH[n] (0x003005E4 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
BPRCH	15:0	0x0	RW1C	Broadcast Packets Received Count High. Counts number of broadcast packets received by this port. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.37 Port Receive Packets Discarded Count - GLPRT_RDPC[n] (0x00300600 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
RDPC	31:0	0x0	RW1C	Port Receive Packets Discarded Count.

11.2.2.17.38 Loopback Packets Discarded Count - GLPRT_LDPC[n] (0x00300620 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
LDPC	31:0	0x0	RW1C	VM to VM Loopback Packets Discarded Count.

11.2.2.17.39 Port Received with No Destination - GLPRT_RUPP[n] (0x00300660 + 0x8*n, n=0...3; RW1C)

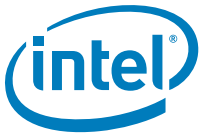
Field	Bit(s)	Init.	Type	Description
RUPP	31:0	0x0	RW1C	Receive packets dropped for this port because thy did not match any S-TAG. (No forwarding rule).

11.2.2.17.40 Port Good Octets Transmit Count Low - GLPRT_GOTCL[n] (0x00300680 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
GOTCL	31:0	0x0	RW1C	Good Octets Transmit Count Low. Counts 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.

11.2.2.17.41 Port Good Octets Transmit Count High - GLPRT_GOTCH[n] (0x00300684 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
GOTCH	15:0	0x0	RW1C	Good Octets Transmit Count High. Counts number of bytes transmitted by this port higher 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.



11.2.2.17.42 Packets Transmitted [64 Bytes] Count Low - GLPRT_PTC64L[n] (0x003006A0 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (64 bytes) count low. Combined with GLPRT_PTC64H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC64L	31:0	0x0	RW1C	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.

11.2.2.17.43 Packets Transmitted [64 Bytes] Count High - GLPRT_PTC64H[n] (0x003006A4 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (64 bytes) count low. Combined with GLPRT_PTC64L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC64H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.44 Packets Transmitted [65-127 Bytes] Count Low - GLPRT_PTC127L[n] (0x003006C0 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (65 to 127 bytes) count low. Combined with GLPRT_PTC127H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC127L	31:0	0x0	RW1C	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.

11.2.2.17.45 Packets Transmitted [65-127 Bytes] Count High - GLPRT_PTC127H[n] (0x003006C4 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (65 to 127 bytes) count low. Combined with GLPRT_PTC127L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC127H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.46 Packets Transmitted [128-255 Bytes] Count Low - GLPRT_PTC255L[n] (0x003006E0 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (128 to 255 bytes) count low. Combined with GLPRT_PTC255H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC255L	31:0	0x0	RW1C	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.

11.2.2.17.47 Packets Transmitted [128-255 Bytes] Count High - GLPRT_PTC255H[n] (0x003006E4 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (128 to 255 bytes) count low. Combined with GLPRT_PTC255L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC255H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.48 Packets Transmitted [256-511 Bytes] Count Low - GLPRT_PTC511L[n] (0x00300700 + 0x8*n, n=0...3; RW1C)

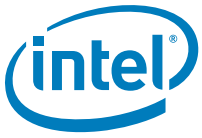
Port packets transmitted (256 to 511 bytes) count low. Combined with GLPRT_PTC511H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC511L	31:0	0x0	RW1C	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.

11.2.2.17.49 Packets Transmitted [256-511 Bytes] Count High - GLPRT_PTC511H[n] (0x00300704 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (256 to 511 bytes) count low. Combined with GLPRT_PTC511L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC511H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.



11.2.2.17.50 Packets Transmitted [512-1023 Bytes] Count Low - GLPRT_PTC1023L[n] (0x00300720 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (512 to 1023 bytes) count low. Combined with GLPRT_PTC1023H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC1023L	31:0	0x0	RW1C	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.

11.2.2.17.51 Packets Transmitted [512-1023 Bytes] Count High - GLPRT_PTC1023H[n] (0x00300724 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (512 to 1023 bytes) count low. Combined with GLPRT_PTC1023L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC1023H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.52 Packets Transmitted [1024-1522 Bytes] Count Low - GLPRT_PTC1522L[n] (0x00300740 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (1024 to 1522 bytes) count low. Combined with GLPRT_PTC1522H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC1522L	31:0	0x0	RW1C	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.

11.2.2.17.53 Packets Transmitted [1024-1522 Bytes] Count High - GLPRT_PTC1522H[n] (0x00300744 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (1024 to 1522 bytes) count low. Combined with GLPRT_PTC1522L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC1522H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.54 Packets Transmitted [1523-9522 bytes] Count Low - GLPRT_PTC9522L[n] (0x00300760 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (1523 to 9522 bytes) count low. Combined with GLPRT_PTC9522H to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC9522L	31:0	0x0	RW1C	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.

11.2.2.17.55 Packets Transmitted [1523-9522 bytes] Count High - GLPRT_PTC9522H[n] (0x00300764 + 0x8*n, n=0...3; RW1C)

Port packets transmitted (1523 to 9522 bytes) count low. Combined with GLPRT_PTC9522L to form a 64-bit register.

Field	Bit(s)	Init.	Type	Description
PTC9522H	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.56 Priority XON Transmitted Count - GLPRT_PXONTXC[n,m] (0x00300780 + 0x8*n + 0x20*m, n=0...3, m=0...7; RW1C)

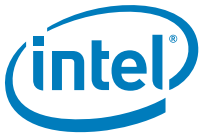
Port Priority FC XON transmitted count (array of 8 per port).

Field	Bit(s)	Init.	Type	Description
PRPXONTXC	31:0	0x0	RW1C	Priority XON Transmitted Count. Number of XON packets transmitted.

11.2.2.17.57 Priority XOFF Transmitted Count - GLPRT_PXOFFTXC[n,m] (0x00300880 + 0x8*n + 0x20*m, n=0...3, m=0...7; RW1C)

Port Priority FC XOFF transmitted count (array of 8 per port).

Field	Bit(s)	Init.	Type	Description
PRPXOFFTXCNT	31:0	0x0	RW1C	Priority XOFF Transmitted Count. Number of XOFF packets transmitted.



11.2.2.17.58 Port Link XON Transmitted Count - GLPRT_LXONTXC[n] (0x00300980 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
LXONTXC	31:0	0x0	RW1C	Link XON Transmitted Count. Number of XON packets transmitted.

11.2.2.17.59 Port Link XOFF Transmitted Count - GLPRT_LXOFFTXC[n] (0x003009A0 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
LXOFFTXC	31:0	0x0	RW1C	Link XOFF Transmitted Count. Number of XOFF packets transmitted.

11.2.2.17.60 Port Unicast Packets Transmit Count Low - GLPRT_UPTCL[n] (0x003009C0 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
UPTCL	31:0	0x0	RW1C	Unicast Packets Transmit Count Low. Counts 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.

11.2.2.17.61 Port Unicast Packets Transmit Count High - GLPRT_UPTCH[n] (0x003009C4 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
UPTCH	15:0	0x0	RW1C	Unicast Packets Transmit Count High. Counts number of unicast packets transmitted by this VSI. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.62 Port Multicast Packets Transmit Count Low - GLPRT_MPTCL[n] (0x003009E0 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
MPTCL	31:0	0x0	RW1C	Multicast Packets Transmit Count Low. Counts 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.

11.2.2.17.63 Port Multicast Packets Transmit Count High - GLPRT_MPTCH[n] (0x003009E4 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
MPTCH	15:0	0x0	RW1C	Multicast Packets Transmit Count High. Counts number of multicast packets transmitted by this VSI. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.64 Port Broadcast Packets Transmit Count Low - GLPRT_BPTCL[n] (0x00300A00 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
BPTCL	31:0	0x0	RW1C	Broadcast Packet Transmit Count Low. Counts 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.

11.2.2.17.65 Port Broadcast Packets Transmit Count High - GLPRT_BPTCH[n] (0x00300A04 + 0x8*n, n=0...3; RW1C)

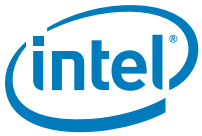
Field	Bit(s)	Init.	Type	Description
BPTCH	15:0	0x0	RW1C	Broadcast Packet Transmit Count High. Counts number of broadcast packets transmitted by this Port. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.66 Transmit Discard on Link Down - GLPRT_TDOLD[n] (0x00300A20 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
GLPRT_TDOLD	31:0	0x0	RW1C	Transmit Discard On Link Down. Packets discarded at the port because the link was down.

11.2.2.17.67 VSI Received Discard Packet Count - GLV_RDPC[n] (0x00310000 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
RDPC	31:0	0x0	RW1C	Received Discard Packet Count. Counts (per VSI) packets that were dropped due to no descriptors in host queue. For EMP VSI's it counts dropped packets due to no EMP buffer space.



11.2.2.17.68 FCoE Last Error Count - GL_FCOELAST[n] (0x00314000 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOELAST	31:0	0x0	RW1C	FCoE Last Error Count. Number of packets received to valid FCoE contexts while their user buffers are exhausted.

11.2.2.17.69 FCoE DDP Count - GL_FCOEDDPC[n] (0x00314480 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOEDDPC	31:0	0x0	RW1C	FCoE DDP Count. Number of FCoE packets received using DDP. ***Note:*** Packets with DIF errors that were processed by the DDP context are counted as well.

11.2.2.17.70 FCoE CRC Error Count - GL_FCOECRC[n] (0x00314D80 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOECRC	31:0	0x0	RW1C	FCoE CRC Error Count. Counts packets with good Ethernet CRC but bad FC CRC.

11.2.2.17.71 FCoE Packets Received Count - GL_FCOEPRC[n] (0x00315200 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOEPRC	31:0	0x0	RW1C	FCoE Packets Received Count. Number of FCoE packets posted to the host. In nominal operation (no save bad frames) it equals to the number of good packets.

11.2.2.17.72 Receive Error Counter 1 Low - GL_RXERR1_L[n] (0x00318000 + 0x8*n, n=0...143; RW1C)

Number of SCSI blocks received with DIF tags.

Field	Bit(s)	Init.	Type	Description
FCOEDIFRC	31:0	0x0	RW1C	Counts dropped packets due to one of the following exceptions (generated internally by the RLAN): • Packet size is larger than RXMAX of the queue. • FCoE packet received to a queue with inactive FCENA. • Internal Receive queue context error (could be a result of a reset in progress). • Receive descriptor Unsupported Request on the PCI or internal Dummy completion (the Dummy completion is a result of a reset in progress).

11.2.2.17.73 FCoE DIF Receive Error Count - GL_FCOEDIFEC[n] (0x00318480 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOEDIFRC	31:0	0x0	RW1C	FCoE DIF Receive Error Count. Counts number of SCSI blocks received with DIF errors.

11.2.2.17.74 FCoE Valid DIX Tag to Host Count - GL_RXERR2_L[n] (0x0031C000 + 0x8*n, n=0...143; RW1C)

Number of SCSI Blocks with added DIX tag from NIC to host.

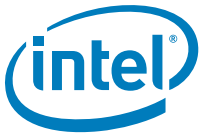
Field	Bit(s)	Init.	Type	Description
FCOEDIXAC	31:0	0x0	RW1C	FCoE Valid DIX Tag to Host Count. Counts dropped packets due to one of the following exceptions (internal indication from RCU or RCB): • Packets directed to invalid receive queues. • Packets directed to disabled receive queues. • Packets dropped by the switch filters (these packets are also counted by the switch statistics). • Packets dropped due to MAC errors or FC CRC errors. • Packets dropped by the FD filter. • Packets dropped due to VM reset, VF reset or PF reset.

11.2.2.17.75 FCoE DWord Received Count Low - GL_FCOEDWRCL[n] (0x00320000 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOEDWRCL	31:0	0x0	RW1C	FCoE DWord Received Count Low. Counts number of DWords in good received packets with no Ethernet CRC or FC CRC errors. The counter relates to FCoE packets starting at the FC header up to and including the FC CRC (it excludes the Ethernet encapsulation). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

11.2.2.17.76 FCoE DWord Received Count High - GL_FCOEDWRCH[n] (0x00320004 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOEDWRCH	15:0	0x0	RW1C	FCoE DWord Received Count High. Counts number of DWords in good received packets with no Ethernet CRC or FC CRC errors. The counter relates to FCoE packets starting at the FC header up to and including the FC CRC (it excludes the Ethernet encapsulation). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.



11.2.2.17.77 FCoE Rx Packets Dropped Count - GL_FCOERPDC[n] (0x00324000 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOERPDC	31:0	0x0	RW1C	FCoE Received Packets Dropped Count. Number of FCoE Rx packets dropped (for any reason).

11.2.2.17.78 VSI Good Octets Transmit Count Low - GLV_GOTCL[n] (0x00328000 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
GOTCL	31:0	0x0	RW1C	Good Octets Transmit Count Low. Counts 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.

11.2.2.17.79 VSI Good Octets Transmit Count High - GLV_GOTCH[n] (0x00328004 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
GOTCH	15:0	0x0	RW1C	Good Octets Transmit Count High. Counts number of bytes transmitted by this VSI. Higher 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.80 Switch Good Octets Transmit Count Low - GLSW_GOTCL[n] (0x0032C000 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
GOTCL	31:0	0x0	RW1C	Good Octets Transmit Count Low. Counts 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.

11.2.2.17.81 Switch Good Octets Transmit Count High - GLSW_GOTCH[n] (0x0032C004 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
GOTCH	15:0	0x0	RW1C	Good Octets Transmit Count High. Counts number of bytes transmitted. Higher 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.82 VEB VLAN Transmit Byte Count Low - GLVEBVL_GOTCL[n] (0x00330000 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	Description
VLBCL	31:0	0x0	RW1C	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.

11.2.2.17.83 VEB VLAN Trasmint Byte Count High - GLVEBVL_GOTCH[n] (0x00330004 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	Description
VLBCH	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.84 VEB TC Transmit Byte Count Low - GLVEBTC_TBCL[n,m] (0x00334000 + 0x8*n + 0x40*m, n=0...7, m=0...15; RW1C)

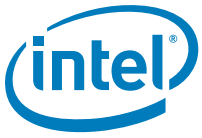
Field	Bit(s)	Init.	Type	Description
TCBCL	31:0	0x0	RW1C	VEB per TC Byte Count Low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

11.2.2.17.85 VEB TC Transmit Byte Count High - GLVEBTC_TBCH[n,m] (0x00334004 + 0x8*n + 0x40*m, n=0...7, m=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
TCBCH	15:0	0x0	RW1C	VEB per TC 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	31:16	0x0	RSV	Reserved.

11.2.2.17.86 VEB TC Transmit Packet Count Low - GLVEBTC_TPCL[n,m] (0x00338000 + 0x8*n + 0x40*m, n=0...7, m=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
TCPCL	31:0	0x0	RW1C	VEB per TC Packet Count Low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.



11.2.2.17.87 VEB TC Transmit Packet Count High - GLVEBTC_TPCH[n,m] (0x00338004 + 0x8*n + 0x40*m, n=0...7, m=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
TCPCH	15:0	0x0	RW1C	VEB per TC 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	31:16	0x0	RSV	Reserved.

11.2.2.17.88 VSI Unicast Packets Transmit Count Low - GLV_UPTCL[n] (0x0033C000 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
UPTCL	31:0	0x0	RW1C	Unicast Packets Transmit Count Low. Counts 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.

11.2.2.17.89 VSI Unicast Packets Transmit Count High - GLV_UPTCH[n] (0x0033C004 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
UPTCH	15:0	0x0	RW1C	Unicast Packets Transmit Count High. Counts number of unicast packets transmitted by this VSI. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.90 VSI Multicast Packets Transmit Count Low - GLV_MPTCL[n] (0x0033CC00 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
MPTCL	31:0	0x0	RW1C	Multicast Packets Transmit Count Low. Counts 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.

11.2.2.17.91 VSI Multicast Packets Transmit Count High - GLV_MPTCH[n] (0x0033CC04 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
MPTCH	15:0	0x0	RW1C	Multicast Packets Transmit Count High. Counts number of multicast packets transmitted by this VSI. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.92 VSI Broadcast Packets Transmit Count Low - GLV_BPTCL[n] (0x0033D800 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
BPTCL	31:0	0x0	RW1C	Broadcast Packets Transmit Count Low. Counts 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.

11.2.2.17.93 VSI Broadcast Packets Transmit Count High - GLV_BPTCH[n] (0x0033D804 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
BPTCH	15:0	0x0	RW1C	Broadcast Packets Transmit Count High. Counts number of broadcast packets transmitted by this VSI. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.94 Switch Unicast Packets Transmit Count Low - GLSW_UPTCL[n] (0x00340000 + 0x8*n, n=0...15; RW1C)

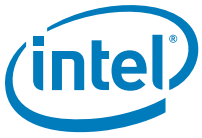
Field	Bit(s)	Init.	Type	Description
UPTCL	31:0	0x0	RW1C	Unicast Packets Transmit Count Low. Counts number of unicast packets transmitted. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

11.2.2.17.95 Switch Unicast Packets Transmit Count High - GLSW_UPTCH[n] (0x00340004 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
UPTCH	15:0	0x0	RW1C	Unicast Packets Transmit Count High. Counts number of unicast packets transmitted. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.96 Switch Multicast Packets Transmit Count Low - GLSW_MPTCL[n] (0x00340080 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
MPTCL	31:0	0x0	RW1C	Multicast Packets Transmit Count Low. Counts 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.



11.2.2.17.97 Switch Multicast Packets Transmit Count High - GLSW_MPTCH[n] (0x00340084 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
MPTCH	15:0	0x0	RW1C	Multicast Packets Transmit Count High. Counts number of multicast packets transmitted. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.98 Switch Broadcast Packets Transmit Count Low - GLSW_BPTCL[n] (0x00340100 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
BPTCL	31:0	0x0	RW1C	Broadcast Packets Transmit Count Low. Counts 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.

11.2.2.17.99 Switch Broadcast Packets Transmit Count High - GLSW_BPTCH[n] (0x00340104 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
BPTCH	15:0	0x0	RW1C	Broadcast Packets Transmit Count High. Counts number of broadcast packets transmitted. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.100 VSI Transmit Error Packet Count - GLV_TEPC[VSI] (0x00344000 + 0x4*VSI, VSI=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
TEPC	31:0	0x0	RW1C	Transmit Error Packet Count.

11.2.2.17.101 FCoE Packets Transmitted Count - GL_FCOEPTC[n] (0x00344C00 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOEPTC	31:0	0x0	RW1C	FCoE Packets Transmitted Count. Number of FCoE packets transmitted.

11.2.2.17.102 Switch Transmit Packets Discarded Count - GLSW_TDPC[n] (0x00348000 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
TDPC	31:0	0x0	RW1C	Transmit Discard Packet Count.

11.2.2.17.103 FCoE DWord Transmitted Count Low - GL_FCOEDWTCL[n] (0x00348080 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOEDWTCL	31:0	0x0	RW1C	FCoE DWord Transmitted Count Low. Number of DWords count in transmitted packets. The counter relates to FCoE packets starting at the FC header up to and including the FC CRC (it excludes the Ethernet encapsulation). The low and high registers are part of a 64 bit register and are read using 64-bit read accesses only.

11.2.2.17.104 FCoE DWord Transmitted Count High - GL_FCOEDWTCH[n] (0x00348084 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOEDWTCH	15:0	0x0	RW1C	FCoE DWord Transmitted Count Low. Number of DWords count in transmitted packets. The counter relates to FCoE packets starting at the FC header up to and including the FC CRC (it excludes the Ethernet encapsulation). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.105 FCoE Bad DIX Error from Host Count - GL_FCOEDIXEC[n] (0x0034C000 + 0x8*n, n=0...143; RW1C)

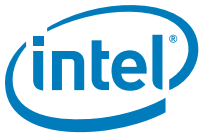
Field	Bit(s)	Init.	Type	Description
FCOEDIXEC	31:0	0x0	RW1C	FCoE DIX Error Count. Counts Number of SCSI Blocks with DIX tag errors sent from host to NIC.

11.2.2.17.106 FCoE Valid DIX Tag from Host Count - GL_FCOEDIXVC[n] (0x00350000 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOEDIXVC	31:0	0x0	RW1C	FCoE DIX Valid Count. Counts Number of SCSI Blocks with Valid DIX tags from host to NIC.

11.2.2.17.107 FCoE DIF Block Transmit Count Low - GL_FCOEDIFTCL[n] (0x00354000 + 0x8*n, n=0...143; RW1C)

Field	Bit(s)	Init.	Type	Description
FCOEDIFTC	31:0	0x0	RW1C	FCoE DIF Transmit Count. Counts Number of SCSI blocks transmitted with DIF tags.



11.2.2.17.108 VSI Good Octets Received Count Low - GLV_GORCL[n] (0x00358000 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
GORCL	31:0	0x0	RW1C	Good Octets Received Count Low. Counts number of bytes received 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.

11.2.2.17.109 VSI Good Octets Received Count High - GLV_GORCH[n] (0x00358004 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
GORCH	15:0	0x0	RW1C	Good Octets Received Count High. Counts number of bytes received by this VSI. Higher 16 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.110 Switch Good Octets Received Count Low - GLSW_GORCL[n] (0x0035C000 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
GORCL	31:0	0x0	RW1C	Good Octets Received Count Low. Counts 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.

11.2.2.17.111 Switch Good Octets Received Count High - GLSW_GORCH[n] (0x0035C004 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
GORCH	15:0	0x0	RW1C	Good Octets Received Count High. Counts number of bytes received. Higher 16 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.112 VEB VLAN Receive Byte Count Low - GLVEBVL_GORCL[n] (0x00360000 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	Description
VLBCL	31:0	0x0	RW1C	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.

11.2.2.17.113 VEB VLAN Receive Byte Count High - GLVEBVL_GORCH[n] (0x00360004 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	Description
VLBCH	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.114 VEB TC Receive Byte Count Low - GLVEBTC_RBCL[n,m] (0x00364000 + 0x8*n + 0x40*m, n=0...7, m=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
TCBCL	31:0	0x0	RW1C	VEB per TC Byte Count Low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

11.2.2.17.115 VEB TC Receive Byte Count High - GLVEBTC_RBCH[n,m] (0x00364004 + 0x8*n + 0x40*m, n=0...7, m=0...15; RW1C)

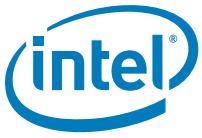
Field	Bit(s)	Init.	Type	Description
TCBCH	15:0	0x0	RW1C	VEB per TC Byte Count High. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.116 VEB TC Receive Packet Count Low - GLVEBTC_RPCL[n,m] (0x00368000 + 0x8*n + 0x40*m, n=0...7, m=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
TCPCL	31:0	0x0	RW1C	VEB per TC Packet Count Low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

11.2.2.17.117 VEB TC Receive Packet Count High - GLVEBTC_RPCH[n,m] (0x00368004 + 0x8*n + 0x40*m, n=0...7, m=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
TCPCH	15:0	0x0	RW1C	VEB per TC 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	31:16	0x0	RSV	Reserved.



11.2.2.17.118 VSI Unicast Packets Received Count Low - GLV_UPRCL[n] (0x0036C000 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
UPRCL	31:0	0x0	RW1C	Unicast Packets Received Count Low. Counts number of unicast packets received 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.

11.2.2.17.119 VSI Unicast Packets Received Count High - GLV_UPRCH[n] (0x0036C004 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
UPRCH	15:0	0x0	RW1C	Unicast Packets Received Count High. Counts number of unicast packets received by this VSI. Upper 32 bits. 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	Reserved.

11.2.2.17.120 VSI Multicast Packets Received Count Low - GLV_MPRCL[n] (0x0036CC00 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
MPRCL	31:0	0x0	RW1C	Multicast Packets Received Count Low. Counts number of multicast packets received 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.

11.2.2.17.121 VSI Multicast Packets Received Count High - GLV_MPRCH[n] (0x0036CC04 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
MPRCH	15:0	0x0	RW1C	Multicast Packets Received Count High. Counts number of multicast packets received by this VSI. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.122 VSI Broadcast Packets Received Count Low - GLV_BPRCL[n] (0x0036D800 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
BPRCL	31:0	0x0	RW1C	Broadcast Packets Received Count Low. Counts number of broadcast packets received 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.

11.2.2.17.123 VSI Broadcast Packets Received Count High - GLV_BPRCH[n] (0x0036D804 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
BPRCH	15:0	0x0	RW1C	Broadcast Packets Received Count High. Counts number of broadcast packets received by this VSI. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.124 VSI Received Unknown Packet Protocol Count - GLV_RUPP[n] (0x0036E400 + 0x8*n, n=0...383; RW1C)

Field	Bit(s)	Init.	Type	Description
RUPP	31:0	0x0	RW1C	Received Unknown Packet Protocol count. Receive Packets dropped because of an a unknown protocol or no forward destination.

11.2.2.17.125 Switch Unicast Packets Received Count Low - GLSW_UPRCL[n] (0x00370000 + 0x8*n, n=0...15; RW1C)

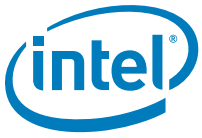
Field	Bit(s)	Init.	Type	Description
UPRCL	31:0	0x0	RW1C	Unicast Packets Received Count Low. Counts number of unicast packets received. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

11.2.2.17.126 Switch Unicast Packets Received Count High - GLSW_UPRCH[n] (0x00370004 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
UPRCH	15:0	0x0	RW1C	Unicast Packets Received Count High. Counts number of unicast packets received. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.127 Switch Multicast Packets Received Count Low - GLSW_MPRCL[n] (0x00370080 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
MPRCL	31:0	0x0	RW1C	Multicast Packets Received Count Low. Counts 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.



11.2.2.17.128 Switch Multicast Packets Received Count High - GLSW_MPRCH[n] (0x00370084 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
MPRCH	15:0	0x0	RW1C	Multicast Packets Received Count Low. Counts number of multicast packets received. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.129 Switch Broadcast Packets Received Count Low - GLSW_BPRCL[n] (0x00370100 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
BPRCL	31:0	0x0	RW1C	Broadcast Packets Received Count Low. Counts 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.

11.2.2.17.130 Switch Broadcast Packets Received Count High - GLSW_BPRCH[n] (0x00370104 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
BPRCH	15:0	0x0	RW1C	Broadcast Packets Received Count High. Counts number of broadcast packets received. Upper 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.131 Switch Received Unknown Packet Protocol Count - GLSW_RUPP[n] (0x00370180 + 0x8*n, n=0...15; RW1C)

Field	Bit(s)	Init.	Type	Description
RUPP	31:0	0x0	RW1C	Received Unknown Packet Protocol count. Receive Packets dropped because of an a unknown protocol or no forward destination.

11.2.2.17.132 VEB VLAN Unicast Packet Count Low - GLVEBVL_UPCL[n] (0x00374000 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	Description
VLUPCL	31:0	0x0	RW1C	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.

11.2.2.17.133 VEB VLAN Unicast Packet Count High - GLVEBVL_UPCH[n] (0x00374004 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	Description
VLUPCH	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.

11.2.2.17.134 VEB VLAN Multicast Packet Count Low - GLVEBVL_MPCL[n] (0x00374400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	Description
VLMPCL	31:0	0x0	RW1C	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.

11.2.2.17.135 VEB VLAN Multicast Packet Count High - GLVEBVL_MPCH[n] (0x00374404 + 0x8*n, n=0...127; RW1C)

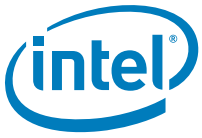
Field	Bit(s)	Init.	Type	Description
VLMPCH	15:0	0x0	RW1C	VEB per VLAN Multicast Packet Count High. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.17.136 VEB VLAN Broadcast Packet Count Low - GLVEBVL_BPCL[n] (0x00374800 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	Description
VLBPCL	31:0	0x0	RW1C	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.

11.2.2.17.137 VEB VLAN Broadcast Packet Count High - GLVEBVL_BPCH[n] (0x00374804 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	Description
VLBPCH	15:0	0x0	RW1C	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	31:16	0x0	RSV	Reserved.



11.2.2.17.138 Port Transmit Packets Discarded Count - GLPRT_TDPC[n] (0x00375400 + 0x8*n, n=0...3; RW1C)

Field	Bit(s)	Init.	Type	Description
TDPC	31:0	0x0	RW1C	Transmit Discard Packet Count. Packets that were discarded on transmit while link was down.

11.2.2.18 PF - LAN Transmit Receive Registers

11.2.2.18.1 Global TSO TCP Mask First - GLLAN_TSOMSK_F (0x000442D8; RW)

Field	Bit(s)	Init.	Type	Description
TCPMSKF	11:0	0x0	RW	TCP Flags mask for the first segment in the TSO. Any bit set to one in the *TCPMSK* 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. Default setting: clear the FIN and PSH flags.
RSVD	31:12	0x0	RSV	Reserved.

11.2.2.18.2 Global TSO TCP Mask Middle - GLLAN_TSOMSK_M (0x000442DC; RW)

Field	Bit(s)	Init.	Type	Description
TCPMSKM	11:0	0x0	RW	TCP Flags mask for the middle segments in the TSO. See TCPMSKF for the impact of each bit in the field. Default setting: clear the FIN, PSH and CWR flags.
RSVD	31:12	0x0	RSV	Reserved.

11.2.2.18.3 Global TSO TCP Mask Last - GLLAN_TSOMSK_L (0x000442E0; RW)

Field	Bit(s)	Init.	Type	Description
TCPMSKL	11:0	0x0	RW	TCP Flags mask for the last segment in the TSO. See TCPMSKF for the impact of each bit in the field. By default clear the CWR flag.
RSVD	31:12	0x0	RSV	Reserved.

11.2.2.18.4 Receive Processing Block Control - GL_RDPU_CNTRL (0x00051060; RW)

Field	Bit(s)	Init.	Type	Description
RX_PAD_EN	0	1b	RW	Pad Rx packets with zeros that do not include CRC or the CRC is stripped to be at least 60 bytes long. Padding and CRC stripping for EMP packets is always done regardless of the setting of this flag.
ECO	31:1	0x0	RW	Reserved.

11.2.2.18.5 VF PF Queue Mapping Table - VPLAN_QTABLE[n,VF] (0x00070000 + 0x400*n + 0x4*VF, n=0...15, VF=0...127; RW)

This register affects the VF, but is exposed only to the parent PF.

Field	Bit(s)	Init.	Type	Description
QINDEX	10:0	0x7FF	RW	Defines the index of VF queue 'n' in the PF queues space (while 'n' is the register index). Setting the *QINDEX* to 0x7FF means that the queue is not valid for the VF.
RSVD	31:11	0x0	RSV	Reserved.

11.2.2.18.6 VF LAN Enablement - VPLAN_MAPENA[VF] (0x00074000 + 0x4*VF, VF=0...127; RW)

GLLAN_VFENA[n] register is initialized by VFR of VF 'n'.

Field	Bit(s)	Init.	Type	Description
TXRX_ENA	0	0b	RW	The VFLAN_QTABLE for the VF is enabled only when the *TXRX_ENA* flag is set.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.18.7 Global Transmit Queue Head - QTX_HEAD[Q] (0x000E4000 + 0x4*Q, Q=0...1535; RW)

Field	Bit(s)	Init.	Type	Description
RSVD	31:0	0x0	RSV	Reserved.

11.2.2.18.8 Global Transmit Pre Queue Disable - GLLAN_TXPRE_QDIS[n] (0x000E6500 + 0x4*n, n=0...11; RW)

Queue Enable Register.

Field	Bit(s)	Init.	Type	Description
QINDX	10:0	0x0	RW	Queue Index. Absolute index in the device space. Each register 'n' covers 128*n-128*n-1 queues. Software is expected to use the matched register when accessing a specific queue.
RSVD	29:11	0x0	RSV	Reserved.
SET_QDIS	30	0b	RW	Set Queue Disable. Setting this flag to 1b sets an internal QDIS flag of the transmit queue (that is indicated by *QINDX* field in this register). As a result, any accumulated quanta of the queue are invalidated which is a needed step before the queue is disabled. Setting the *SET_QDIS* flag is mutually exclusive with the *CLEAR_QDIS* flag.
CLEAR_QDIS	31	0b	RW	Clear Queue Disable. Setting this flag to 1b clears an internal QDIS flag of the transmit queue (that is indicated by *QINDX* field in this register). This step should be made before the queue is enabled. Setting the *CLEAR_QDIS* flag is mutually exclusive with the *SET_QDIS* flag.

11.2.2.18.9 Global Transmit Queue Enable - QTX_ENA[Q] (0x00100000 + 0x4*Q, Q=0...1535; RW)

Field	Bit(s)	Init.	Type	Description
QENA_REQ	0	0b	RW	Transmit Queue Enable Request. If this bit is set, 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* before it is permitted to revert the state of the *QENA_REQ* once again.
FAST_QDIS	1	0b	RW	Fast Queue Disable. See “Fast Transmit Queue Disable Flow” section for the usage of this flag.
QENA_STAT	2	0b	RO	Transmit Queue Enable Status indication. 0b = The queue is inactive. 1b = Indicates that the queue is active.
RSVD	31:3	0x0	RSV	Reserved.

11.2.2.18.10 Global Transmit Queue Control - QTX_CTL[Q] (0x00104000 + 0x4*Q, Q=0...1535; RW)

Field	Bit(s)	Init.	Type	Description
PFVF_Q	1:0	00b	RW	Queue association to PF or VF as follows: 00b = VF Queue 01b = VM Queue 10b = PF Queue 11b = Reserved
PF_INDX	5:2	0x0	RW	PF Index between 0 and 15.
RSVD	6	0b	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
VFVM_INDX	15:7	0x0	RW	VF / VM index should be programmed per PFVF_Q setting as follows: **PFVF_Q setting** — **Matched VF / VM index programming** 00b = VF queue — The absolute VF index (between 0-127) 01b = VM queue — The absolute VSI index (between 0-383) Else = PF or EMP queue — Must be set to zero
RSVD	31:16	0x0	RSV	Reserved

11.2.2.18.11 Global Transmit Queue Tail - QTX_TAIL[Q] (0x00108000 + 0x4*Q, Q=0...1535; RW)

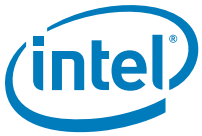
Field	Bit(s)	Init.	Type	Description
TAIL	12:0	0x0	RW	Transmit Tail. Defines the first descriptor that the software prepares for the hardware (it is the last valid descriptor plus one). The Tail is a relative descriptor index to the beginning of the transmit descriptor ring.
RSVD	31:13	0x0	RSV	Reserved.

11.2.2.18.12 Global Receive Queue Enable - QRX_ENA[Q] (0x00120000 + 0x4*Q, Q=0...1535; RW)

Field	Bit(s)	Init.	Type	Description
QENA_REQ	0	0b	RW	Receive Queue Enable Request. If this bit is set, 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* before it is permitted to revert the state of the *QENA_REQ* once again.
FAST_QDIS	1	0b	RW1C	Fast Queue Disable. See “Fast Receive Queue Disable Flow” section for the usage of this flag. This flag is auto-cleared by the hardware.
QENA_STAT	2	0b	RO	Receive Queue Enable Status indication. 0b = The queue is inactive. 1b = Indicates that the queue is active.
RSVD	31:3	0x0	RSV	Reserved.

11.2.2.18.13 Global Receive Queue Tail - QRX_TAIL[Q] (0x00128000 + 0x4*Q, Q=0...1535; RW)

Field	Bit(s)	Init.	Type	Description
TAIL	12:0	0x0	RW	Receive 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.
RSVD	31:13	0x0	RSV	Reserved.



11.2.2.18.14 Global RLAN Control 0 - GLLAN_RCTL_0 (0x0012A500; RW1C)

Field	Bit(s)	Init.	Type	Description
PXE_MODE	0	1b	RW1C	When *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.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.18.15 PF Queue Allocation - PFLAN_QALLOC (0x001C0400; RO)

These registers define the LAN queue pairs allocation to the PFs.

Field	Bit(s)	Init.	Type	Description
FIRSTQ	10:0	0x0	RW	The first LAN Queue pair allocated to this PF. Valid only if the VALID flag is set. Valid values are 0-1535.
RSVD	15:11	0x0	RSV	Reserved.
LASTQ	26:16	0x0	RW	The last LAN Queue pair allocated to this PF. Valid only if the VALID flag is set. Valid values are 0-1535.
RSVD	30:27	0x0	RSV	Reserved.
VALID	31	0b	RW	Indicates that queues are allocated to this PF. For any active PF this flag must be set.

11.2.2.18.16 VSI Receive Queue Mapping Table - VSILAN_QTABLE[n,VSI] (0x00200000 + 0x800*n + 0x4*VSI, n=0...7, VSI=0...383; RO)

Field	Bit(s)	Init.	Type	Description
QINDEX_0	10:0	0x0	RW	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 *QINDEX* plus PFLAN_QALLOC.*FIRSTQ* of the parent PF. Setting the *QINDEX* to 0x7FF means the queue is not valid.
RSVD	15:11	0x0	RSV	Reserved
QINDEX_1	26:16	0x0	RW	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 the queue is not valid.
RSVD	31:27	0x0	RSV	Reserved

11.2.2.18.17 VSI Queue Control - VSILAN_QBASE[VSI] (0x0020C800 + 0x4*VSI, VSI=0...383; RO)

Field	Bit(s)	Init.	Type	Description
VSIBASE	10:0	0x0	RW	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	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*
RSVD	31:12	0x0	RSV	Reserved.

11.2.2.19 PF - Rx Filters Registers

11.2.2.19.1 VF Queue Filter Control - VPQF_CTL[VF] (0x001C0000 + 0x4*VF, VF=0...127; RW)

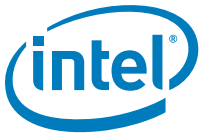
This register affects the VF, but is exposed only to the parent PF.

Field	Bit(s)	Init.	Type	Description
PEHSIZE	4:0	0x0	RW	Defines the number of "buckets" of the PE Quad Hash filter table for the VF, defined in power of 2 equals to 1K x 2 ** *PEHSIZE*. The *PEHSIZE* can have any value between 0 and 10. *PEHSIZE* = 0, 1,... 10 is equivalent to 1K, 2K, 4K... 1M buckets.
PEDSIZE	9:5	0x0	RW	Defines the number of PE Quad Hash contexts for the VF, defined in power of 2 equals to 0.5K x 2 ** *PEDSIZE*. The *PEDSIZE* can have any value between 0 and 9. *PEHSIZE* = 0, 1,... 9 is equivalent to 0.5K, 1K, 2K, 4K... 256K contexts.
FCHSIZE	13:10	0x0	RW	Defines the number of "buckets" of the FCOE filter table for the VF. This parameter must be set to the same value of the *VFFCHSIZE* for all VFs of the PF.
FCDSIZE	15:14	0x0	RW	Defines the number of FCOE DDP contexts for the VF. This parameter must be set to the same value of the *VFFCDSIZE* for all VFs of the PF.
RSVD	31:16	0x0	RSV	Reserved.

11.2.2.19.2 PF Queue Filter Control 0 - PFQF_CTL_0 (0x001C0AC0; RW)

Settings in this register allow the hardware auto-clear the FCoE DDP counter and internal table pointers, as well as the internal PE Quad Hash context counter and table pointers of the function.

Field	Bit(s)	Init.	Type	Description
PEHSIZE	4:0	0x0	RW	Defines the number of "buckets" of the PE Quad Hash filter table for the PF, defined in power of 2 equals to 1K x 2 ** *PEHSIZE*. The *PEHSIZE* can have any value between 0 and 10. *PEHSIZE* = 0, 1,... 10 is equivalent to 1K, 2K, 4K... 1M buckets.
PEDSIZE	9:5	0x0	RW	Defines the number of PE Quad Hash contexts for the PF, defined in power of 2 equals to 0.5K x 2 ** *PEDSIZE*. The *PEDSIZE* can have any value between 0 and 9. *PEHSIZE* = 0, 1,... 9 is equivalent to 0.5K, 1K, 2K, 4K... 256K contexts.



Field	Bit(s)	Init.	Type	Description
PFFCHSIZE	13:10	0x0	RW	Defines the number of "buckets" of the FCOE filter table for the PF, defined in power of 2 equals to 1K x 2 ** *PFFCHSIZE*. The *PFFCHSIZE* can be set to either 4K, 16K or 32K buckets.
PFFCDSIZE	15:14	00b	RW	Defines the number of FCOE DDP contexts for the PF, defined in power of 2 equals to 0.5K x 2 ** *PFFCDSIZE*. The *PFFCDSIZE* can be set to 0, 1 or 2, *PFFCDSIZE* = 0, 1, 2 is equivalent to 0.5K, 2K or 4K DDP contexts.
HASHLUTSIZE	16	0b	RW	Defines the size of the Hash LUT as follows: 0b = 128 1b = 512
FD_ENA	17	0b	RW	Enable Flow Director filters for the PF and its VFs.
ETYPE_ENA	18	0b	RW	Enable Ethertype queue filters for the PF and its VFs.
MACVLAN_ENA	19	0b	RW	Enable MAC/VLAN queue filters for the PF and its VFs.
VFFCHSIZE	23:20	0x0	RW	Defines the number of "buckets" of the FCOE filter table for the VFs of this PF, defined in power of 2 equals to 1K x 2 ** *VFFCHSIZE*. The *VFFCHSIZE* can be set to either 4K or 16K buckets.
VFFCDSIZE	25:24	00b	RW	Defines the number of FCOE DDP contexts for the VFs of this PF, defined in power of 2 equals to 0.5K x 2 ** *VFFCDSIZE*. The *VFFCDSIZE* can be set to either 0 or 2, *VFFCDSIZE* = 0 or 2 is equivalent to 0.5K or 2K DDP contexts.
RSVD	31:26	0x0	RSV	Reserved.

11.2.2.19.3 VSI Receive Traffic Class Queues - VSIQF_TCREGION[n,VSI] (0x00206000 + 0x800*n + 0x4*VSI, n=0...3, VSI=0...383; RO)

Field	Bit(s)	Init.	Type	Description
TC_OFFSET	8:0	0x0	RW	Relative queue index to the beginning of the VSI that the TC '2*n' uses, where 'n' is the register index. ***Note:*** The hardware does not check if the queue index exceeds the VSI range. Therefore, it is the responsibility of the PF software to ensure that *TC_SIZE* + *TC_OFFSET* does not exceeds the VSI range.
TC_SIZE	11:9	000b	RW	The number of receive queues allocated to TC '2*n' for the VSI, where 'n' is the register index. *TC_SIZE* can have any value of 0 to 6, which means any of the following respective number of allocated queues: 1; 2; 4; 8; 16; 32; 64. The value 7 is reserved.
RSVD	15:12	0x0	RSV	Reserved.
TC_OFFSET2	24:16	0x0	RW	Relative queue index to the beginning of the VSI that the TC '2*n+1' uses, where 'n' is the register index. ***Note:*** The hardware does not check if the queue index exceeds the VSI range. Therefore, it is the responsibility of the PF software to ensure that *TC_SIZE* + *TC_OFFSET* does not exceeds the VSI range.
TC_SIZE2	27:25	000b	RW	The number of receive queues allocated to TC '2*n+1' for the VSI, where 'n' is the register index. *TC_SIZE2* can have any value of 0 to 6, which means any of the following respective number of allocated queues: 1; 2; 4; 8; 16; 32; 64. The value 7 is reserved.
RSVD	31:28	0x0	RSV	Reserved.

11.2.2.19.4 VSI Queue Filter Control - VSIQF_CTL[VSI] (0x0020D800 + 0x4*VSI, VSI=0...383; RO)

Field	Bit(s)	Init.	Type	Description
FCOE_ENA	0	0b	RW	Enable FCoE filter context for the VSI. When this flag is cleared, the hardware does not look for matched FCoE filter. ***Note:*** This flag can be enabled only for PF VSIs. Software should not enable it for VF or VM VSIs.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.19.5 VF Queue Filter Hash LUT - VFQF_HLUT[n,VF] (0x00220000 + 0x400*n + 0x4*VF, n=0...15, VF=0...127; RW)

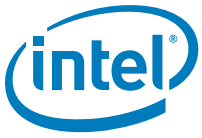
Field	Bit(s)	Init.	Type	Description
LUT0	3:0	0x0	RW	Hash redirection LUT entry 4 x 'n', where 'n' is the register index.
RSVD	7:4	0x0	RSV	Reserved.
LUT1	11:8	0x0	RW	Hash redirection LUT entry 4 x 'n' + 1, where 'n' is the register index.
RSVD	15:12	0x0	RSV	Reserved.
LUT2	19:16	0x0	RW	Hash redirection LUT entry 4 x 'n' + 2, where 'n' is the register index.
RSVD	23:20	0x0	RSV	Reserved.
LUT3	27:24	0x0	RW	Hash redirection LUT entry 4 x 'n' + 3, where 'n' is the register index.
RSVD	31:28	0x0	RSV	Reserved.

11.2.2.19.6 VF Queue Filter Hash Key - VFQF_HKEY[n,VF] (0x00228000 + 0x400*n + 0x4*VF, n=0...12, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
KEY_0	7:0	0x0	RW	Toeplitz key byte '4*n+0', where 'n' is the register index.
KEY_1	15:8	0x0	RW	Toeplitz key byte '4*n+1', where 'n' is the register index.
KEY_2	23:16	0x0	RW	Toeplitz key byte '4*n+2', where 'n' is the register index.
KEY_3	31:24	0x0	RW	Toeplitz key byte '4*n+3', where 'n' is the register index.

11.2.2.19.7 VF Queue Filter Hash Region of Queues - VFQF_HREGION[n,VF] (0x0022E000 + 0x400*n + 0x4*VF, n=0...7, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
OVERRIDE_ENA_0	0	0b	RW	Override Traffic Class Region for packet type '8 x n', where 'n' is the register index.
REGION_0	3:1	000b	RW	Receive queue region for packet type '8 x n', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_1	4	0b	RW	Override Traffic Class Region for packet type '8 x n + 1', where 'n' is the register index.
REGION_1	7:5	000b	RW	Receive queue region for packet type '8 x n + 1', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.



Field	Bit(s)	Init.	Type	Description
OVERRIDE_ENA_2	8	0b	RW	Override Traffic Class Region for packet type '8 x n + 2', where 'n' is the register index.
REGION_2	11:9	000b	RW	Receive queue region for packet type '8 x n + 2', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_3	12	0b	RW	Override Traffic Class Region for packet type '8 x n + 3', where 'n' is the register index.
REGION_3	15:13	000b	RW	Receive queue region for packet type '8 x n + 3', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_4	16	0b	RW	Override Traffic Class Region for packet type '8 x n + 4', where 'n' is the register index.
REGION_4	19:17	000b	RW	Receive queue region for packet type '8 x n + 4', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_5	20	0b	RW	Override Traffic Class Region for packet type '8 x n + 5', where 'n' is the register index.
REGION_5	23:21	000b	RW	Receive queue region for packet type '8 x n + 5', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_6	24	0b	RW	Override Traffic Class Region for packet type '8 x n + 6', where 'n' is the register index.
REGION_6	27:25	000b	RW	Receive queue region for packet type '8 x n + 6', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_7	28	0b	RW	Override Traffic Class Region for packet type '8 x n + 7', where 'n' is the register index.
REGION_7	31:29	000b	RW	Receive queue region for packet type '8 x n + 7', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.

11.2.2.19.8 VF Queue Filter Hash Enabled Packet Type - VFQF_HENA[n,VF] (0x00230800 + 0x400*n + 0x4*VF, n=0...1, VF=0...127; RW)

Field	Bit(s)	Init.	Type	Description
PTYPE_ENA	31:0	0x0	RW	Packet Type Enablement of the Hash filter for the function. Bit 'm' in register 'n' enables packet type '32 x n + m' as defined in the “Packet Types for the Classification Filters” table.

11.2.2.19.9 PF Queue Filter Hash LUT - PFQF_HLUT[n] (0x00240000 + 0x80*n, n=0...127; RW)

Field	Bit(s)	Init.	Type	Description
LUT0	5:0	0x0	RW	Hash redirection LUT entry 4 x 'n', where 'n' is the register index.
RSVD	7:6	00b	RSV	Reserved.
LUT1	13:8	0x0	RW	Hash redirection LUT entry 4 x 'n' + 1, where 'n' is the register index.
RSVD	15:14	00b	RSV	Reserved.
LUT2	21:16	0x0	RW	Hash redirection LUT entry 4 x 'n' + 2, where 'n' is the register index.
RSVD	23:22	00b	RSV	Reserved.
LUT3	29:24	0x0	RW	Hash redirection LUT entry 4 x 'n' + 3, where 'n' is the register index.
RSVD	31:30	00b	RSV	Reserved.

11.2.2.19.10 PF Queue Filter Hash Key - PFQF_HKEY[n] (0x00244800 + 0x80*n, n=0...12; RW)

Field	Bit(s)	Init.	Type	Description
KEY_0	7:0	0x0	RW	Toeplitz key byte '4*n+0', where 'n' is the register index.
KEY_1	15:8	0x0	RW	Toeplitz key byte '4*n+1', where 'n' is the register index.
KEY_2	23:16	0x0	RW	Toeplitz key byte '4*n+2', where 'n' is the register index.
KEY_3	31:24	0x0	RW	Toeplitz key byte '4*n+3', where 'n' is the register index.

11.2.2.19.11 PF Queue Filter Hash Enabled Packet Type - PFQF_HENA[n] (0x00245900 + 0x80*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	Description
PTYPE_ENA	31:0	0x0	RW	Packet Type Enablement of the Hash filter for the function. Bit 'm' in register 'n' enables packet type '32 x n + m' as defined in the "Packet Types for the Classification Filters" table.

11.2.2.19.12 PF Queue Filter Control 1 - PFQF_CTL_1 (0x00245D80; RW)

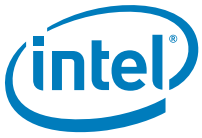
Field	Bit(s)	Init.	Type	Description
CLEARFDTABLE	0	0b	RW1C	If the *CLEARFDTABLE* flag is set by software, the hardware invalidates all entries of the PF in the FD table. Once all entries of the PF are invalidated, the *CLEARFDTABLE* flag is cleared as well.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.19.13 PF Queue Filter Flow Director Allocation - PFQF_FDALLOC (0x00246280; RW)

Field	Bit(s)	Init.	Type	Description
FDALLOC	7:0	0x0	RW	FD Allocation. Defines the number of "guaranteed" entries in the FD table. It is defined in granularity of 32 entries.
FDBEST	15:8	0x0	RW	FD Best Effort. Defines the maximum number of entries the PF can consume from the shared space. It is defined in granularity of 32 entries.
RSVD	31:16	0x0	RSV	Reserved.

11.2.2.19.14 PF Queue Filter Flow Director Allocation Status - PFQF_FDSTAT (0x00246380; RO)

Field	Bit(s)	Init.	Type	Description
GUARANT_CNT	12:0	0x0	RO	Guaranteed Count. Indicates the number of FD entries consumed by the PF out of its "guaranteed" space.
RSVD	15:13	000b	RSV	Reserved.



Field	Bit(s)	Init.	Type	Description
BEST_CNT	28:16	0x0	RO	Best Count. Indicates the number of FD entries consumed by the PF out of its "best effort" space.
RSVD	31:29	000b	RSV	Reserved.

11.2.2.19.15 Port Queue Filter Flow Director Input Set - PRTQF_FD_FLXINSET[n] (0x00253800 + 0x20*n, n=0...63; RW)

Field	Bit(s)	Init.	Type	Description
INSET	7:0	0x0	RW	Bit 'i' of the *INSET* enables word 7 minus 'i' of the flexible payload in the field vector.
RSVD	31:8	0x0	RSV	Reserved.

11.2.2.19.16 Port Queue Filter - Flexible Parser Information Table - PRTQF_FLX_PIT[n] (0x00255200 + 0x20*n, n=0...8; RW)

These registers define the byte stream extracted to the field vector (unique setting option per LAN port).

Field	Bit(s)	Init.	Type	Description
SOURCE_OFF	4:0	0x0	RW	Source word offset in the mapped protocol layer header starting from its beginning. Setting Source Offset rules: • *SOURCE_OFF* + *FSIZE* should not exceed byte 480 of the packet. • Must be programmed in ascending order: Current Offset >= previous offset + previous *FSIZE*. • The above rule applies for all entries, including non-used ones.
FSIZE	9:5	0x0	RW	Field Size defined in word units. For non-used registers the *FSIZE* must be set to 0x1.
DEST_OFF	15:10	0x0	RW	Destination word offset in the Field Vector. The destination offset can be set to 50...57 matching offset 0...7 in the flexible field vector, respectively, where *DEST_OFF* + *FSIZE* must not be greater than 58. For non-used registers the *DEST_OFF* must be set to 0x63 (outside the range for active entries).
RSVD	31:16	0x0	RSV	Reserved.

11.2.2.19.17 Port Queue Filter Control 0 - PRTQF_CTL_0 (0x00256E60; RW)

Field	Bit(s)	Init.	Type	Description
HSYM_ENA	0	0b	RW	Enable symmetric hash for this physical port.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.19.18 Global Queue Filter Packet Counter - GLQF_PCNT[n] (0x00266800 + 0x4*n, n=0...511; RW1C)

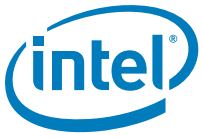
Field	Bit(s)	Init.	Type	Description
PCNT	31:0	0x0	RW1C	Packet Counter indicated by the FD filter(s). The counter wraps around after 0xFF...F.

11.2.2.19.19 Global Queue Filter SWAP Fields - GLQF_SWAP[n,m] (0x00267E00 + 0x4*n + 0x8*m, n=0...1, m=0...63; RW)

Field	Bit(s)	Init.	Type	Description
OFF0_SRC0	5:0	0x0	RW	Offset of the first field of the first couple in the Field Vector to be swapped. The offset is defined in word units.
OFF0_SRC1	11:6	0x0	RW	Offset of the second field of the first couple in the Field Vector to be swapped. The offset is defined in word units.
FLEN0	15:12	0x0	RW	Field Length in word units. When the *FLEN0* is set to zero, the fields defined by *OFF0* are not candidates for swapping.
OFF1_SRC0	21:16	0x0	RW	Offset of the first field of the second couple in the Field Vector to be swapped. The offset is defined in word units.
OFF1_SRC1	27:22	0x0	RW	Offset of the second field of the second couple in the Field Vector to be swapped. The offset is defined in word units.
FLEN1	31:28	0x0	RW	Field Length in word units. When the FLEN1 is set to zero, the fields defined by *OFF1* are not candidates for swapping.

11.2.2.19.20 Global Queue Filter Control - GLQF_CTL (0x00269BA4; RO)

Field	Bit(s)	Init.	Type	Description
RSVD	0	0b	RSV	Reserved.
HTOEP	1	0b	RW	Hash Toeplitz Select for all packet types other than FCoE packets. 0b = The hash filters are based on a simple 32 bit XOR. 1b = The hash filters are based on the standard Toeplitz scheme, while the Hash key is defined per function by the xxQFHKEY registers.
HTOEP_FCOE	2	0b	RW	Hash Toeplitz Select for FCoE packet type. 0b = The hash filters are based on a simple 32 bit XOR. 1b = The hash filters are based on the standard Toeplitz scheme, while the Hash key is defined per function by the xxQFHKEY registers.
PCNT_ALLOC	5:3	000b	RW	Controls the GLQF_PCNT counters allocation to the PF as follows: 000b = All counters are exposed to all PFs. 100b = Each PF is allocated 1/2 of the GLQF_PCNT counters. 101b = Each PF is allocated 1/4 of the GLQF_PCNT counters. 110b = Each PF is allocated 1/8 of the GLQF_PCNT counters. 111b = Each PF is allocated 1/16 of the GLQF_PCNT counters. All other values are reserved.
FD_AUTO_PCTYPE	6	0b	RW	0b - The *PCTYPE* of FD filter entries are defined by the *PCTYPE* field in the FD programming descriptor. 1b = The *PCTYPE* of FD filter entries are extracted from the programming packet, the same as it is done for received packets.
RSVD	16:7	0b	RSV	Reserved.



Field	Bit(s)	Init.	Type	Description
FDBEST	24:17	0x0	RW	FD Best Effort. Define the total number of entries in the FD table. It must not exceed the FD table size (equal to 8K) minus the sum of *FDALLOC* for all PFs. The global *FDBEST* is defined in granularity of 32 entries.
PROGPRIO	25	0b	RW	Priority ordering at filter programming between "best effort" space and "guaranteed" space. 0b = At filter programming, the hardware tries the "guaranteed" space first. Only when it is exhausted, does the hardware use the "best effort" space. 1b = At filter programming, the hardware tries the "best effort" space first. Only when it is exhausted or the PF exhausted its budget in the "best effort" space, does the hardware use the "guaranteed" space.
INVALPRIO	26	0b	RW	Priority ordering at filter invalidation between "best effort" space and "guaranteed" space. 0b = At filter invalidation, the hardware first tries to increment the "best effort" space. The "guaranteed" space is incremented only when the global "best effort" space is at it max value, or the "best effort" space of the PF is at its max value. 1b = At filter invalidation, the hardware first tries to increment its "guaranteed" space. The "best effort" space is incremented only when it is already at its max value.
IGNORE_IP	27	0b	RW	PE Quad hash filters ignore (bypass) the IP address table hit/miss indication.
RSVD	31:28	0x0	RSV	Reserved.

11.2.2.19.21 Global Queue Filter Flow Director Status 0 - GLQF_FDCNT_0 (0x00269BAC; RO)

Field	Bit(s)	Init.	Type	Description
GUARANT_CNT	12:0	0x0	RO	Total number of FD entries in "guaranteed" spaces of all PFs.
BESTCNT	25:13	0x0	RO	Total number of FD entries in the "best effort" space.
RSVD	31:26	0x0	RSV	Reserved.

11.2.2.19.22 Global Queue Filter Symmetric Hash Enablement - GLQF_HSYM[n] (0x00269D00 + 0x4*n, n=0...63; RW)

Field	Bit(s)	Init.	Type	Description
SYMH_ENA	0	0b	RW	Enables symmetric hash for PCTYPE 'n', where 'n' is the register index.
RSVD	31:1	0x0	RSV	Reserved.

11.2.2.19.23 Global Queue Filter Hash Key - GLQF_HKEY[n] (0x00270140 + 0x4*n, n=0...12; RW)

Field	Bit(s)	Init.	Type	Description
KEY_0	7:0	0x0	RW	Toeplitz key byte '4*n+0', where 'n' is the register index.
KEY_1	15:8	0x0	RW	Toeplitz key byte '4*n+1', where 'n' is the register index.
KEY_2	23:16	0x0	RW	Toeplitz key byte '4*n+2', where 'n' is the register index.
KEY_3	31:24	0x0	RW	Toeplitz key byte '4*n+3', where 'n' is the register index.

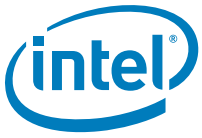
11.2.2.20 PF - TimeSync (IEEE 1588) Registers

11.2.2.20.1 Port Time Sync Control 1 - PRTTSYN_CTL1 (0x00085020; RW)

Field	Bit(s)	Init.	Type	Description
V1MESSTYPE0	7:0	0x01	RW	PTP V1 Message Type 0 for sampled timestamp of received 1588 packets. Default setting is 0x01 for Sync and Delay_Req packets. Setting this field to 0xFF is like a "wild card" option, enabling all PTP V1 packets.
V1MESSTYPE1	15:8	0x01	RW	PTP V1 Message Type 1 for sampled timestamp of received 1588 packets. Default setting is 0x01 for Sync and Delay_Req packets. Setting this field to 0xFF is like a "wild card" option, enabling all PTP V1 packets.
V2MESSTYPE0	19:16	0x0	RW	PTP V2 Message Type 0 for sampled timestamp of received 1588 packets. Default setting is 0x0 for Sync packets. Other interesting values are: 0x1 = Delay Req 0x2 = Pdelay Req 0x3 = Pdelay Resp Setting this field to 0xF is like a "wild card" option, enabling all PTP V2 packets.
V2MESSTYPE1	23:20	0x1	RW	PTP V2 Message Type 1 for sampled timestamp of received 1588 packets. Default setting is 0x1 for Delay Request.
TSYNTYPE	25:24	00b	RW	Receive packets types sampled by the 1588 timer. 00b = L2 Version 2 packets (Message Type is defined by the PRTTSYN_CTL1 register). 01b = UDP Version 1 packets (UDP ports are enabled by *UDP_ENA* field in this register and Message Type field is defined by the PRTTSYN_CTL1 register). 10b = L2 and UDP Version 2 packets (UDP ports are enabled by *UDP_ENA* field in this register and Message Type is defined by the PRTTSYN_CTL1 register). 11b = L2 and UDP Version 2 Event packets (UDP ports are enabled by *UDP_ENA* field in this register and Message Type < 8).
UDP_ENA	27:26	00b	RW	Enable the UDP ports recognized as 1588 packets. 00b = No UDP packet recognition. 01b = UDP port number equals to 0x013F. 10b = UDP port number equals to 0x0140. 11b = UDP port numbers equals to either 0x013F or 0x140.
RSVD	30:28	000b	RSV	Reserved.
TSYNENA	31	0b	RW	Enable the 1588 Logic. When the *TSYNENA* flag is cleared, the 1588 logic is not functional. This flag must be set the same as the *TSYNENA* flag in the PRTTSYN_CTL0 register.

11.2.2.20.2 Port Time Sync Receive PTP Packet Time High - PRTTSYN_RXTIME_H[n] (0x00085040 + 0x20*n, n=0...3; RO)

Field	Bit(s)	Init.	Type	Description
RXTIEM_H	31:0	0x0	RO	32 MS bits of the receive PTP Packet Time matches the units of the TSYN_TIME_H register.



11.2.2.20.3 Port Time Sync Receive PTP Packet Time Low - PRTTSYN_RXTIME_L[n] (0x000850C0 + 0x20*n, n=0...3; RO)

Field	Bit(s)	Init.	Type	Description
RXTIEM_L	31:0	0x0	RO	32 LS bits of the receive PTP Packet Time matches the units of the TSYN_TIME_L register.

11.2.2.20.4 Port Time Sync Status 1 - PRTTSYN_STAT_1 (0x00085140; RO)

Field	Bit(s)	Init.	Type	Description
RXT0	0	0b	RO	PRTTSYN_RXTIME[0] register contains valid timestamp. This bit is set by the hardware when received packet reception time is captured in the PRTTSYN_RXTIME[0] register, and it is auto-cleared when the software reads the PRTTSYN_RXTIME[0] register.
RXT1	1	0b	RO	PRTTSYN_RXTIME[1] register contains valid timestamp. This bit is set by the hardware when received packet reception time is captured in the PRTTSYN_RXTIME[1] register, and it is auto-cleared when the software reads the PRTTSYN_RXTIME[1] register.
RXT2	2	0b	RO	PRTTSYN_RXTIME[2] register contains valid timestamp. This bit is set by the hardware when received packet reception time is captured in the PRTTSYN_RXTIME[2] register, and it is auto-cleared when the software reads the PRTTSYN_RXTIME[2] register.
RXT3	3	0b	RO	PRTTSYN_RXTIME[3] register contains valid timestamp. This bit is set by the hardware when received packet reception time is captured in the PRTTSYN_RXTIME[3] register. and it is auto-cleared when the software reads the PRTTSYN_RXTIME[3] register.
RSVD	31:4	0x0	RSV	Reserved.

11.2.2.20.5 Port Time Sync Increment Value Low - PRTTSYN_INC_L (0x001E4040; RW)

Field	Bit(s)	Init.	Type	Description
TSYNINC_L	31:0	0x0	RW	32 LS bits of the Increment Value added to the 96-bit TSYNTIME registers for each MAC clock.

11.2.2.20.6 Port Time Sync Increment Value High - PRTTSYN_INC_H (0x001E4060; RW)

Field	Bit(s)	Init.	Type	Description
TSYNINC_H	5:0	0x0	RW	6 MS bits of the Increment Value added to the 96-bit TSYNTIME registers for each MAC clock.
RSVD	31:6	0x0	RSV	Reserved.

11.2.2.20.7 Port Time Sync Event Time Low - PRTTSYN_EVNT_L[n] (0x001E4080 + 0x20*n, n=0...1; RO)

Field	Bit(s)	Init.	Type	Description
TSYNEVNT_L	31:0	0x0	RO	32 LS bit of the sampled event time. The sampled event is defined by *EVNTLVL* field in the PRTTSYN_AUX register.

11.2.2.20.8 Port Time Sync Event Time High - PRTTSYN_EVNT_H[n] (0x001E40C0 + 0x20*n, n=0...1; RO)

Field	Bit(s)	Init.	Type	Description
TSYNEVNT_H	31:0	0x0	RO	32 MS bit of the sampled event time of a 1588 event defined by the PRTTSYN_AUX register.

11.2.2.20.9 Port Time Sync Time Low - PRTTSYN_TIME_L (0x001E4100; RW)

Field	Bit(s)	Init.	Type	Description
TSYNTIME_L	31:0	0x0	RW	Bits 32...63 of the 96-bit timer. If the lowest 32 bits define the fraction of nanosecond, this register defines the lower 32 bits of the nanosecond units.

11.2.2.20.10 Port Time Sync Time High - PRTTSYN_TIME_H (0x001E4120; RW)

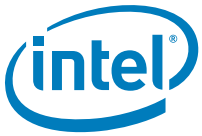
Field	Bit(s)	Init.	Type	Description
TSYNTIME_H	31:0	0x0	RW	Upper 32 bit of the 96-bit timer.

11.2.2.20.11 Port Time Sync Target Time Low - PRTTSYN_TGT_L[n] (0x001E4140 + 0x20*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	Description
TSYNTGTT_L	31:0	0x0	RW	32 LS bits of the target time of an event out in one of the AUX IO signals.

11.2.2.20.12 Port Time Sync Target Time High - PRTTSYN_TGT_H[n] (0x001E4180 + 0x20*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	Description
TSYNTGTT_H	31:0	0x0	RW	32 MS bits of the target time of an event out in one of the AUX IO signals.



11.2.2.20.13 Port Time Sync Transmit Packet Time Low - PRTTSYN_TXTIME_L (0x001E41C0; RO)

Field	Bit(s)	Init.	Type	Description
TXTIEM_L	31:0	0x0	RO	32 LS bits of the sampled 1588 Time of a Tx packet matches the units of the TSYN_TIME_L register.

11.2.2.20.14 Port Time Sync Transmit Packet Time High - PRTTSYN_TXTIME_H (0x001E41E0; RO)

Field	Bit(s)	Init.	Type	Description
TXTIEM_H	31:0	0x0	RO	32 MS bits of the sampled 1588 Time of a Tx packet matches the units of the TSYN_TIME_L register.

11.2.2.20.15 Port Time Sync Control 0 - PRTTSYN_CTL0 (0x001E4200; RW)

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
TXTIME_INT_ENA	1	0b	RW	Interrupt Enable when the TSYNTXTIME registers samples the transmission time in this port. The event is reported in the *TXTIME* bit in the PRTTSYN_STAT_0 register.
EVENT_INT_ENA	2	0b	RW	Interrupt Enable when an event is sampled in any of the PRTTSYN_EVNT registers.
TGT_INT_ENA	3	0b	RW	Interrupt Enable when the target time is sampled in any of the PRTTSYN_TGT registers.
RSVD	7:4	0x0	RSV	Reserved.
PF_ID	11:8	0x0	RW	Software Indication for the PF Function ID that controls the 1588 logic of the port (no hardware impact). This field is expected to be loaded from NVM or set by management agent. During nominal operation the PF software driver is not expected to change its setting.
TSYNACT	13:12	00b	RW	Software indication for 1588 mode of operation (no hardware impact). 00b = Inactive agent. 01b = Synchronized to local time. 10b = Synchronized to the standard TAI. 11b = Reserved.
RSVD	30:14	0x0	RSV	Reserved.
TSYNENA	31	0b	RW	Enable the 1588 Logic. When the *TSYNENA* flag is cleared, the 1588 logic is not functional. This flag must be set the same as the *TSYNENA* flag in the PRTTSYN_CTL1 register.

11.2.2.20.16 Port Time Sync Status 0 - PRTTSYN_STAT_0 (0x001E4220; RCW)

Field	Bit(s)	Init.	Type	Description
EVENT0	0	0b	RCW	Set to one when the PRTTSYN_EVNT[0] captures an input event timestamp.
EVENT1	1	0b	RCW	Set to one when the PRTTSYN_EVNT[1] captures an input event timestamp.
TGT0	2	0b	RCW	Set to one when the PRTTSYN_TGT[0] timer is expired.
TGT1	3	0b	RCW	Set to one when the PRTTSYN_TGT[1] timer is expired.
TXTIME	4	0b	RCW	Set to one when the PRTTSYN_TXTIME register samples a Tx packet.
RSVD	31:5	0x0	RSV	Reserved.

11.2.2.20.17 Port Time Sync Clock Out Duration - PRTTSYN_CLKO[n] (0x001E4240 + 0x20*n, n=0...1; RW)

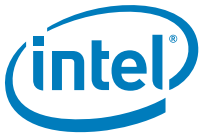
Field	Bit(s)	Init.	Type	Description
TSYNCLKO	31:0	0x0	RW	Clock output duration as described in “Auxiliary 1588 IO signals” section.

11.2.2.20.18 Port Time Sync Adjustment - PRTTSYN_ADJ (0x001E4280; RW)

Field	Bit(s)	Init.	Type	Description
TSYNADJ	30:0	0x0	RW	Absolute value of the time adjust matched to the units of the TSYN_TIME_L register.
SIGN	31	0b	RW	The sign of the “time adjustment”. 0b = Positive adjustment 1b = Negative adjustment

11.2.2.20.19 Port Time Sync AUX Control 0 - PRTTSYN_AUX_0[n] (0x001E42A0 + 0x20*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	Description
OUT_ENA	0	0b	RW	Synchronized output enablement. When set to 1b, the synchronized output signal is enabled according to the other parameters in this register.
OUTMOD	2:1	00b	RW	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	Output level driven on the IO signal at the Target Time.
RSVD	7:4	0x0	RSV	Reserved.
PULSEW	11:8	0x0	RW	Output pulse width for “Output Pulse Mode” equals to 16 x (*PULSEW* + 1) clocks. The clock frequency is defined per link speed in the “1588 Clock Registers” section.
RSVD	15:12	0x0	RSV	Reserved.



Field	Bit(s)	Init.	Type	Description
EVNTLVL	17:16	00b	RW	Event level on the IO signal configured as 1588 input. 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.
RSVD	31:18	0x0	RSV	Reserved.

11.2.2.20.20 Port Time Sync AUX Control 1 - PRTTSYN_AUX_1[n] (0x001E42E0 + 0x20*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	Description
INSTNT	0	0b	RW	If this flag is set, the "OUTSIG" signal is forced to "OUTLVL" value. This flag is auto-cleared by the hardware.
SAMPLE_TIME	1	0b	RW	Setting this flag triggers instant sampling of the PRTTSYN_TIME to the matched PRTTSYN_EVNT register. This flag is auto-cleared by the hardware.
RSVD	31:2	0x0	RSV	Reserved.

11.2.2.21 PF - FCoE Registers

11.2.2.21.1 FC Receive Control - GLFCOE_RCTL (0x00269B94; RW)

Field	Bit(s)	Init.	Type	Description
RSVD	15:0	0x0	RSV	Reserved.
MAX_SIZE	29:16	0x0	RW	Maximum received FC payload size (including DIF/DIX) that is viable for DDP. Packets with larger FC payload are not candidates for DDP and are reported as "FCoE protocol Error" in the receive descriptor. Default value is 2 KB.
RSVD	31:30	00b	RSV	Reserved.

11.2.2.22 PF - Manageability Registers

11.2.2.22.1 Firmware Reset Count - GL_FWRESETCNT (0x00083100; RO)

Field	Bit(s)	Init.	Type	Description
FWRESETCNT	31:0	0x0	RO	Firmware resets count. Updated by Hardware. Saturates at 0xFFFF,FFFF.

11.2.2.22.2 Flexible TCO Filter Table Registers - Mask - PRT_MNG_FTFT_MASK[n] (0x00085160 + 0x20*n, n=0...7; RO)

Note: The mask field must be 8 bytes aligned even if the length field is not 8 bytes aligned, as the hardware implementation compares 8 bytes at a time so it should get extra masks until the end of the next quad word. Any mask bit that is located after the length should be set to 0, indicating no comparison should be done.

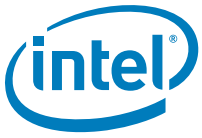
Note: In case the actual length which is defined by the length field register and the mask bits is not 8 bytes aligned, there may be a case that a packet which is shorter than the actual required length passes the flexible filter. This may occur due to comparison of up to 7 bytes that come after the packet, but are not a real part of the packet.

Field	Bit(s)	Init.	Type	Description
MASK	15:0	0x0	RW	Masks for the filter bytes: PRT_MNG_FTFT_MASK[0] : Mask for bytes 0:15 PRT_MNG_FTFT_MASK[1] : Mask for bytes 16:31 .... PRT_MNG_FTFT_MASK[6] : Mask for bytes 96:111 PRT_MNG_FTFT_MASK[7] : Mask for bytes 112:127 Values are: 0b = Ignore 1b = Compare
RESERVED	31:16	0x0	RSV	Reserved.

11.2.2.22.3 Flexible TCO Filter Table Registers - Length - PRT_MNG_FTFT_LENGTH (0x00085260; RO)

Contains the length of the filter.

Field	Bit(s)	Init.	Type	Description
LENGTH	7:0	0x0	RW	This field contains the length of the filter defining the number of bytes from the beginning of the packet compared by this filter. If actual packet length is less than length specified by this field, the filter fails. Otherwise, it depends on the result of actual byte comparison. The value should not be greater than 128.
RESERVED	31:8	0x0	RSV	Reserved.



11.2.2.22.4 Flexible TCO Filter Table Registers - Data - PRT_MNG_FTFT_DATA[n] (0x000852A0 + 0x20*n, n=0...31; RO)

The Flexible TCO Filter Table registers (FTFT) contains a 128-byte pattern and a corresponding 128-bit mask array. If enabled, the first 128 bytes of the received packet are compared against the masked bytes in the PRT_MNG_FTFT_DATA registers.

The 128-byte filter is composed of 32 DW entries, accompanied by a 128-bit mask (PRT_MNG_FTFT_MASK, one bit per filter byte). The bytes in each DWs are written in network order (i.e., Byte0 written to Bits[7:0], Byte1 to Bits[15:8], and so on). The mask field is set so that Bit[0] in the mask masks Byte0, Bit[1] masks Byte1, and so on. A value of 1 in the mask field means that the appropriate byte in the filter should be compared to the appropriate byte in the incoming packet.

The PRT_MNG_FTFT_LENGTH register contains a length field defining the number of bytes from the beginning of the packet compared by this filter. If actual packet length is less than length specified by this field, the filter fails. Otherwise, it depends on the result of actual byte comparison. The value should not be greater than 128.

Field	Bit(s)	Init.	Type	Description
DWORD	31:0	0x000000	RW	Filter Data.

11.2.2.22.5 Hardware Arbitration Control - GL_MNG_HWARB_CTRL (0x000B6130; RO)

Field	Bit(s)	Init.	Type	Description
NCSI_ARB_EN	0	0b	RW	Hardware Arbitration Enable. If this bit is set, it is assumed the NCSI_ARB_IN and NCSI_ARB_OUT are connected to an Hardware arbitration ring. Otherwise, the NCSI_ARB_IN pin is pulled up internally.
RESERVED	31:1	0x0	RSV	Reserved.

11.2.2.22.6 Firmware Semaphore - GL_MNG_FWSM (0x000B6134; RO)

Field	Bit(s)	Init.	Type	Description
FW_MODES	1:0	00b	RW	Firmware Mode. Indicate in which mode the FW operates. 00b = Normal mode 01b = Debug mode 10b = Recovery mode 11b = Debug + Recovery mode.
RESERVED	9:2	0x0	RSV	Reserved.
EEP_RELOAD_IND	10	0b	RW	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.

Field	Bit(s)	Init.	Type	Description
CRC_ERROR_MODULE	14:11	0x0	RW	Index of (first) module for which a CRC error was found by EMP check. 0x0 = No CRC error found by EMP. **Note:** The PE may have found a CRC error in one of the modules it handles. It is reported in GLPE_FWLDSTATUS register. 0x1 = CRC error on EMP Image module. 0x2 = CRC error on PCIe Analog module. 0x3 = CRC error on PHY Analog module. 0x4 = CRC error on EMP Global module. 0x5 = CRC error on Manageability module. 0x6 = CRC error on EMP Settings module.
FW_STATUS_VALID	15	0b	RW	Firmware Valid Bit. Hardware clears Bits[15:0] in EMP reset de-assertion so software can know firmware status is invalid. Firmware should set this bit to 1b when it is ready (end of init sequence). Whenever setting this bit to 1b, a PFINT_ICR0.ADMINQ interrupt must be issued to host.
RESET_CNT	18:16	000b	RW	Reset Counter. Firmware increments the count on every EMP reset. After 7 EMP reset events, counter stays stuck at 7 and does not wrap around.
EXT_ERR_IND	24:19	0x0	RW	External Error Indication. Firmware writes here the reason that the firmware operation has stopped. For example, NVM CRC error, etc. Possible values: 0x0 = No error 0x1 = CRC error on a module handled by EMP. Refer to CRC_ERROR_MODULE field for details. 0x2 = Switch module failed. 0x3 = Scheduler module failed. 0x4 = DCB module failed 0x5 = Link module failed 0x6 = LLDP module failed 0x7 = Manage module failed. 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 a value of 0x0 is placed in this field, the PFINT_ICR0.*ADMINQ* bit is not set, and an interrupt is not generated.
RESERVED	25	0b	RSV	Reserved.
PHY_SERDES0_CONFIG_ERR	26	0b	RW	PHY/SerDes Configuration Error Indication - Port 0. Set by firmware when it fails to configure LAN PHY/SerDes. Cleared by firmware upon successful configuration of LAN PHY/SerDes.
PHY_SERDES1_CONFIG_ERR	27	0b	RW	PHY/SerDes Configuration Error Indication - Port 1. Set by firmware when it fails to configure LAN PHY/SerDes. Cleared by firmware upon successful configuration of LAN PHY/SerDes.
PHY_SERDES2_CONFIG_ERR	28	0b	RW	PHY/SerDes Configuration Error Indication - Port 2. Set by firmware when it fails to configure LAN PHY/SerDes. Cleared by firmware upon successful configuration of LAN PHY/SerDes.
PHY_SERDES3_CONFIG_ERR	29	0b	RW	PHY/SerDes Configuration Error Indication - Port 3. Set by firmware when it fails to configure LAN PHY/SerDes. Cleared by firmware upon successful configuration of LAN PHY/SerDes.

Field	Bit(s)	Init.	Type	Description
RESERVED	31:30	00b	RSV	Reserved.

11.2.2.22.7 Manageability IPv6 Address Filter - PRT_MNG_MIPAF6[n] (0x00254200 + 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	Description
MIPAF	31:0	0x000000	RW	Manageability IP Address Filters. For each n, m, m=0...3, n=0...3, MIPAF[m,n] register holds DW `n' of IPv6 filter `m' (4 x IPv6 filters).

11.2.2.22.8 Management Flex UDP/TCP Ports - PRT_MNG_MFUTP[n] (0x00254E00 + 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 10.3.

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	Description
MFUTP_N	15:0	0x0	RW	n-th Management Flex UDP/TCP port.
UDP	16	0b	RW	Match if port is UDP.
TCP	17	0b	RW	Match if port is TCP.
SOURCE_DESTINATION	18	0b	RW	0b = Compare Destination port. 1b = Compare Source port.
RESERVED	31:19	0x0	RSV	Reserved.

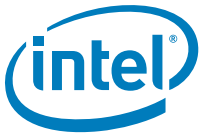
11.2.2.22.9 Management VLAN TAG Value - PRT_MNG_MAVTV[n] (0x00255900 + 0x20*n, n=0...7; RO)

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 as described in the Management chapter.

Field	Bit(s)	Init.	Type	Description
VID	11:0	0x0	RW	Contains the VLAN ID that should be compared with the incoming packet's inner VLAN ID, if the corresponding bit in MDEF is set.
RSVD	31:12	0x0	RSV	Reserved.

11.2.2.22.10 Manageability Decision Filters1 - PRT_MNG_MDEF[n] (0x00255D00 + 0x20*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	Description
MAC_EXACT_AND	3:0	0x0	RW	Exact. Controls the inclusion of Exact MAC address 0 to 3. In the manageability filter decision (AND section). Bit[0] corresponds to exact MAC address 0 (MMAL0 and MMAH0), etc.
BROADCAST_AND	4	0b	RW	Broadcast. Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section).
VLAN_AND	12:5	0x0	RW	VLAN. Controls the inclusion of VLAN tag 0 to 7, respectively, in the manageability filter decision (AND section). Bit[5] corresponds to VLAN tag 0, etc.
IPV4_ADDRESS_AND	16:13	0x0	RW	IPv4 Address. Controls the inclusion of IPV4 address 0 to 3, respectively, in the manageability filter decision (AND section). Bit[13] corresponds to IPV4 address 0, etc. ***Note:*** These bits are also set for an ARP request packet if the Target IP matches the IP address configured in the MIPAF register.
IPV6_ADDRESS_AND	20:17	0x0	RW	IPv6 Address. Controls the inclusion of IPV6 address 0 to 3, respectively, in the manageability filter decision (AND section). Bit[17] corresponds to IPV6 address 0, etc.
MAC_EXACT_OR	24:21	0x0	RW	Exact. Controls the inclusion of exact MAC address 0 to 3 In the manageability filter decision (OR section). Bit[21] corresponds to exact MAC address 0 (MMAL0 and MMAH0), etc.
BROADCAST_OR	25	0b	RW	Broadcast. Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section).
MULTICAST_AND	26	0b	RW	Multicast. Controls the inclusion of Multicast address filtering in the manageability filter decision (AND section). Broadcast packets are not included by this bit.
ARP_REQUEST_OR	27	0b	RW	ARP Request. Controls the inclusion of ARP Request filtering in the manageability filter decision (OR section).
ARP_RESPONSE_OR	28	0b	RW	ARP Response. Controls the inclusion of ARP Response filtering in the manageability filter decision (OR section).
NEIGHBOR_DISCOVERY_134_OR	29	0b	RW	Neighbor Discovery. 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	Port 0x298. Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section).
PORT_0X26F_OR	31	0b	RW	Port 0x26F. Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section).



11.2.2.22.11 Manageability Decision Filters - PRT_MNG_MDEF_EXT[n] (0x00255F00 + 0x20*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	Description
L2_ETHERTYPE_AND	3:0	0x0	RW	L2 EtherType. Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section).
L2_ETHERTYPE_OR	7:4	0x0	RW	L2 EtherType. Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section).
FLEX_PORT_OR	23:8	0x0	RW	Flex Port. Controls the inclusion of Flex port filtering in the manageability filter decision (OR section). Bit[16] corresponds to flex port 0, etc.
FLEX_TCO	24	0b	RW	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	Neighbor Discovery. 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	Neighbor Discovery. 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	Neighbor Discovery. 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	Controls the inclusion of ICMP filtering in the manageability filter decision (OR section).
MLD	29	0b	RW	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	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	0b = This decision filter does not apply to traffic received from the host. 1b = This decision filter applies to traffic received from the host.

11.2.2.22.12 Manageability IPv4 Address Filter - PRT_MNG_MIPAF4[n] (0x00256280 + 0x20*n, n=0...3; RO)

The Manageability IPv4 Address Filter register stores IPv4 addresses for manageability filtering.

Note: These registers should be written in network order.

Field	Bit(s)	Init.	Type	Description
MIPAF	31:0	0x000000	RW	Manageability IP Address Filters. For each n, m, m=0...3, n=0...3, MIPAF[m,n] register holds DW `n' of IPv6 filter `m' (4 x IPv6 filters).

11.2.2.22.13 Manageability MAC Address High - PRT_MNG_MMAH[n] (0x00256380 + 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 10.3.

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	Description
MMAH	15:0	0x000000	RW	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	Reserved. Reads as 0. Ignored on write.

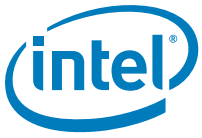
11.2.2.22.14 Manageability MAC Address Low - PRT_MNG_MMAL[n] (0x00256480 + 0x20*n, n=0...3; RO)

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 10.3.

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 MMAL.MMAL field should be written in network order.

Field	Bit(s)	Init.	Type	Description
MMAL	31:0	0x000000	RW	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).



11.2.2.22.15 Management Decision Filters Buffers - PRT_MNG_MDEFVSI[n] (0x00256580 + 0x20*n, n=0...3; RO)

This register is used to define the VSIs 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.

Field	Bit(s)	Init.	Type	Description
MDEFVSI_2N	15:0	0x0	RW	Defines the VSI used for packets matching MDEF 2*n.
MDEFVSI_2NP1	31:16	0x0	RW	Defines the VSI used for packets matching MDEF 2*n+1.

11.2.2.22.16 Management Ethernet Type Filters - PRT_MNG_METF[n] (0x00256780 + 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 10.3.

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	Description
ETYPE	15:0	0x0	RW	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	Reserved.
POLARITY	30	0b	RW	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	Reserved.

11.2.2.22.17 Management Control Register - PRT_MNG_MANC (0x00256A20; RO)

The MANC register can be written by the BMC and is not accessible to the host for writing.

Field	Bit(s)	Init.	Type	Description
FLOW_CONTROL_DISCARD	0	0b	RW	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	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	Reserved.
RCV_TCO_EN	17	0b	RW	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	Reserved

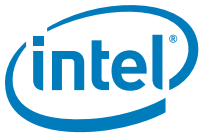
Field	Bit(s)	Init.	Type	Description
FIXED_NET_TYPE	25	0b	RW	Fixed Next Type. If set, only packets matching the net type defined by the *NET_TYPE* field pass to manageability. Otherwise, both tagged and un-tagged packets may be forwarded to manageability engine.
NET_TYPE	26	0b	RW	Net Type. 0b = Pass only un-tagged packets. 1b = Pass only VLAN tagged packets. Valid only if *FIXED_NET_TYPE* is set.
RESERVED	27	0b	RSV	Reserved.
EN_BMC2OS	28	0b	RW	Enable BMC to OS and OS to BMC traffic. 0b = The BMC can not 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.
EN_BMC2NET	29	0b	RW	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. ***Note:*** Initial value loaded according to value of Port n traffic types field in NVM.
RESERVED	31:30	0x0	RSV	Reserved.

11.2.2.22.18 Management Only Traffic Register - PRT_MNG_MNGONLY (0x00256A60; 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 will not be 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 may also block the packet from being forwarded to the host if the corresponding MNGONLY bit is set.

Field	Bit(s)	Init.	Type	Description
EXCLUSIVE_TO_MANAGEABILITY	7:0	0x0	RW	Exclusive to MNG. When set, indicates that packets forwarded by the manageability filters to manageability are not sent to the host. Bits 0...7 correspond to decision rules defined in registers MDEF[0...7] and MDEF_EXT[0...7].
RESERVED	31:8	0x0	RSV	Reserved.



11.2.2.22.19 Manageability Special Filters Modifiers - PRT_MNG_MSFM (0x00256AA0; RO)

Field	Bit(s)	Init.	Type	Description
PORT_26F_UDP	0	1b	RW	Port 0x26F match if protocol is UDP.
PORT_26F_TCP	1	1b	RW	Port 0x26F match if protocol is TCP.
PORT_298_UDP	2	1b	RW	Port 0x298 match if protocol is UDP.
PORT_298_TCP	3	1b	RW	Port 0x298 match if protocol is TCP.
IPV6_0_MASK	4	0b	RW	Compare only 24 LSB bits of IPv6 Address 0 (MIPAF[0]).
IPV6_1_MASK	5	0b	RW	Compare only 24 LSB bits of IPv6 Address 1 (MIPAF[1]).
IPV6_2_MASK	6	0b	RW	Compare only 24 LSB bits of IPv6 Address 2 (MIPAF[2]).
IPV6_3_MASK	7	0b	RW	Compare only 24 LSB bits of IPv6 Address 3 (MIPAF[3]).
RESERVED	31:8	0x0	RSV	Reserved.

11.2.3 BAR3 Registers Summary

Table 11-7. BAR3 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Page
PF - MSI-X Table Registers			
0x00000000 + 0x10*n, n=0...128	MSIX_TADD[n]	MSI-X Message Address Low	1476
0x00000004 + 0x10*n, n=0...128	MSIX_TUADD[n]	MSI-X Message Address High	1477
0x00000008 + 0x10*n, n=0...128	MSIX_TMSG[n]	MSI-X Message Data	1477
0x0000000C + 0x10*n, n=0...128	MSIX_TVCTRL[n]	MSI-X Vector Control	1477
0x00001000 + 0x4*n, n=0...5	MSIX_PBA[n]	MSI-X PBA Structure	1477
0x00002000 + 0x4*n, n=0...19	VFMSIX_PBA[n]	VF MSI-X PBA Structure	1478
0x00002100 + 0x10*n, n=0...639	VFMSIX_TADD[n]	VF MSI-X Message Address Low	1478
0x00002104 + 0x10*n, n=0...639	VFMSIX_TUADD[n]	VF MSI-X Message Address High	1478
0x00002108 + 0x10*n, n=0...639	VFMSIX_TMSG[n]	VF MSI-X Message Data	1478
0x0000210C + 0x10*n, n=0...639	VFMSIX_TVCTRL[n]	VF MSI-X Vector Control	1479

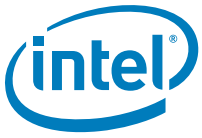
11.2.4 Detailed Register Description - PF BAR3

11.2.4.1 PF - MSI-X Table Registers

11.2.4.1.1 MSI-X Message Address Low - MSIX_TADD[n] (0x00000000 + 0x10*n, n=0...128; RW)

Message address for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MSIXTADD10	1:0	00b	RW	Message Address. For proper Dword alignment, software must always write zeros to these two bits. Otherwise, the result is undefined. The state of these bits after reset must be 0b. These bits are permitted to be read-only or read/write.
MSIXTADD	31:2	0x0	RW	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.



11.2.4.1.2 MSI-X Message Address High - MSIX_TUADD[n] (0x00000004 + 0x10*n, n=0...128; RW)

Message upper address for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MSIXTUADD	31:0	0x0	RW	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.

11.2.4.1.3 MSI-X Message Data - MSIX_TMSG[n] (0x00000008 + 0x10*n, n=0...128; RW)

Message data for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MSIXTMSG	31:0	0x0	RW	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.

11.2.4.1.4 MSI-X Vector Control - MSIX_TVCTRL[n] (0x0000000C + 0x10*n, n=0...128; RW)

Vector control for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MASK	0	1b	RW	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	Reserved. After reset, the state of these bits must be 0b. 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.

11.2.4.1.5 MSI-X PBA Structure - MSIX_PBA[n] (0x00001000 + 0x4*n, n=0...5; RO)

Pending bits for MSI-X PBA entries.

Field	Bit(s)	Init.	Type	Description
PENBIT	31:0	0x0	RO	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.

11.2.4.1.6 VF MSI-X PBA Structure - VFMSIX_PBA[n] (0x00002000 + 0x4*n, n=0...19; RO)

Pending bits for MSI-X PBA entries.

Field	Bit(s)	Init.	Type	Description
PENBIT	31:0	0x0	RO	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.

11.2.4.1.7 VF MSI-X Message Address Low - VFMSIX_TADD[n] (0x00002100 + 0x10*n, n=0...639; RW)

Message address for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MSIXTADD10	1:0	00b	RW	Message Address. For proper Dword alignment, software must always write zeros to these two bits. Otherwise, the result is undefined. The state of these bits after reset must be 0b. These bits are permitted to be read-only or read/write.
MSIXTADD	31:2	0x0	RW	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.

11.2.4.1.8 VF MSI-X Message Address High - VFMSIX_TUADD[n] (0x00002104 + 0x10*n, n=0...639; RW)

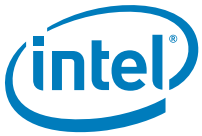
Message upper address for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MSIXTUADD	31:0	0x0	RW	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.

11.2.4.1.9 VF MSI-X Message Data - VFMSIX_TMSG[n] (0x00002108 + 0x10*n, n=0...639; RW)

Message data for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MSIXTMSG	31:0	0x0	RW	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.



11.2.4.1.10 VF MSI-X Vector Control - VFMSIX_TVCTRL[n] (0x0000210C + 0x10*n, n=0...639) msix_regs

Vector control for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MASK	0	1b	RW	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	Reserved. After reset, the state of these bits must be 0b. 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.

11.3 Device Registers - VF

11.3.1 VF Registers Mapping in the PF Space

Figure 11-1 VF Registers Summary

Abbreviation	Virtual Address	Physical Address
VFMSIX_TADD	0x00000000 + 0x10*n, n=0...16	0x00002100 + 0x10*n, n=0...639
QTX_TAIL	0x00000000 + 0x4*Q, Q=0...15	0x00108000 + 0x4*Q, Q=0...1535
VFMSIX_TUADD	0x00000004 + 0x10*n, n=0...16	0x00002104 + 0x10*n, n=0...639
VFMSIX_TMSG	0x00000008 + 0x10*n, n=0...16	0x00002108 + 0x10*n, n=0...639
VFMSIX_TVCTRL	0x0000000C + 0x10*n, n=0...16	0x0000210C + 0x10*n, n=0...639
VFMSIX_PBA	0x00002000	0x00002000 + 0x4*n, n=0...19
QRX_TAIL	0x00002000 + 0x4*Q, Q=0...15	0x00128000 + 0x4*Q, Q=0...1535
VFINT_ITRN	0x00002800 + 0x40*n + 0x4*INTVF, n=0...2, INTVF=0...15	0x00020000 + 0x800*n + 0x4*INTVF, n=0...2, INTVF=0...511
VFINT_DYN_CTLN	0x00003800 + 0x4*INTVF, INTVF=0...15	0x00024800 + 0x4*INTVF, INTVF=0...511
VFINT_ICR0	0x00004800	0x0002BC00 + 0x4*VF, VF=0...127
VFINT_ITR0	0x00004C00 + 0x4*n, n=0...2	0x00028000 + 0x400*n + 0x4*VF, n=0...2, VF=0...127
VFINT_ICR0_ENA	0x00005000	0x0002C000 + 0x4*VF, VF=0...127
VFINT_STAT_CTL0	0x00005400	0x0002A000 + 0x4*VF, VF=0...127
VFINT_DYN_CTL0	0x00005C00	0x0002A400 + 0x4*VF, VF=0...127
VF_ARQBAH	0x00006000	0x00081400 + 0x4*VF, VF=0...127
VF_ATQH	0x00006400	0x00082000 + 0x4*VF, VF=0...127
VF_ATQLEN	0x00006800	0x00081800 + 0x4*VF, VF=0...127
VF_ARQBAL	0x00006C00	0x00080C00 + 0x4*VF, VF=0...127
VF_ARQT	0x00007000	0x00082C00 + 0x4*VF, VF=0...127
VF_ARQH	0x00007400	0x00082400 + 0x4*VF, VF=0...127
VF_ATQBAH	0x00007800	0x00081000 + 0x4*VF, VF=0...127
VF_ATQBAL	0x00007C00	0x00080800 + 0x4*VF, VF=0...127
VF_ARQLEN	0x00008000	0x00081C00 + 0x4*VF, VF=0...127
VF_ATQT	0x00008400	0x00082800 + 0x4*VF, VF=0...127
VFGEN_RSTAT	0x00008800	0x00074400 + 0x4*VF, VF=0...127
VFQF_HENA	0x0000C400 + 0x4*n, n=0...1	0x00230800 + 0x400*n + 0x4*VF, n=0...1, VF=0...127
VFQF_HKEY	0x0000CC00 + 0x4*n, n=0...12	0x00228000 + 0x400*n + 0x4*VF, n=0...12, VF=0...127
VFQF_HLUT	0x0000D000 + 0x4*n, n=0...15	0x00220000 + 0x400*n + 0x4*VF, n=0...15, VF=0...127
VFQF_HREGION	0x0000D400 + 0x4*n, n=0...7	0x0022E000 + 0x400*n + 0x4*VF, n=0...7, VF=0...127

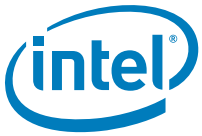


Figure 11-1 VF Registers Summary

Abbreviation	Virtual Address	Physical Address
VFCM_PE_ERRINFO	0x0000D800	0x00138400 + 0x4*VF, VF=0...127
VFCM_PE_ERRDATA	0x0000DC00	0x00138800 + 0x4*VF, VF=0...127
PFPCI_VF_FLUSH_DONE	0x0000E400	0x0009C600 + 0x4*VF, VF=0...127

11.3.2 BAR0 Registers Summary

Table 11-8. BAR0 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Page
PF - PCIe Registers			
0x0000E400	PFPCI_VF_FLUSH_DONE	PCIe VF Flush Done	1482
VF - General Registers			
0x00008800	VFGEN_RSTAT	VF Reset Status	1482
VF - Interrupts			
0x00002800 + 0x40*n + 0x4*INTVF, n=0...2, INTVF=0...15	VFINT_ITRN[n,INTVF]	VF Interrupt Throttling for Interrupt N	1483
0x00003800 + 0x4*INTVF, INTVF=0...15	VFINT_DYN_CTLN[INTVF]	VF Interrupt N Dynamic Control	1483
0x00004800	VFINT_ICR0	VF Interrupt Zero Cause	1484
0x00004C00 + 0x4*n, n=0...2	VFINT_ITR0[n]	VF Interrupt Throttling for Interrupt Zero	1484
0x00005000	VFINT_ICR0_ENA	VF Interrupt Zero Cause Enablement	1484
0x00005400	VFINT_STAT_CTL0	VF Interrupt Zero Static Control	1485
0x00005C00	VFINT_DYN_CTL0	VF Interrupt Zero Dynamic Control	1485
VF - Admin Queue			
0x00006000	VF_ARQBAH	VF Admin Receive Queue Base Address High	1486
0x00006400	VF_ATQH	VF Admin Transmit Head	1486
0x00006800	VF_ATQLEN	VF Admin Transmit Queue Length	1486
0x00006C00	VF_ARQBAL	VF Admin Receive Queue Base Address Low	1486
0x00007000	VF_ARQT	VF Admin Receive Queue Tail	1487
0x00007400	VF_ARQH	VF Admin Receive Queue Head	1487
0x00007800	VF_ATQBAH	VF Admin Transmit Queue Base Address High	1487
0x00007C00	VF_ATQBAL	VF Admin Transmit Queue Base Address Low	1487
0x00008000	VF_ARQLEN	VF Admin Receive Queue Length	1487
0x00008400	VF_ATQT	VF Admin Transmit Tail	1488
VF - LAN Transmit Receive Registers			
0x00000000 + 0x4*Q, Q=0...15	QTX_TAIL[Q]	Global Transmit Queue Tail	1488
0x00002000 + 0x4*Q, Q=0...15	QRX_TAIL[Q]	Global Receive Queue Tail	1488

Table 11-8. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Page
VF - Rx Filters Registers			
0x0000C400 + 0x4*n, n=0...1	VFQF_HENA[n]	VF Queue Filter Hash Enabled Packet Type	1489
0x0000CC00 + 0x4*n, n=0...12	VFQF_HKEY[n]	VF Queue Filter Hash Key	1489
0x0000D000 + 0x4*n, n=0...15	VFQF_HLUT[n]	VF Queue Filter Hash LUT	1489
0x0000D400 + 0x4*n, n=0...7	VFQF_HREGION[n]	VF Queue Filter Hash Region of Queues	1489
VF - PE Registers			
0x0000D800	VFCM_PE_ERRINFO	CMPE VF Error Info	1490
0x0000DC00	VFCM_PE_ERRDATA	CMPE VF Error Data	1491

11.3.3 Detailed Register Description - VF BAR0

11.3.3.1 PF - PCIe Registers

11.3.3.1.1 PCIe VF Flush Done - PFPCI_VF_FLUSH_DONE (0x0000E400; RO)

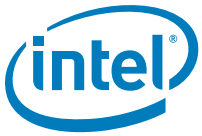
Field	Bit(s)	Init.	Type	Description
FLUSH_DONE	0	0b	RO	
RSVD	31:1	0x0	RSV	

11.3.3.2 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.

11.3.3.2.1 VF Reset Status - VFGEN_RSTAT (0x00008800; RW)

Field	Bit(s)	Init.	Type	Description
VFR_STATE	1:0	00b	RW	The 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.
RSVD	31:2	0x0	RSV	Reserved.



11.3.3.3 VF - Interrupts

11.3.3.3.1 VF Interrupt Throttling for Interrupt N - VFINT_ITRN[n,INTVF] (0x00002800 + 0x40*n + 0x4*INTVF, n=0...2, INTVF=0...15; RW)

Register index 'n' relates to interrupt 'n+1', while interrupt zero is controlled by the VFINT_ITR0 register.

Field	Bit(s)	Init.	Type	Description
INTERVAL	11:0	0x0	RW	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 also be programmed by setting the *INTERVAL* field in the matched xxINT_DYN_CTLx register.
RSVD	31:12	0x0	RSV	Reserved.

11.3.3.3.2 VF Interrupt N Dynamic Control - VFINT_DYN_CTLN[INTVF] (0x00003800 + 0x4*INTVF, INTVF=0...15; RW)

Register index 'n' relates to interrupt 'n+1', while interrupt zero is controlled by the VFINT_DYN_CTL0 register.

Field	Bit(s)	Init.	Type	Description
INTENA	0	0b	RW	Interrupt Enable. 0b = Interrupt disabled. 1b = Interrupt enabled. Refer to auto-clear policy in the “Interrupt Enablement” section. This bit is meaningful only if the *INTENA_MSK* flag in this register is not set.
CLEARPBA	1	0b	RW1C	Setting this bit clears the matched PBA bit. This bit is auto-cleared by hardware.
SWINT_TRIG	2	0b	RW1C	Trigger SW Interrupt. When this bit is set, a SW interrupt is triggered. This bit is auto-cleared by hardware.
ITR_INDX	4:3	00b	RW1C	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 hardware.
INTERVAL	16:5	0x0	RW1C	The interval for the ITR defined by the *ITR_INDX* field in this register. This field is auto-cleared by hardware.
RSVD	23:17	0x0	RSV	Reserved.
SW_ITR_INDX_ENA	24	0b	RW1C	Enables the programming of the *SW_ITR_INDX* field in this register. This flag is auto-cleared by hardware.

Field	Bit(s)	Init.	Type	Description
SW_ITR_INDX	26:25	00b	RW	ITR Index of the SW interrupt: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR When programming this field, the *SW_ITR_INDX_ENA* flag in this register should be set as well.
RSVD	30:27	0x0	RSV	Reserved.
INTENA_MSK	31	0b	RW1C	When this bit is set, the *INTENA* setting does not impact the device setting. This bit is auto-cleared by hardware.

11.3.3.3.3 VF Interrupt Zero Cause - VFINT_ICR0 (0x00004800; RCW)

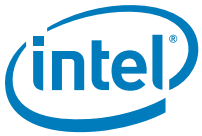
Field	Bit(s)	Init.	Type	Description
INTEVENT	0	0b	RCW	Interrupt Event indication. This bit is set on assertion of any causes for this interrupt, and cleared when the interrupt is asserted to the Rate Limit logic.
QUEUE_0	1	0b	RCW	Queue 0 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_1	2	0b	RCW	Queue 1 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_2	3	0b	RCW	Queue 2 interrupt for LAN Transmit and Receive queues and PE CEQs.
QUEUE_3	4	0b	RCW	Queue 3 interrupt for LAN Transmit and Receive queues and PE CEQs.
RSVD	29:5	0x0	RSV	Reserved.
ADMINQ	30	0b	RCW	Send/Receive Admin queue interrupt indication.
SWINT	31	0b	RCW	Software Interrupt indication.

11.3.3.3.4 VF Interrupt Throttling for Interrupt Zero - VFINT_ITR0[n] (0x00004C00 + 0x4*n, n=0...2; RW)

Field	Bit(s)	Init.	Type	Description
INTERVAL	11:0	0x0	RW	ITR 'n' interval, where 'n' is the register index = 0,1,2 for the three ITRs per interrupt. 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 also be programmed by setting the *INTERVAL* field in the matched xxINT_DYN_CTLx register.
RSVD	31:12	0x0	RSV	Reserved.

11.3.3.3.5 VF Interrupt Zero Cause Enablement - VFINT_ICR0_ENA (0x00005000; RW)

Field	Bit(s)	Init.	Type	Description
RSVD	29:0	0x0	RSV	Reserved.
ADMINQ	30	0b	RW	Enable this interrupt at '1'.
RSVD	31	0b	RW	Reserved.



11.3.3.3.6 VF Interrupt Zero Static Control - VFINT_STAT_CTL0 (0x00005400; RW)

In case of MSI or Legacy INTA mode of operation, Interrupt zero is the only valid interrupt.

Field	Bit(s)	Init.	Type	Description
RSVD	1:0	00b	RSV	Reserved.
OTHER_ITR_INDX	3:2	00b	RW	ITR Index of the "other" interrupt causes: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR
RSVD	31:4	0x0	RSV	Reserved.

11.3.3.3.7 VF Interrupt Zero Dynamic Control - VFINT_DYN_CTL0 (0x00005C00; RW)

Field	Bit(s)	Init.	Type	Description
INTENA	0	0b	RW	Interrupt Enable. 0b = Interrupt disabled. 1b = Interrupt enabled. Refer to the auto-clear policy in the "Interrupt Enablement" section. This bit is meaningful only if *INTENA_MSK* flag in this register is not set.
CLEARPBA	1	0b	RW1C	Setting this bit clears the matched PBA bit. This bit is auto-cleared by hardware.
SWINT_TRIG	2	0b	RW1C	Trigger SW Interrupt. When the bit is set, a SW interrupt is triggered. This bit is auto-cleared by hardware.
ITR_INDX	4:3	00b	RW1C	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 hardware:
INTERVAL	16:5	0x0	RW1C	The interval for the ITR defined by the *ITR_INDX* field in this register. This field is auto-cleared by hardware.
RSVD	23:17	0x0	RSV	Reserved.
SW_ITR_INDX_ENA	24	0b	RW1C	Enables the programming of the *SW_ITR_INDX* field in this register. This flag is auto-cleared by hardware.
SW_ITR_INDX	26:25	00b	RW	ITR Index of the SW interrupt: 00b = ITR0 01b = ITR1 10b = ITR2 11b = NoITR When programming this field, the *SW_ITR_INDX_ENA* flag in this register should be set as well.
RSVD	30:27	0x0	RSV	Reserved.
INTENA_MSK	31	0b	RW1C	When this bit is set, the *INTENA* setting does not impact the device setting. This bit is auto-cleared by hardware.

11.3.3.4 VF - Admin Queue

11.3.3.4.1 VF Admin Receive Queue Base Address High - VF_ARQBAH (0x00006000; RW)

Field	Bit(s)	Init.	Type	Description
ARQBAH	31:0	0x0	RW	Receive descriptor base address high.

11.3.3.4.2 VF Admin Transmit Head - VF_ATQH (0x00006400; RW)

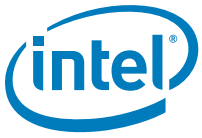
Field	Bit(s)	Init.	Type	Description
ATQH	9:0	0x0	RW	Transmit queue head pointer. At queue initialization the software clears the head pointer and during nominal operation the Firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	Reserved.

11.3.3.4.3 VF Admin Transmit Queue Length - VF_ATQLEN (0x00006800; RW)

Field	Bit(s)	Init.	Type	Description
ATQLEN	9:0	0x0	RW	Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	Reserved.
ATQVFE	28	0b	RW	VF Error. Set by FW on a PF queue when one of its VFs had an admin queue error.
ATQOVFL	29	0b	RW	Overflow Error. Set by FW when a message was lost because there was no room on the queue.
ATQCRIT	30	0b	RW	Critical Error. Set by FW when a critical error has been detected on this queue.
ATQENABLE	31	0b	RW	Enable. Set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is cleared by VFR.

11.3.3.4.4 VF Admin Receive Queue Base Address Low - VF_ARQBAL (0x00006C00; RW)

Field	Bit(s)	Init.	Type	Description
ARQBAL	31:0	0x0	RW	Receive descriptor base address low. Must be 64-byte aligned.



11.3.3.4.5 VF Admin Receive Queue Tail - VF_ARQT (0x00007000; RW)

Field	Bit(s)	Init.	Type	Description
ARQT	9:0	0x0	RW	Receive queue tail. Incremented to indicate that there are new valid descriptors on the ring. SW may only write to this register once the queue is fully configured, and clear to zero at queue initialization.
RESERVED	31:10	0x0	RSV	Reserved.

11.3.3.4.6 VF Admin Receive Queue Head - VF_ARQH (0x00007400; RW)

Field	Bit(s)	Init.	Type	Description
ARQH	9:0	0x0	RW	Receive queue head pointer. At queue initialization the software clears the head pointer and during nominal operation the Firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	Reserved.

11.3.3.4.7 VF Admin Transmit Queue Base Address High - VF_ATQBAH (0x00007800; RW)

Field	Bit(s)	Init.	Type	Description
ATQBAH	31:0	0x0	RW	Transmit descriptor base address high.

11.3.3.4.8 VF Admin Transmit Queue Base Address Low - VF_ATQBAL (0x00007C00; RW)

Field	Bit(s)	Init.	Type	Description
ATQBAL	31:0	0x0	RW	Transmit descriptor base address low. Must be 64-byte aligned.

11.3.3.4.9 VF Admin Receive Queue Length - VF_ARQLEN (0x00008000; RW)

Field	Bit(s)	Init.	Type	Description
ARQLEN	9:0	0x0	RW	Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	Reserved.
ARQVFE	28	0b	RW	VF Error. Set by FW on a PF queue when one of its VFs had an admin queue error.
ARQOVFL	29	0b	RW	Overflow Error. Set by FW when a message was lost because there was no room on the queue.
ARQCRIT	30	0b	RW	Critical Error. Set by FW when a critical error has been detected on this queue.
ARQENABLE	31	0b	RW	Enable. Set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is cleared by PFR.

11.3.3.4.10 VF Admin Transmit Tail - VF_ATQT (0x00008400; RW)

Field	Bit(s)	Init.	Type	Description
ATQT	9:0	0x0	RW	Transmit queue tail. Incremented to indicate that there are new valid descriptors on the ring. SW may 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	Reserved.

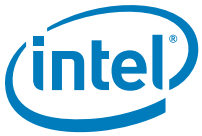
11.3.3.5 VF - LAN Transmit Receive Registers

11.3.3.5.1 Global Transmit Queue Tail - QTX_TAIL[Q] (0x00000000 + 0x4*Q, Q=0...15; RW)

Field	Bit(s)	Init.	Type	Description
TAIL	12:0	0x0	RW	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.
RSVD	31:13	0x0	RSV	Reserved.

11.3.3.5.2 Global Receive Queue Tail - QRX_TAIL[Q] (0x00002000 + 0x4*Q, Q=0...15; RW)

Field	Bit(s)	Init.	Type	Description
TAIL	12:0	0x0	RW	The Receive 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.
RSVD	31:13	0x0	RSV	Reserved.



11.3.3.6 VF - Rx Filters Registers

11.3.3.6.1 VF Queue Filter Hash Enabled Packet Type - VFQF_HENA[n] (0x0000C400 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	Description
PTYPE_ENA	31:0	0x0	RW	Packet Type Enablement of the Hash filter for the function. Bit 'm' in register 'n' enables packet type '32 x n + m' as defined in the “Packet Types for the Classification Filters” table.

11.3.3.6.2 VF Queue Filter Hash Key - VFQF_HKEY[n] (0x0000CC00 + 0x4*n, n=0...12; RW)

Field	Bit(s)	Init.	Type	Description
KEY_0	7:0	0x0	RW	Toeplitz key byte '4*n+0', where 'n' is the register index.
KEY_1	15:8	0x0	RW	Toeplitz key byte '4*n+1', where 'n' is the register index.
KEY_2	23:16	0x0	RW	Toeplitz key byte '4*n+2', where 'n' is the register index.
KEY_3	31:24	0x0	RW	Toeplitz key byte '4*n+3', where 'n' is the register index.

11.3.3.6.3 VF Queue Filter Hash LUT - VFQF_HLUT[n] (0x0000D000 + 0x4*n, n=0...15; RW)

Field	Bit(s)	Init.	Type	Description
LUT0	3:0	0x0	RW	Hash redirection LUT entry 4 x 'n', where 'n' is the register index.
RSVD	7:4	0x0	RSV	Reserved.
LUT1	11:8	0x0	RW	Hash redirection LUT entry 4 x 'n' + 1, where 'n' is the register index.
RSVD	15:12	0x0	RSV	Reserved.
LUT2	19:16	0x0	RW	Hash redirection LUT entry 4 x 'n' + 2, where 'n' is the register index.
RSVD	23:20	0x0	RSV	Reserved.
LUT3	27:24	0x0	RW	Hash redirection LUT entry 4 x 'n' + 3, where 'n' is the register index.
RSVD	31:28	0x0	RSV	Reserved.

11.3.3.6.4 VF Queue Filter Hash Region of Queues - VFQF_HREGION[n] (0x0000D400 + 0x4*n, n=0...7; RW)

Field	Bit(s)	Init.	Type	Description
OVERRIDE_ENA_0	0	0b	RW	Override Traffic Class Region for packet type '8 x n', where 'n' is the register index.
REGION_0	3:1	000b	RW	Receive queue region for packet type '8 x n', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_1	4	0b	RW	Override Traffic Class Region for packet type '8 x n + 1', where 'n' is the register index.
REGION_1	7:5	000b	RW	Receive queue region for packet type '8 x n + 1', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.

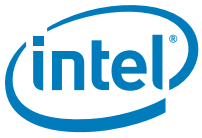
Field	Bit(s)	Init.	Type	Description
OVERRIDE_ENA_2	8	0b	RW	Override Traffic Class Region for packet type '8 x n + 2', where 'n' is the register index.
REGION_2	11:9	000b	RW	Receive queue region for packet type '8 x n + 2', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_3	12	0b	RW	Override Traffic Class Region for packet type '8 x n + 3', where 'n' is the register index.
REGION_3	15:13	000b	RW	Receive queue region for packet type '8 x n + 3', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_4	16	0b	RW	Override Traffic Class Region for packet type '8 x n + 4', where 'n' is the register index.
REGION_4	19:17	000b	RW	Receive queue region for packet type '8 x n + 4', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_5	20	0b	RW	Override Traffic Class Region for packet type '8 x n + 5', where 'n' is the register index.
REGION_5	23:21	000b	RW	Receive queue region for packet type '8 x n + 5', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_6	24	0b	RW	Override Traffic Class Region for packet type '8 x n + 6', where 'n' is the register index.
REGION_6	27:25	000b	RW	Receive queue region for packet type '8 x n + 6', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_7	28	0b	RW	Override Traffic Class Region for packet type '8 x n + 7', where 'n' is the register index.
REGION_7	31:29	000b	RW	Receive queue region for packet type '8 x n + 7', where 'n' is the register index. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.

11.3.3.7 VF - PE Registers

11.3.3.7.1 CMPE VF Error Info - VFCM_PE_ERRINFO (0x0000D800; RO)

This register reports errors for internal CM clients.

Field	Bit(s)	Init.	Type	Description
ERROR_VALID	0	0b	RO	Indicates the information in ERRINFO and ERRDATA is valid. Writing a 1 to this bit clears the contents of ERRINFO and ERRDATA. Writing a 0 to this bit has no affect.
RESERVED	3:1	000b	RSV	Reserved.
ERROR_INST	6:4	000b	RO	Indicates the internal unit that reported the error. 001b = DBL 010b = RLU 011b = RLS All other values are reserved.
RESERVED	7	0b	RSV	Reserved.
DBL_ERROR_CNT	15:8	0x0	RO	Number of DBL errors reported.
RLU_ERROR_CNT	23:16	0x0	RO	Number of RLU errors reported.
RLS_ERROR_CNT	31:24	0x0	RO	Number of RLS errors reported.



11.3.3.7.2 CMPE VF Error Data - VFCM_PE_ERRDATA (0x0000DC00; RO)

This register reports errors for internal CM clients.

Field	Bit(s)	Init.	Type	Description
ERROR_CODE	3:0	0x0	RO	Error code. 0001b = PMAT Error — Other 0010b = PMAT Error — Dummy completion 0100b = Context CRC Error All other values are reserved.
Q_TYPE	6:4	000b	RO	
RESERVED	7	0b	RSV	Reserved.
Q_NUM	25:8	0x0	RO	
RESERVED	31:26	0x0	RSV	Reserved.

11.3.4 BAR3 Registers Summary

Table 11-9. BAR3 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Block	Page
VF - MSI-X Table Registers				
0x00000000 + 0x10*n, n=0...16	VFMSIX_TADD[n]	VF MSI-X Message Address Low	VF_BAR	1492
0x00000004 + 0x10*n, n=0...16	VFMSIX_TUADD[n]	VF MSI-X Message Address High	VF_BAR	1492
0x00000008 + 0x10*n, n=0...16	VFMSIX_TMSG[n]	VF MSI-X Message Data	VF_BAR	1493
0x0000000C + 0x10*n, n=0...16	VFMSIX_TVCTRL[n]	VF MSI-X Vector Control	VF_BAR	1493
0x00002000	VFMSIX_PBA	VF MSI-X PBA Structure	VF_BAR	1493

11.3.5 Detailed Register Description - VF BAR3

11.3.5.1 VF - MSI-X Table Registers

11.3.5.1.1 VF MSI-X Message Address Low - VFMSIX_TADD[n] (0x00000000 + 0x10*n, n=0...16; RW)

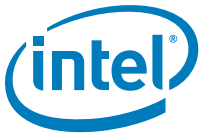
Message address for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MSIXTADD10	1:0	00b	RW	Message Address. For proper Dword alignment, software must always write zeros to these two bits. Otherwise, the result is undefined. The state of these bits after reset must be 0b. These bits are permitted to be read-only or read/write.
MSIXTADD	31:2	0x0	RW	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.

11.3.5.1.2 VF MSI-X Message Address High - VFMSIX_TUADD[n] (0x00000004 + 0x10*n, n=0...16; RW)

Message upper address for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MSIXTUADD	31:0	0x0	RW	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.



11.3.5.1.3 VF MSI-X Message Data - VFMSIX_TMSG[n] (0x00000008 + 0x10*n, n=0...16; RW)

Message data for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MSIXTMSG	31:0	0x0	RW	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.

11.3.5.1.4 VF MSI-X Vector Control - VFMSIX_TVCTRL[n] (0x0000000C + 0x10*n, n=0...16; RW)

Vector control for MSI-X table entries.

Field	Bit(s)	Init.	Type	Description
MASK	0	1b	RW	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	Reserved. After reset, the state of these bits must be 0b. 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.

11.3.5.1.5 VF MSI-X PBA Structure - VFMSIX_PBA (0x00002000; RO)

Pending bits for MSI-X PBA entries.

Field	Bit(s)	Init.	Type	Description
PENBIT	31:0	0x0	RO	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.

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

12.0 PCIe* Programming Interface

12.1 Overview

The XL710 supports the following configuration register sets:

- PCI basic configuration registers - described in Section 12.2
- PCI and PCIe capabilities in the PCI configuration space - described in Section 12.3
  - Includes the PCIe Capability Structure - described in Section 12.3.5
- PCIe capabilities residing in the PCIe Extended Configuration Space - described in Section 12.4
- SR-IOV VF configuration space - described in Section 12.5

12.1.1 Functions Mapping

The XL710 is a multi-function device with the following characteristics:

- Up to 8 physical functions (PFs) in non-ARI mode and up to 16 functions in ARI mode.
- Up to 128 SR-IOV virtual functions (VFs)
  - Each PF may be allocated a different number of VFs in the range {0,...,128} (as long as the total number of VFs does not exceed 128)

The following rules apply 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.
- The number of PFs associated with an enabled LAN port may differ among ports and may be any number in the range {1,...,8}

The number of enabled physical functions may be deducted from the PFPCI_STATUS1.FUNC_VALID bits (one per PF). See Section 4.3.4 on how enabled functions are determined.

The ARI capability allows 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 8 physical or virtual functions. Table 12-1 and Table 12-2 map physical functions to PCI Requester ID.

Table 12-1. 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 15	B,1,7	B,00001,111	PF #15

Table 12-2. 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.

12.1.2 Supported Features

Table 12-3 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 12.3 and Section 12.4 for details on specific capabilities.

Table 12-3. PCI Capabilities supported by function

PCI Capability	PFs	VFs
PCI configuration	Mandatory	Mandatory
Power Management	Yes	Yes
MSI	Yes	No
MSI-X	Yes	Yes
Vital Product Data (VPD)	Yes	No
PCI Express	Yes	Yes
Advanced Error Reporting (AER)	Yes	Yes
Device Serial Number	Yes	No (N/A)
Alternative RID Interpretation (ARI)	Yes	Yes
Single Root I/O Virtualization (SR-IOV)	Yes	No (N/A)

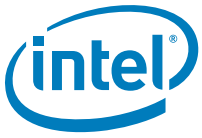


Table 12-3. PCI Capabilities supported by function

PCI Capability	PFs	VFs
TPH Requester	Yes	Yes
Access Control Services (ACS)	Yes	Yes
Secondary PCI Express	Yes	No (N/A)

12.2 PCI Configuration Space

Configuration registers are assigned one of the attributes described in the table that follows.

12.2.1 Register Attributes

The following table lists the register attributes used in this section.

RD/WR	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: 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 below 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.

12.2.2 Reset Rules

Reset of the PCI configuration space (including any capability lists) is per the PCIe specification. Several cases require special attention.

12.2.2.1 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

12.2.2.2 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

12.2.3 PCI Configuration Space Summary

Table 12-4 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 12.3. 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 11.1.1.6 for more details).

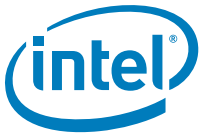


Table 12-4. PCI Configuration Space - PF

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)

12.2.4 Sharing Among PCI Functions

The XL710 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 12-5. Configuration sharing of PCI configuration space

Field	Sub-field	Shared?	Replicated?	Comments
Vendor ID	Vendor ID	x		
Device ID	Device ID		x	
Command Register	I/O Access Enable		x	Issue UR per PF if disabled
	Memory Access Enable		x	Issue UR per PF if disabled
	Bus Master Enable		x	
	Parity Error Response		x	Enables certain error reporting per PF
	SERR# Enable		x	Controls error reporting per PF
	Interrupt Disable		x	Selection of interrupt method per PF
Status Register	Interrupt Status		x	
	Capabilities List	x		Hardwired to 1b
	Data Parity Reported / Master Data Parity Error		x	Reports poisoned packets per PF

Table 12-5. Configuration sharing of PCI configuration space

Field	Sub-field	Shared?	Replicated?	Comments
	Signaled Target Abort		x	Reports Completer Abort per PF
	Received Target Abort		x	Reports receiving a Completer Abort per PF
	Received Master Abort		x	Reports receiving an UR per PF
	Signaled System Error		x	Reports Fatal / non-fatal message per PF
	Detected Parity Error		x	Reports receiving a poisoned TLP per PF
Revision Register		x		
Class Code Register			x	Per function type
Cache Line Size Register			x	Does not affect device behavior
Latency Timer		x		Hardwired to 00h in PCIe
Header Type Register		x		
BIST		x		Not supported (00h)
BARs	Memory BAR		x	
	I/O BAR		x	
	MSI-X BAR		x	See MSI-X capability
I/O BAR mapping	IOADDR, IODATA		x	
Subsystem Vendor ID		x		
Subsystem ID			x	
Expansion ROM		x		Each PF has its own BAR
Cap_Ptr Register		x		
Interrupt Line Register			x	Just store the register value
Interrupt Pin Register			x	Separate interrupt number (A-D) per PF
Min Grant		x		
Max Latency		x		

12.2.5 Mandatory PCI Configuration Registers — Except BARs

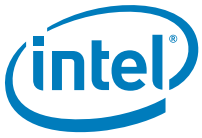
12.2.5.1 Vendor ID Register (0x0; RO)

This is a read-only register that has the same value for all PCI functions.

- Vendor ID is loaded from 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

12.2.5.2 Device ID Register (0x2; RO)

This is a read-only register that identifies individual XL710 PCI functions. All functions have the same default value of 0x154B, and can be auto-loaded from the NVM during initialization with different values for each function as well as the dummy function (See Section 4.3 for dummy function details).



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 value of each PF is loaded to the respective *PFPCI_DEVID.PF_DEV_ID* field

12.2.5.3 Command Register (0x4; RW)

Shaded bits are not used by this implementation and are hardwired to 0b. Each function has its own Command register (see Table 12-5).

Bit(s)	Initial Value	RW	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	0b	RO	Reserved.

12.2.5.4 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	Initial Value	RW	Description
2:0	0b		Reserved.
3	0b	RO	Interrupt Status.[1]
4	1b	RO	New Capabilities: Indicates that a device implements extended capabilities. The XL710 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.

Bits	Initial Value	RW	Description
8	0b	RW1C	Data Parity Reported.
10:9	00b		DEVSEL Timing – Hard wired to 0b.
11	0b	RW1C	Signaled Target Abort.
12	0b	RW1C	Received Target Abort.
13	0b	RW1C	Received Master Abort.
14	0b	RW1C	Signaled System Error.
15	0b	RW1C	Detected Parity Error.

1. The *Interrupt Status* field is a RO field that indicates that an interrupt message is pending internally to the device.

12.2.5.5 Revision Register (0x8; RO)

The default revision ID of this device is 0x00 for A0 step and 0x01 for B0 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 NVM, reflected via the *GLPCI_REVID* register.

12.2.5.6 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.

Device default value is class code of an Ethernet adapter. The value is overwritten from NVM. It is loaded to the RO PFPCI_CLASS register.

12.2.5.7 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 (Spec required).

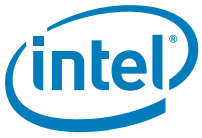
12.2.5.8 Latency Timer (0xD; RO)

Not used. Hard wired to 0b.

12.2.5.9 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.
- Functions may be disabled during the Power-On-Reset flow (through strapping pins, SMASH/CLP commands, NC-SI commands) affecting this bit. See Section 4.3.4.

12.2.5.10 Subsystem Vendor ID Register (0x2C; RO)

This value is loaded from 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_SUBSYSID* register.

12.2.5.11 Subsystem ID Register (0x2E; RO)

This value is loaded from NVM if the *GLPCI_CAPSUP.LOAD_SUBSYS_ID* bit is set. Each PF is loaded to the respective PFPCI_SUBSYSID.PF_SUBSYS_ID field.

12.2.5.12 Capabilities Pointer Register (0x34; RO)

The *Capabilities Pointer* field (Cap_Ptr) is an 8-bit field that provides an offset in the XL710's PCI configuration space for the location of the first item in the capabilities linked list. The XL710 supports this field and implements a capabilities list. Its value is 0x40, which is the address of the first entry: PCI power management.

12.2.5.13 Interrupt Line Register (0x3C; RW)

Read/write register programmed by software to indicate which of the system interrupt request lines the XL710's interrupt pin is bound to. Refer to the PCI definition for more details.

12.2.5.14 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 NVM. It can be read through the RO PFPCI_CNF register. Device default value is 0x0

If only a single function is enabled, the I*nterrupt Pin* field of the enabled function reports INTA# usage.

Reports a value of 0x0 (Function uses no legacy interrupt Message) for a dummy function

12.2.5.15 MIN_GNT and MAX_LAT (0x3E; RO)

Not used. Hard-wired to 0b.

12.2.6 Mandatory PCI Configuration Registers — BARs

12.2.6.1 Memory and IO Base Address Registers (0x10...0x27; RW)

Base Address Registers (BARs) are used to map the XL710 register space of the device functions. The XL710 has a memory BAR, an I/O BAR (optional) an MSI-X BAR as described in Table 12-6. The BARs location and sizes are described in the Table 12-8 and Table 12-7. The fields within each BAR are then described in Table 12-8.

Table 12-6. XL710 Base Address Registers Description

Mapping Windows	Mapping Description
Memory BAR	The internal registers memories and external Flash device are accessed as direct memory mapped offsets from the BAR. Software can access a Dword or 64 bits.
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 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 Pending Bit Array (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 12-7. XL710 Base Address Setting in 64bit 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)					

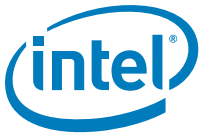


Table 12-8. Base Address Registers' Fields

Field	bits	RW	Description	
Memory and I/O Space Indication	0	RO	0b = Indicates memory space. 1b = Indicates I/O.	
Memory Type	2:1	RO	00b = 32-bit BAR 10b = 64-bit BAR This field is loaded from NVM. HW default is 64-bit. The field is exposed in the GLPCI_LBARCTRL register	
Prefetch Memory	3	RO	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	
Address Space (low register for 64-bit memory BARs)	31:4	RW	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 and Flash memory access	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

12.2.6.2 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	Bit(s)	RW	Initial Value	Description
En	0	RW	0b	1b = Enables expansion ROM access. 0b = Disables expansion ROM access.
Reserved	10:1	R	0b	Always read as 0b. Writes are ignored.
Address	31:11	RW	0b	The number of bits that are not hardwired to 0b is determined by the value of the GLPCI_LBARCTRL.EXROM_SIZE register field, loaded from NVM

12.3 Capabilities in PCI Configuration Space

The first entry of the PCI capabilities link list is pointed to by the Cap_Ptr register. Table 12-9 lists the capabilities supported by the XL710 that reside in the PCI configuration space.

Table 12-9. PCI Capabilities List

Address range	Item	Cases where capability does not exist	Next Pointer
0x40-4F	Power Management	None (always exists)	0x50 / 0xA0
0x50-6F	MSI	Dummy function	0x70 / 0xA0
0x70-8F	MSI-X	(1) PFPCI_CNF.MSI_EN is 0b (2) Dummy function	0xA0
0xA0-DF	PCI Express	None (always exists)	0xE0 / 0x00
0xE0-0xEF	VPD	(1) VPD Enable bit (GLPCI_CAPCTRL.VPD_EN) is cleared (2) Dummy function	0x00

12.3.1 PCI Power Management Capability

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 & Status	

Table 12-10 summarizes sharing of the Power Management Capability registers among the different PCI functions.

Table 12-10. Sharing the Power Management Capability registers

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Power Management Capabilities	PME_Support	x		
	D2_Support	x		
	D1_Support	x		
	AUX Current	x		
	DSI	x		
	PME Clock	x		Hardwired to 0b
	Version	x		
Power Management Control / Status	PME_Status		x	
	Data_Scale	x		
	Data_Select		x	
	PME_En		x	

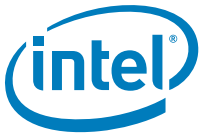


Table 12-10. Sharing the Power Management Capability registers

Field	Sub-field	Shared?	Replicated?	Comments
	No_Soft_Reset	x		
	PowerState		x	
Data Register			x	

12.3.1.1 Capability ID Register (0x40; RO)

This field equals 0x01 indicating the linked list item as being the PCI Power Management registers.

12.3.1.2 Next Pointer Register (0x41; RO)

This field provides an offset to the next capability item in the capability list. See Table 12-9 for possible values of the next pointer register.

12.3.1.3 Power Management Capabilities — PMCR (0x42; RO)

This field describes the device functionality during the power management states as listed in the following table.

Bits	Default	RW	Description
15:11	01001b	RO	PME_Support. This 5-bit field indicates the power states in which the function can generate a PME event. Condition Functionality Values are as follow: • No AUX Pwr PME at D0 and D3hot = 01001b • AUX Pwr PME at D0, D3hot, and D3cold = 11001b Note: For dummy function, this field is RO - zero.
10	0b	RO	D2_Support – The XL710 does not support the D2 state.
9	0b	RO	D1_Support – The XL710 does not support the D1 state.
8:6	000b	RO	AUX Current – Required current defined in the Data register.
5	1b	RO	DSI – The XL710 requires its device driver to be executed following a transition to the D0 uninitialized state.
4	0b	RsvdP	Reserved.
3	0b	RO	PME_Clock – Disabled. Hard wire to 0b.
2:0	011b	RO	Version – The XL710 complies with the PCI PM specification revision 1.2.

12.3.1.4 Power Management Control / Status Register — PMCSR (0x44; RW)

This register (shown in the following table) is used to control and monitor power management events in the device.

Bits	Default	RW	Description
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.
14:13	00b	RO	Data_Scale. This field indicates the scaling factor that's used when interpreting the value of the Data register. This field is loaded from NVM (through the GLPCI_PWRDATA.DATA_SCALE register) for legal values of Data_Select (0, 3, 4, 7, (and 8 for function 0)). The nominal value is 01b (indicating 0.1 watt/units) It is Reserved (00b) for any other values of Data_Select.
12:9	0000b	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.
8	0b at power up	RWS	PME_En. Writing a 1b to this register enables Wakeup.
7:4	0000b	RO	Reserved.
3	1b	RO	No_Soft_Reset. This bit is always set to 1b to indicate that the XL710 does not perform an internal reset upon transitioning 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 in order to return the XL710 to the D0 Initialized state.
2	0b	RO	Reserved for PCIe.
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

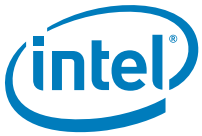
12.3.1.5 PMCSR_BSE Bridge Support Extensions Register (0x46; RO)

This register is not implemented in the XL710; values set to 0x00.

12.3.1.6 Data Register (0x47; RO)

This optional register is used to report power consumption and heat dissipation. The reported register is controlled by the *Data_Select* field in the PMCSR; the power scale is reported in the *Data_Scale* field in the PMCSR. The data for this field is loaded from NVM with a default value of 0x00. It is exposed through the GLPCI_PWRDATA register.

The values for the XL710 functions are as follows (the relevant column is selected based on the value of the Data_Select 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 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).

12.3.2 MSI Capability

This structure is enabled when the PFPCI_CNF.MSI_EN bit is set for the function

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 12-11 summarizes configuration sharing of the MSI Capability registers among the different PCI functions.

Table 12-11. Configuration sharing of the MSI Capability

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Message Control	MSI Enable		x	
	Multiple Messages Capable	x		
	Multiple Message Enable		x	
	64-bit Capable	x		
	MSI per-vector masking	x		
Message Address Low			x	
Message Address High			x	

Table 12-11. Configuration sharing of the MSI Capability

Field	Sub-field	Shared?	Replicated?	Comments
Message Data			x	
Mask Bits			x	
Pending Bits			x	

12.3.2.1 Capability ID Register (0x50; RO)

This field equals 0x05 indicating that the linked list item as being the MSI registers.

12.3.2.2 Next Pointer Register (0x51; RO)

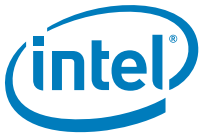
This field provides an offset to the next capability item in the capability list. See Table 12-9 for possible values of the next pointer register.

12.3.2.3 Message Control Register (0x52; RW)

Bits	Default	RW	Description
0	0b	RW	MSI Enable. 1b = Message Signaled Interrupts. The XL710 generates an MSI for interrupt assertion instead of INTx signaling.
3:1	000b	RO	Multiple Messages Capable. The XL710 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 XL710 requests a single vector in the “Multiple Messages Capable” field, SW is expected to write 000b to this field
7	1b	RO	64-bit Capable. A value of 1b indicates that the XL710 is capable of generating 64-bit message addresses.
8	1b*	RO	MSI per-vector masking. A value of 0b indicates that the XL710 is not capable of per-vector masking. A value of 1b indicates that the XL710 is capable of per-vector masking. (*) The value is loaded from NVM. It is exposed through the GLPCI_CAPSUP register
15:9	0b	RO	Reserved. Reads as 0b

12.3.2.4 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.



12.3.2.5 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.

12.3.2.6 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.

12.3.2.7 Mask Bits Register (0x60; RW)

The Mask Bits and Pending Bits registers enable software to disable or defer message sending on a per-vector basis. As the XL710 supports only one message, only bit 0 of these registers are implemented.

Bits	Default	RW	Description
0	0b	RW	MSI Vector 0 Mask. If set, the XL710 is prohibited from sending MSI messages.
31:1	0x0	RO	Reserved

12.3.2.8 Pending Bits Register (0x64; RW)

Bits	Default	RW	Description
0	0b	RO	If set, the XL710 has a pending MSI message.
31:1	0x0	RO	Reserved

12.3.3 MSI-X Capability

More than one MSI-X capability structure per function is prohibited while a function is permitted to have both an MSI and an MSI-X capability structure.

In contrast to the MSI capability structure, which directly contains all of the control/status information for the function's vectors, the MSI-X capability structure instead points to an MSI-X table structure and an MSI-X Pending Bit Array (PBA) structure, each residing in memory space.

A BAR is allocated for the MSI-X structures, described in Section 12.2.6. 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 can be either 32-bit or is 64-bit.

The number of MSI-X vectors per PF (denoted as N below) varies with the number of physical and virtual functions as depicted in Table 7-139.

The MSI-X BAR is 32KB 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 12.3.3.2) 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 per Table 7-139. The maximum value of N is 129 per PF (for the case of up to 4 PFs)
  - 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 12.3.3.2.2] 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 0x1000 (4KB) from start of BAR
  - Contains the pending bits for the PF. The PF may 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 6 (for the case of 129 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.

12.3.3.1 The 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 12-12 summarizes configuration sharing of the MSI-X Capability registers among the different PCI functions.

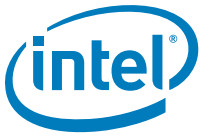


Table 12-12. Configuration sharing of the MSI-X Capability

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Message Control	Table Size	x		
Function Mask			x	
MSI-X Enable			x	
MSI-X Table Offset	Table BIR		x	
	Table Offset		x	
MSI-X Pending Bit Array	PBA BIR		x	
	PBA Offset		x	
MSI-X Table			x	
MSI-X PBA Structure			x	

12.3.3.1.1 Capability ID Register (0x70; RO)

This field equals 0x11 indicating that the linked list item as being the MSI-X registers.

12.3.3.1.2 Next Pointer Register (0x71; RO)

This field provides an offset to the next capability item in the capability list. See Table 12-9 for possible values of the next pointer register.

12.3.3.1.3 Message Control Register (0x72; RW)

Bits	Default	RW	Description
10:0	0x80 (129 vectors)	RO	Table Size. System software reads this field to determine the MSI-X Table Size N, which is encoded as N-1. N varies with the number of physical functions as depicted inTable 7-139. This field is loaded from NVM. It is reflected in the GLPCI_CNF2.MSI_X_PF_N CSR field
13:11	0x0	RO	Always returns 0b on a read. A write operation has no effect.
14	0b	RW	Function Mask. If 1b, all of the vectors associated with the function are masked, regardless of their per-vector *Mask* bit states. If 0b, each vector's *Mask* bit determines whether the vector is masked or not. Setting or clearing the MSI-X *Function Mask* bit has no effect on the state of the per-vector *Mask* bits.
15	0b	RW	MSI-X Enable. 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. If 0b, the function is prohibited from using MSI-X to request service.

12.3.3.1.4 MSI-X Table Offset Register (0x74; RW)

Bits	Default	RW	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…0x 24 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 that this field is read only.

12.3.3.1.5 MSI-X Pending Bit Array — PBA Offset (0x78; RW)

Bits	Default	RW	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…0x 24 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. This field is read only.

12.3.3.2 The PF MSI-X Table Structure

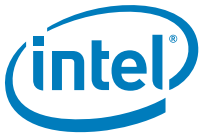
The MSI-X Table is made of two structures:

- MSI-X Vector Table
- MSI-X Pending Bits Table

Both are described below and in more details in the Programming Interface chapter

12.3.3.2.1 MSI-X Vector Table

Dword3 — MSIXTVCTRL	Dword2 — MSIXTMSG	Dword1 — MSIXTUADD	Dword0 — MSIXTADD	Entry Number	BAR 3 — Offset
Vector Control	Msg Data	Msg Upper Addr	Msg Lower Addr	0	Base (0x0000)
Vector Control	Msg Data	Msg Upper Addr	Msg Lower Addr	1	Base + 1*16
Vector Control	Msg Data	Msg Upper Addr	Msg Lower Addr	2	Base + 2*16
…	…	…	…	…	
Vector Control	Msg Data	Msg Upper Addr	Msg Lower Addr	(N-1)	Base + (N-1)*16



12.3.3.2.2 MSI-X Pending Bits Table

Field	Bit(s)	Init Val.	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 7.5.1.3 for more details

12.3.4 VPD Registers

The XL710 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.

Note: The VPD structure is available through all physical functions. As the interface is common to the functions, accessing the VPD structure of one function while an access to the NVM is in process on another function can yield to unexpected results. See Section 3.3.11 for more details

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xE0	VPD Address		Next Pointer	Capability ID (0x03)
0xE4	VPD Data			

12.3.4.1 Capability ID Register (0xE0; RO)

This field equals 0x3 indicating the linked list item as being the VPD registers.

12.3.4.2 Next Pointer Register (0xE1; RO)

Offset to the next capability item in the capability list. See Table 12-9 for possible values of the next pointer register.

12.3.4.3 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	Default	Rd/Wr	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.

12.3.4.4 VPD Data Register (0xE4; RW)

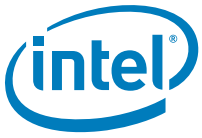
VPD read/write data.

Bits	Default	Rd/Wr	Description
31:0	X	RW	VPD Data: VPD data can be read or written through this register. The LS byte of this register (at offset 4 in this capability structure) corresponds to the byte of VPD at the address specified by the VPD Address register. The data read from or written to this register uses the normal PCI byte transfer capabilities. Four bytes are always transferred between this register and the VPD storage component. Reading or writing data outside of the VPD space in the storage component is not allowed. In a write access, the data should be set before the address and the flag is set.

12.3.5 PCIe Capability Structure

The XL710 implements the PCIe capability structure linked to the legacy PCI capability list for endpoint devices as follows:

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xA0	PCI Express Capability Register (0x0002)		Next Pointer	Capability ID (0x10)
0xA4	Device Capability			
0xA8	Device Status		Device Control	
0xAC	Link Capability			
0xB0	Link Status		Link Control	
0xB4	Reserved			
0xB8	Reserved		Reserved	
0xBC	Reserved			
0xC0	Reserved		Reserved	
0xC4	Device Capability 2			
0xC8	Reserved		Device Control 2	
0xCC	Reserved			



Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xD0	Link Status 2		Link Control 2	
0xD4	Reserved			
0xD8	Reserved		Reserved	

Table 12-13 summarizes configuration sharing of the PCIe Capability registers among the different PCI functions.

Table 12-13. Configuration sharing of the PCIe Capability

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
PCIe Capabilities		x		
Device Capabilities	Max Payload Size Supported	x		
	Phantom Functions Supported	x		Not supported
	Extended Tag Field Supported	x		
	Endpoint L0s Acceptable Latency	x		
	Endpoint L1 Acceptable Latency	x		
	Function Level Reset Capability	x		
Device Control	Correctable Error Reporting Enable		x	
	Non-Fatal Error Reporting Enable		x	
	Fatal Error Reporting Enable		x	
	Unsupported Request Reporting Enable		x	
	Enable Relaxed Ordering		x	
	Max Payload Size		x	Use minimum of all configured values. In ARI mode, use value in function 0.
	Extended Tag field Enable		x	
	Aux Power PM Enable		x	Same policy for all PFs (Logical OR of the PFs' bits)
	Enable No Snoop		x	
	Max Read Request Size		x	Use minimum of all configured values.
	Initiate Function Level Reset		x	
Device Status	Correctable Detected		x	
	Non-Fatal Error Detected		x	

Table 12-13. Configuration sharing of the PCIe Capability

Field	Sub-field	Shared?	Replicated?	Comments
	Fatal Error Detected		x	
	Unsupported Request Detected		x	
	Aux Power Detected	x		
	Transactions Pending		x	
Link Capabilities	Supported Link Speeds	x		
	Max Link Width	x		
	Active State Link PM Support	x		
	L0s Exit Latency	x		
	L1 Exit Latency	x		
	Clock Power Management	x		
	Port Number	x		
Link Control	Active State Link PM Control		x	Same policy for all PFs (Logical AND of the PFs' bits). In ARI mode, use value in function 0.
	Read Completion Boundary (RCB)		x	
	Common Clock Configuration		x	Same policy for all PFs (Logical AND of the PFs' bits) In ARI mode, use value in function 0
	Extended Sync		x	Same policy for all PFs (Logical OR of the PFs' bits)
Link Status	Current Link Speed	x		
	Negotiated Link Width	x		
	Slot Clock Configuration	x		
Device Capabilities 2	Completion Timeout Ranges Supported	x		
	Completion Timeout Disable Supported	x		
	LTR Mechanism Supported	x		
	TPH Completer Supported	x		
	Extended Fmt Field Supported	x		
	OBFF Supported	x		
Device Control 2	Completion Timeout Value		x	Completion timeout decision per PF or use the largest configured value among PFs
	Completion Timeout Disable		x	Completion timeout mechanism enabled per PF
	IDO Request Enable		x	
	IDO Completion Enable		x	
	LTR Mechanism Enable		x	PF0 only. RsvdP on other functions
	OBFF Enable		x	PF0 only. RsvdP on other functions

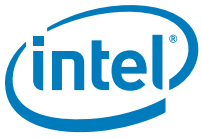


Table 12-13. Configuration sharing of the PCIe Capability

Field	Sub-field	Shared?	Replicated?	Comments
Link Capabilities 2		x		
Link Control 2		x		PF0 only. RsvdP on other functions
Link Status 2		x		

12.3.5.1 Capability ID Register (0xA0; RO)

This field equals 0x10 indicating that the linked list item as being the PCIe Capabilities registers.

12.3.5.2 Next Pointer Register (0xA1; RO)

Offset to the next capability item in the capability list. See Table 12-9 for possible values of the next pointer register.

12.3.5.3 PCIe Capabilities Register (0xA2; RO)

The PCIe Capabilities register identifies PCIe device type and associated capabilities.

Bits	Default	RW	Description
3:0	0010b	RO	Capability Version. Indicates the PCIe capability structure version. The XL710 supports PCIe version 2 (loaded from NVM). It is reflected in the GLPCI_CAPSUP register
7:4	0000b/ 1001b	RO	Device/Port Type. Indicates the type of PCIe functions. All functions are native PCI functions with a value of 0000b
8	0b	RO	Slot Implemented. The XL710 does not implement slot options. Therefore, this field is hard wired to 0b.
13:9	00000b	RO	Interrupt Message Number. This field is hard wired to 0x0 and is assumed to be irrelevant for endpoints.
15:14	00b	RO	Reserved.

12.3.5.4 Device Capabilities Register (0xA4; RO)

This register identifies the PCIe device specific capabilities.

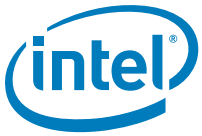
Bits	Rd/Wr	Default	Description
2:0	RO	100b	Max Payload Size Supported. This field indicates the maximum payload that the XL710 can support for TLPs. It is loaded from the NVM with a default value of 2K bytes. It is loaded via the *GLPCI_LINKCAP* register.
4:3	RO	00b	Phantom Function Supported. Not supported by the XL710.

Bits	Rd/Wr	Default	Description
5	RO	1b	Extended Tag Field Supported. Maximum supported size of the *Tag* field. The XL710 supports an 8-bit *Tag* field for all functions.
8:6	RO	011b	Endpoint L0s Acceptable Latency. This field indicates the acceptable latency that the XL710 can withstand due to the transition from L0s state to the L0 state. All functions share the same value loaded from the NVM. It is reflected in the *GLPCI_PMSUP* register. The default value of 011b denotes a maximum512 ns.
11:9	RO	110b	Endpoint L1 Acceptable Latency. This field indicates the acceptable latency that the XL710 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	RO	0b	Used to be Attention Button Present. Hard wired in the XL710 to 0b for all functions.
13	RO	0b	Used to be Attention Indicator Present. Hard wired in the XL710 to 0b for all functions.
14	RO	0b	Used to be Power Indicator Present. Hard wired in the XL710 to 0b for all functions.
15	RO	1b	Role Based Error Reporting. Hard wired in the XL710 to 1b for all functions.
17:16	RO	000b	Reserved 0b.
25:18	RO	0x00	Slot Power Limit Value. Used in upstream ports only. Hard wired in the XL710 to 0x00 for all functions.
27:26	RO	00b	Slot Power Limit Scale. Used in upstream ports only. Hard wired in the XL710 to 0b for all functions.
28	RO	1b	Function Level Reset Capability – A value of 1b indicates the function supports the optional Function Level Reset (FLR) mechanism.
31:29	RO	0000b	Reserved.

12.3.5.5 Device Control Register (0xA8; RW)

This register controls the PCIe specific parameters.

Bits	RW	Default	Description
0	RW	0b	Correctable Error Reporting Enable. Enable error report.
1	RW	0b	Non-Fatal Error Reporting Enable. Enable error report.
2	RW	0b	Fatal Error Reporting Enable. Enable error report.
3	RW	0b	Unsupported Request Reporting Enable. Enable error report.
4	RW	1b	Enable Relaxed Ordering. If this bit is set, the XL710 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	RW	000b (128 bytes)	Max Payload Size. This field sets the maximum TLP payload size for the XL710 functions. As a receiver, the XL710 must handle TLPs as large as the set value. As a transmitter, the XL710 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	RW	1b	Extended Tag field Enable. The XL710 uses 8-bit tags when this bit is set and a 5-bit tag when disabled ***Note:***
9	RW	0b	Phantom Functions Enable. Not implemented in the XL710.



Bits	RW	Default	Description
10	RWS	0b	Aux Power PM Enable. When set, enables the XL710 to draw AUX power independent of PME AUX power. The XL710 is a multi-function device, therefore allowed to draw AUX power if at least one of the functions has this bit set.
11	RW	0b	Enable No Snoop. Hard-wired to 0b as the XL710 never sets the No Snoop attribute in a TLP.
14:12	RW	010b	Max Read Request Size. This field sets maximum read request size for the XL710 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	RW	0b	Initiate FLR – A write of 1b initiates FLR to the function. The value read by software from this bit is always 0b.

12.3.5.6 Device Status Register (0xAA; RW1C)

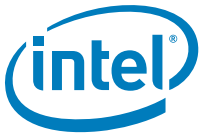
This register provides information about PCIe device specific parameters.

Bits	RW	Default	Description
0	RW1C	0b	Correctable Detected. Indicates status of correctable error detection.
1	RW1C	0b	Non-Fatal Error Detected. Indicates status of non-fatal error detection.
2	RW1C	0b	Fatal Error Detected. Indicates status of fatal error detection.
3	RW1C	0b	Unsupported Request Detected. Indicates that the XL710 received an unsupported request. This field is separate per PF. However, in case where an error cannot be associated with a PF, this bit is set in all PFs
4	RO	0b	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	RO	0b	Transactions Pending. Indicates whether the XL710 has ANY transactions pending. (transactions include completions for any outstanding non-posted request for all used traffic classes).
15:6	RsvdZ	0x00	Reserved

12.3.5.7 Link Capabilities Register (0xAC; RO)

This register identifies PCIe link-specific capabilities.

Bits	RW	Default	Description
3:0	RO	N/A	Max Link Speed – This field indicates the maximum 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	RO	0x08	Max Link Width. Indicates the maximum link width. The XL710 supports a x1, x4 and x8-link width. This field is loaded from NVM.It is reflected in the GLPCI_LINKCAP register. Defined encoding: 000000b = Reserved 000001b = x1 000010b = Reserved 000100b = x4 001000b = x8
11:10	RO	11b	Active State Link PM Support. Indicates the level of the active state of power management supported in the XL710. 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. It is reflected in the GLPCI_PMSUP register.
14:12	RO	101b	L0s Exit Latency. Indicates the exit latency from L0s to L0 state. 000b = Less than 64 ns. 001b = 64 ns – 128 ns. 010b — 128ns – 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. It is reflected in the GLPCI_PMSUP register.
17:15	RO	100b	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. It is reflected in the GLPCI_PMSUP register.
18	RO	0b	Clock Power Management - not supported



Bits	RW	Default	Description
19	RO	0b	Surprise Down Error Reporting Capable. Hard wired to 0b.
20	RO	0b	Data Link Layer Link Active Reporting Capable.
21	RO	0b	Link Bandwidth Notification Capability. Hard wired to 0b (not applicable to endpoints).
22	HwInit	1b	ASPM Optionality Compliance Software is permitted to use the value of this bit to help determine whether to enable ASPM or whether to run ASPM compliance tests.
23	RO	0b	Reserved
31:24	HwInit	0x0	Port Number. The PCIe port number for the given PCIe link. This field is set in the link training phase. The field is hard-wired to 0x0

12.3.5.8 Link Control Register (0xB0; RO)

This register controls PCIe link specific parameters.

Bits	RW	Default	Description
1:0	RW	00b	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	RsvdP	0b	Reserved
3	RW	0b	Read Completion Boundary.
4	RO	0b	Link Disable. Reserved for endpoint devices. Hard wired to 0b.
5	RO	0b	Retrain Clock. Not applicable for endpoint devices. Hard wired to 0b.
6	RW	0b	Common Clock Configuration. When set, indicates that the XL710 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	RW	0b	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, then the component must transmit the additional Ordered Sets when exiting L0s or when in Recovery
8	RO	0b	Enable Clock Power Management. Not supported - hard wired to 0b.

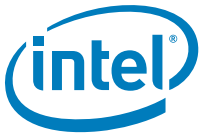
Bits	RW	Default	Description
9	RO/RsvdP	0b	Hardware Autonomous Width Disable. When set to 1b, this bit disables hardware from changing the link width for reasons other than attempting to correct an unreliable link operation by reducing link width. Not supported - function 0 is hard wired to 0b (RO). Other functions are RsvdP.
10	RO	0b	Link Bandwidth Management Interrupt Enable. Not applicable to endpoints. Hard wired to 0.
11	RO	0b	Link Autonomous Bandwidth Interrupt Enable. Not applicable to endpoints. Hard wired to 0.
15:12	RsvdP	0x0	Reserved

12.3.5.9 Link Status Register (0xB2; RO)

This register provides information about PCIe Link specific parameters.

Bits	RW	Default	Description
3:0	RO	Undefined	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	RO	Undefined	Negotiated Link Width. Indicates the negotiated width of the link. Relevant encodings for the XL710 are: 000001b = x1. 000010b = X2. 000100b = x4. 001000b = x8.
10	RO	0b	Undefined.
11	RO	0b	Link Training. Indicates that link training is in progress. This field is not applicable and is reserved for endpoint devices, and is hardwired to 0b.
12	HwInit	1b	Slot Clock Configuration. When set, indicates that the XL710 uses the physical reference clock that the platform provides at the connector. This bit must be cleared if the XL710 uses an independent clock. The *Slot Clock Configuration* bit is loaded from NVM. It is reflected in the GLPCI_PMSUP register.
13	RO	0b	Data Link Layer Link Active. Not supported in the XL710. Hard wired to 0b.
14	RO	0b	Link Bandwidth Management Status. Not supported in the XL710. Hard wired to 0b.
15	RO	0b	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.



12.3.5.10 Device Capabilities 2 Register (0xC4; RO)

This register identifies the PCIe device-specific capabilities.

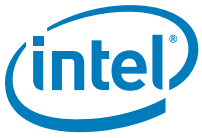
Bits	RW	Default	Description
3:0	HwInit	1111b	Completion Timeout Ranges Supported. This field indicates the XL710's support for the optional completion timeout programmability mechanism. Four time value ranges are defined: • Range A: 50 µs to 10 ms. • Range B: 10 ms to 250 ms. • Range C: 250 ms to 4 s. • Range D: 4 s to 64 s. Bits are set according to the following values to show the timeout value ranges that the XL710 supports. • 0000b = Completion timeout programming not supported. The XL710 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	RO	1b	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	RO	0b	ARI Forwarding Supported. Applicable only to Switch Downstream Ports and Root Ports; must be 0b for other function types.
10:6	RO	0x00	Not supported - hard-wired to 0x00
11	RO	0b	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	RO	00b	TPH Completer Supported - Value indicates Completer support for TPH or Extended TPH This capability is not supported.
17:14	RsvdP	0x0	Reserved
19:18	HWinit	00b	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	RO	0b	Extended Fmt Field Supported - If Set, the Function supports the 3-bit definition of the Fmt field. If Clear, the Function supports a 2-bit definition of the Fmt field. Not supported by this device
21	RO	0b	End-End TLP Prefix Supported – Indicates whether End-End TLP Prefix support is offered by a Function. Not supported by this device.
23:22	RsvdP	0b	Max End-End TLP Prefixes – Indicates the maximum number of End-End TLP Prefixes supported by this Function. Reserved for this device

Bits	RW	Default	Description
31:24	RsvdP	0x00	Reserved

12.3.5.11 Device Control 2 Register (0xC8; RW)

This register controls the PCIe specific parameters.

Bits	RW	Default	Description
3:0	RW	0x0	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
4	RW	0b	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	RO	0b	ARI Forwarding Enable.Applicable only to switch devices.
7:6	RO	00b	Not supported - hardwired to 00b
8	RW	0b	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	RW	0b	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



Bits	RW	Default	Description
10	RO / RsvdP	0b	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, then this bit is RO with a value of 0b.
12:11	RO	0x0	Reserved
14:13	RW / RsvdP	00b	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	RsvdP	0b	End-End TLP Prefix Blocking Not applicable to endpoints (RsvdP).

12.3.5.12 Link Capabilities 2 Register (0xCC)

Bits	RW	Default	Description
0	RsvdP	0b	Reserved
7:1	RO	0x01	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 Bits 7:4 - 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 reflected in the GLPCI_LINKCAP register
8	RO	0b	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.
31:9	RsvdP	0x00	Reserved

12.3.5.13 Link Control 2 Register (0xD0; RWS)

Bits	RW	Default	Description
3:0	RWS (func 0) / RsvdP (else)	0011b (func 0) 0000b (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 XL710 (as reported in the *Supported Link Speeds* field of the Link Capabilities register).
4	RWS (func 0) / RsvdP (else)	0b	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	RWS (func 0) / RsvdP (else)	0b	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	RO	0b	Selectable De-Emphasis. This bit is not applicable and reserved for endpoints.
9:7	RWS (func 0) / RsvdP (else)	000b	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	RWS (func 0) / RsvdP (else)	0b	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	RWS (func 0) / RsvdP (else)	0b	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	RWS (func 0) / RsvdP (else)	0x0	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

12.3.5.14 Link Status 2 Register (0xD2; RW)

Bits	RW	Default	Description
0	RO	0b	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: 1b -3.5 dB. 0b -6 dB. Note: same value must be reported for all functions
1	ROS/RsvdZ	0b	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	ROS/RsvdZ	0b	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	ROS/RsvdZ	0b	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	ROS/RsvdZ	0b	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	RW1C/ RsvdZ	0b	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.
15:6	RsvdZ	0x00	Reserved

12.4 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 XL710 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 XL710 supports the following PCIe extended capabilities:

Table 12-14. 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
0x140 - 0x148	Device Serial Number	NVM is not valid	Any of the below / 0x000
0x150 - 0x154	Alternative RID Interpretation (ARI)	ARI Enabled bit in NVM is set to 0b	Any of the below / 0x000
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
0x1A0 - 0x1A8	TPH Requester	- TPH Enabled bit in NVM is set to 0b - This is a dummy function	Any of the below / 0x000
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
0x1D0 - 0x1E8	Secondary PCI Express	- Secondary PCIe Enabled bit in NVM is set to 0b - Function is not function 0	0x000

12.4.1 Advanced Error Reporting Capability (AER)

The PCIe advanced error reporting capability is an optional extended capability to support advanced error reporting. The tables that follow list the PCIe advanced error reporting extended capability structure for PCIe devices.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x100	Next Capability Ptr.	Version (0x1)	AER Capability ID (0x0001)	
0x104	Uncorrectable Error Status			
0x108	Uncorrectable Error Mask			
0x10C	Uncorrectable Error Severity			
0x110	Correctable Error Status			
0x114	Correctable Error Mask			
0x118	Advanced Error Capabilities and Control Register			
0x11C... 0x128	Header Log			

Table 12-15 summarizes configuration sharing of the AER Capability registers among the different PCI functions.

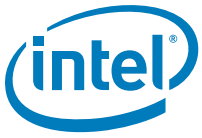


Table 12-15. Configuration sharing of the AER Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
Uncorrectable Error Status			x	
Uncorrectable Error Mask			x	
Uncorrectable Error Severity			x	
Correctable Error Status			x	
Correctable Error Mask			x	
Advanced Error Capabilities and Control	First Error Pointer		x	
	ECRC Generation Capable	x		
	ECRC Generation Enable		x	ECRC insertion is per PF
	ECRC Check Capable	x		
	ECRC Check Enable		x	See Section 3.1.2.9
Header Log			x	

12.4.1.1 Advanced Error Reporting Enhanced Capability Header Register (0x100; RO)

Bit Location	Attribute	Default Value	Description
15:0	RO	0x0001	Extended Capability ID. PCIe extended capability ID indicating advanced error reporting capability.
19:16	RO	0x2	Version Number. PCIe advanced error reporting extended capability version number.
31:20	RO	See description	Next Capability Offset. Next PCIe extended capability offset. See Table 12-14 for possible values of the next capability offset.

12.4.1.2 Uncorrectable Error Status Register (0x104; RW1CS)

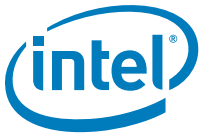
The Uncorrectable Error Status register reports error status of individual uncorrectable error sources on a PCIe device. An individual error status bit that is set to 1b indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by LAN_PWR_GOOD.

Bit Location	Attribute	Default Value	Description
0	RO	0b	Reserved
3:1	RsvdZ	000b	Reserved
4	RW1CS	0b	Data Link Protocol Error Status.
5	RO	0b	Reserved
11:6	RsvdZ	0x00	Reserved
12	RW1CS	0b	Poisoned TLP Status.
13	RW1CS	0b	Flow Control Protocol Error Status.
14	RW1CS	0b	Completion Timeout Status.
15	RW1CS	0b	Completer Abort Status.
16	RW1CS	0b	Unexpected Completion Status.
17	RW1CS	0b	Receiver Overflow Status.
18	RW1CS	0b	Malformed TLP Status.
19	RW1CS	0b	ECRC Error Status.
20	RW1CS	0b	Unsupported Request Error Status.
21	RO	0b	ACS Violation Status. Not supported. Hardwired to 0b.
25:22	RO	0x0	Not supported
31:26	RsvdZ	0b	Reserved

12.4.1.3 Uncorrectable Error Mask Register (0x108; RWS)

The Uncorrectable Error Mask register controls reporting of individual uncorrectable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. Note that there is a mask bit per bit of the Uncorrectable Error Status register.

Bit Location	Attribute	Default Value	Description
0	RO	0b	Reserved
3:1	RsvdP	000b	Reserved
4	RWS	0b	Data Link Protocol Error Mask.
5	RO	0b	Reserved
11:6	RsvdP	0x00	Reserved
12	RWS	0b	Poisoned TLP Mask.
13	RWS	0b	Flow Control Protocol Error Mask.
14	RWS	0b	Completion Timeout Mask.
15	RWS	0b	Completer Abort Mask.
16	RWS	0b	Unexpected Completion Mask.



Bit Location	Attribute	Default Value	Description
17	RWS	0b	Receiver Overflow Mask.
18	RWS	0b	Malformed TLP Mask.
19	RWS	0b	ECRC Error Mask.
20	RWS	0b	Unsupported Request Error Mask.
21	RO	0b	ACS Violation Mask. Not supported. Hardwired to 0b.
25:22	RO	0x0	Not supported
31:26	RsvdP	0b	Reserved

12.4.1.4 Uncorrectable Error Severity Register (0x10C; RWS)

The Uncorrectable Error Severity register controls whether an individual uncorrectable error is reported as a fatal error. An uncorrectable error is reported as fatal when the corresponding error bit in the severity register is set. If the bit is cleared, the corresponding error is considered non-fatal.

Bit Location	Attribute	Default Value	Description
0	RO	0b	Reserved
3:1	RsvdP	000b	Reserved
4	RWS	1b	Data Link Protocol Error Severity.
5	RO	0b	Reserved
11:6	RsvdP	0x00	Reserved
12	RWS	0b	Poisoned TLP Severity.
13	RWS	1b	Flow Control Protocol Error Severity.
14	RWS	0b	Completion Timeout Severity.
15	RWS	0b	Completer Abort Severity.
16	RWS	0b	Unexpected Completion Severity.
17	RWS	1b	Receiver Overflow Severity.
18	RWS	1b	Malformed TLP Severity.
19	RWS	0b	ECRC Error Severity.
20	RWS	0b	Unsupported Request Error Severity.
21	RO	0b	ACS Violation Severity. Not supported. Hardwired to 0b.
25:22	RO	0x0	Not supported
31:26	RsvdP	0b	Reserved

12.4.1.5 Correctable Error Status Register (0x110; RW1CS)

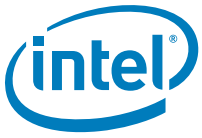
The Correctable Error Status register reports error status of individual correctable error sources on a PCIe device. When an individual error status bit is set to 1b it indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by LAN_PWR_GOOD.

Bit Location	Attribute	Default Value	Description
0	RW1CS	0b	Receiver Error Status.
5:1	RsvdZ	0b	Reserved
6	RW1CS	0b	Bad TLP Status.
7	RW1CS	0b	Bad DLLP Status.
8	RW1CS	0b	REPLAY_NUM Rollover Status.
11:9	RsvdZ	0b	Reserved
12	RW1CS	0b	Replay Timer Timeout Status.
13	RW1CS	0b	Advisory Non-Fatal Error Status.
15:14	RO	0b	Reserved
31:16	RsvdZ	0x00	Reserved

12.4.1.6 Correctable Error Mask Register (0x114; RWS)

The Correctable Error Mask register controls reporting of individual correctable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. There is a mask bit per bit in the Correctable Error Status register.

Bit Location	Attribute	Default Value	Description
0	RWS	0b	Receiver Error Mask.
5:1	RsvdP	0b	Reserved
6	RWS	0b	Bad TLP Mask.
7	RWS	0b	Bad DLLP Mask.
8	RWS	0b	REPLAY_NUM Rollover Mask.
11:9	RsvdP	0b	Reserved
12	RWS	0b	Replay Timer Timeout Mask.
13	RWS	1b	Advisory Non-Fatal Error Mask.
15:14	RO	0b	Reserved



12.4.1.7 Advanced Error Capabilities and Control Register (0x118; RO)

Bit Location	Attribute	Default Value	Description
4:0	ROS	0b	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	RO	1b	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	RWS	0b	ECRC Generation Enable. When set, ECRC generation is enabled.
7	RO	1b	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	RWS	0b	ECRC Check Enable. When set Set, ECRC checking is enabled.
9	RO	0b	Multiple Header Recording Capable. Not supported. hard-wired to 0b
10	RO	0b	Multiple Header Recording Enable Not supported. hard-wired to 0b
11	RsvdP	0b	TLP Prefix Log Present Not supported. hard-wired to 0b
15:12	RsvdP	0x0	Reserved

12.4.1.8 Header Log Register (0x11C:0x128; RO)

The header log register captures the header for the transaction that generated an error. This register is 16 bytes.

Bit Location	Attribute	Default Value	Description
127:0	ROS	0b	Header of the packet in error (TLP or DLLP).

12.4.2 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 and a proper MAC address was not loaded).

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x140	Next Capability Ptr.	Version (0x1)	Serial ID Capability ID (0x0003)	
0x144	Serial Number Register (Lower Dword)			
0x148	Serial Number Register (Upper Dword)			

Table 12-16 summarizes configuration sharing of the Serial Number Capability registers among the different PCI functions.

Table 12-16. Configuration sharing of the Serial Number Capability

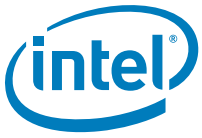
Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
Serial Number Register		x		See comment above

12.4.2.1 Device Serial Number Enhanced Capability Header Register (0x140; RO)

Bit(s)	Attribute	Default Value	Description
15:0	RO	0x0003	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	RO	0x1	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	RO	See description	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 12-14 for possible values of the next capability offset.

12.4.2.2 Serial Number Registers (0x144:0x148; RO)

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit Extended Unique Identifier (EUI-64*). The register at offset 0x144 holds the lower 32 bits and the register at offset 0x148 holds the higher 32 bits. The following figure details the allocation of register fields in the Serial Number register. The table that follows provides the respective bit definitions.



Bit(s)	Attributes	Description
63:0	RO	PCIe Device Serial Number. This field contains the IEEE defined 64-bit EUI-64*. This identifier includes a 24-bit company ID value assigned by IEEE registration authority and a 40-bit extension identifier assigned by the manufacturer.

The serial number uses the Ethernet MAC address according to the following definition:

Field	Company ID			Extension Identifier				
Order	Addr+0	Addr+1	Addr+2	Addr+3	Addr+4	Addr+5	Addr+6	Addr+7
Most Significant Byte						Least Significant Byte		
Most Significant Bit						Least Significant Bit		

The serial number can be constructed from the 48-bit Ethernet MAC address in the following form:

Field	Company ID			MAC Label		Extension identifier		
Order	Addr+0	Addr+1	Addr+2	Addr+3	Addr+4	Addr+5	Addr+6	Addr+7
Most Significant Bytes						Least Significant Byte		
Most Significant Bit						Least Significant Bit		

In this case, the MAC label is 0xFFFF.

For example, assume that the company ID is (Intel) 00-A0-C9 and the extension identifier is 23-45-67 (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						Least Significant Byte		
Most Significant Bit						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

12.4.3 Alternate Routing ID Interpretation (ARI) Capability Structure

In order 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.

The ARI capability structure is as follows:

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x150	Next Capability Ptr.	Version (0x1)	ARI Capability ID (0x000E)	
0x154	ARI Control Register		ARI Capability Register	

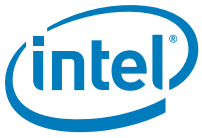
Table 12-17 summarizes configuration sharing of the ARI Capability registers among the different PCI functions.

Table 12-17. Configuration sharing of the ARI Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
ARI capability Register	Next Function Pointer		x	

12.4.3.1 PCIe ARI Header Register (0x150; RO)

Field	Bit(s)	Initial Value	Access	Description
ID	15:0	0x000E	RO	PCIe Extended Capability ID. PCIe extended capability ID for the alternative RID interpretation.
Version	19:16	1b	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 12-14 for possible values of the next capability offset.



12.4.3.2 PCIe ARI Capability Register (0x154; RO)

Field	Bit(s)	Initial Value	Access	Description
M	0	0b	RO	MFVC Function Groups Capability – 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 XL710.
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 XL710.
Reserved	7:2	0b	RsvdP	Reserved
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 may 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 XL710.
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 XL710.
Reserved	19:18	00b	RO	Reserved
FGN	22:20	0b	RO	Function Group Number. Not supported in the XL710.
Reserved	31:23	0b	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.

12.4.4 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 XL710.

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	TotalVFs (RO)		Initial VF (RO)	
0x170	Reserved	Function Dependency Link (RO)	Num VF (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)			
0x198	VF BAR5 (RO)			
0x19C	VF Migration State Array Offset (RO)			

Table 12-18 summarizes configuration sharing of the SR-IOV Capability registers among the different PCI functions.

Table 12-18. Configuration sharing of the SR-IOV Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
SR-IOV Capabilities	VF Migration Capable	x		Not supported
	ARI Capable Hierarchy Preserved		x	PF0 only. RO zero in all other functions
	VF Migration Interrupt Message Number			Not supported
SR-IOV Control	VF Enable		x	
	Memory Space Enable		x	
	ARI Capable Hierarchy	x		PF0 only. RO zero in all other functions
InitialVFs			x	
TotalVFs			x	
Num VFs			x	
Function Dependency Link			x	Each PF indicates its PF number here

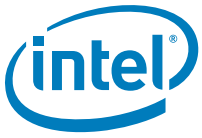


Table 12-18. Configuration sharing of the SR-IOV Capability

Field	Sub-field	Shared?	Replicated?	Comments
First VF Offset			x	
VF Stride			x	
VF Device ID			x	
Supported Page Size		x		
System Page Size			x	
VF BARs			x	

12.4.4.1 PCIe SR-IOV Header Register (0x160; RO)

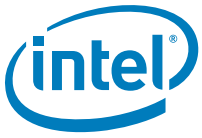
Field	Bit(s)	Initial Value	Access	Description
ID	15:0	0x0010	RO	PCIe Extended Capability ID. PCIe extended capability ID for the SR-IOV capability.
Version	19:16	0x1	RO	Capability Version. This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 0x1 for this version of the specification.
Next 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 12-14 for possible values of the next capability offset.

12.4.4.2 PCIe SR-IOV Capabilities Register (0x164; RO)

Field	Bit(s)	Initial Value	Access	Description
VFMC	0	0b	RO	VF Migration Capable — Migration Capable Device running under Migration Capable MR-PCIM. RO as zero in the XL710. Not supported in the XL710.
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
Reserved	20:2	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 XL710.

12.4.4.3 PCIe SR-IOV Control Register (0x168; RW)

Field	Bit(s)	Initial Value	Access	Description
VFE	0	0b	RW	VF Enable. 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 must be logged after VF Enable is cleared.
VF ME	1	0b	RO	VF Migration Enable. Enables / Disables VF Migration Support. Not supported in the XL710.
VF MIE	2	0b	RO	VF Migration Interrupt Enable — Enables / Disables VF Migration State Change Interrupt Not supported in the XL710.
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 1, virtual function memory space access is permitted only when VF MSE is Set. VFs shall follow the same error reporting rules as defined in the base specification if an attempt is made to access a virtual functions memory space when VF Enable is 1 and VF MSE is zero. Implementation Note: Virtual functions memory space cannot be accessed when VF Enable is zero. Thus, VF MSE is “don't care” when VF Enable is zero, however, software may choose to set VF MSE after programming the VF BARn registers, prior to setting VF Enable to 1.
VF ARI	4	0b	RW (lowest SR-IOV-enabled function) RO (else)	ARI Capable Hierarchy - Device can locate VFs in function numbers 8 to 255 of the captured bus number. If either ARI Capable Hierarchy Preserved is Set or No_Soft_Reset is 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 may scan the VF State Array. Not implemented in the XL710.
Reserved	31:17	0b	RO	Reserved



12.4.4.4 PCIe SR-IOV Initial/Total VFs Register (0x16C; RO)

Field	Bit(s)	Initial Value	Access	Description
InitialVFs	15:0	See Section 12.5.1.1	RO	InitialVFs 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 XL710 this parameter is equal to the TotalVFs in this register.
TotalVFs	31:16	See Section 12.5.1.1	RO	TotalVFs 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.

12.4.4.5 PCIe SR-IOV Num VFs Register (0x170; RW)

Field	Bit(s)	Initial Value	Access	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 XL710 there are no constraints.
Reserved	31:24	0	RO	Reserved

1. Applies to dummy function as well

12.4.4.6 PCIe SR-IOV VF RID Mapping Register (0x174; RO)

Field	Bit(s)	Initial Value	Access	Description
FVO	15:0	0x100 + 0x10 + FirstVF - PF#	RO	First VF offset defines the requestor ID (RID) offset of the first VF that is associated with the PF that contains this capability structure. The first VFs 16-bit RID is calculated by adding the contents of this field to the RID of the PF containing this field. The content of this field is valid only when *VF Enable* is set. If *VF Enable* is 0b, the contents are undefined. If the ARI *Enable* bit is set, this field changes to 0x10 + FirstVF - PF#. .This field is derived from the PF_VT_PFALLOC.FIRSTVF register field loaded from NVM

Field	Bit(s)	Initial Value	Access	Description
VFS	31:16	0x1[1]	RO	VF stride defines the requestor ID (RID) offset from one VF to the next one for all VFs associated with the PF that contains this capability structure. The next VFs 16-bit RID is calculated by adding the contents of this field to the RID of the current VF. The contents of this field is valid only when *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 12.5.1.1

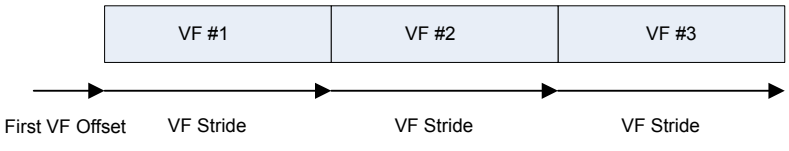


Figure 12-1. VF Stride

12.4.4.7 PCIe SR-IOV VF Device ID Register (0x178; RO)

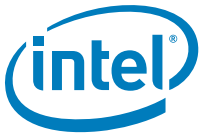
All Virtual functions have the same default value of 0x154C, and can be auto-loaded from the NVM.

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

12.4.4.8 PCIe SR-IOV Supported Page Size Register (0x17C; RO)

Field	Bit(s)	Initial Value	Access	Description
Supported page Size	31:0	0x553	RO	For PFs that supports the stride-based BAR mechanism, this field defines the supported page sizes. This PF supports a page size of 2^(n+12) if bit n is set. For example, if bit 0 is Set, the Endpoint (EP) supports 4KB page sizes. Endpoints are required to support 4 KB, 8 KB, 64 KB, 256 KB, 1 MB and 4 MB page sizes. All other page sizes are optional.



12.4.4.9 PCIe SR-IOV System Page Size Register (0x180; RW)

Field	Bit(s)	Initial Value	Access	Description
Page size	31:0	0x1	RW	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 Sizes* 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 Sizes*. 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.

12.4.4.10 PCIe SR-IOV BAR 0 — Low Register (0x184; RW)

Field	Bit(s)	Initial Value	Access	Description
Mem	0	0b	RO	0b indicates memory space.
Mem Type	2:1	10b	RO	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	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	Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size.

12.4.4.11 PCIe SR-IOV BAR 0 — High Register (0x188; RW)

Field	Bit(s)	Initial Value	Access	Description
BAR0 — MSB	31:0	0x0	RW	MSB part of BAR0.

12.4.4.12 PCIe SR-IOV BAR 2 Register (0x18C; RO)

Field	Bit(s)	Initial Value	Access	Description
BAR2	31:0	0x0	RO	This BAR is not used.

12.4.4.13 PCIe SR-IOV BAR 3 — Low Register (0x190; RW)

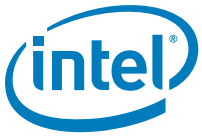
Field	Bit(s)	Initial Value	Access	Description
Mem	0	0b	RO	0b indicates memory space.
Mem Type	2:1	10b	RO	Indicates the address space size. 10b = 64-bit. This bit is loaded from the NVM. It is reflected in the GLPCI_VFSUP register
Prefetch Mem	3	0b*	RO	0b = Non-prefetchable space 1b = Prefetchable space This bit is loaded from the NVM. It is reflected in the GLPCI_VFSUP register
Memory Address Space	31:4	0x0	RW	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 page size.

12.4.4.14 PCIe SR-IOV BAR 3 — High Register (0x194; RW)

Field	Bit(s)	Initial Value	Access	Description
BAR3 — MSB	31:0	0x0	RW	MSB part of BAR3.

12.4.4.15 PCIe SR-IOV BAR 5 Register (0x198; RO)

Field	Bit(s)	Initial Value	Access	Description
BAR5	31:0	0x0	RO	This BAR is not used.



12.4.4.16 PCIe SR-IOV VF Migration State Array Offset Register (0x19C; RO)

Field	Bit(s)	Initial Value	Access	Description
BIR	2:0	0x0	RO	Indicates which PF BAR contains the VF Migration State Array. Not implemented in the XL710.
Offset	31:0	0x0	RO	Offset, relative to the beginning of the BAR of the start of the migration array.Not implemented in the XL710.

12.4.5 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.

The following table lists the TPH extended capability structure for PCIe devices.

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 12-19 summarizes configuration sharing of the TPH Requester Capability registers among the different PCI functions.

Table 12-19. Configuration sharing of the TPH Requester Capability

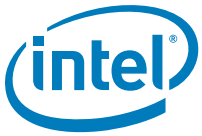
Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
TPH Requester Capability		x		
TPH Requester Control			x	
TPH ST Table			x	The Steering Table Upper fields are not supported

12.4.5.1 TPH Requester Extended Capability Header (0x1A0; RO)

Bit Location	Attribute	Default Value	Description
15:0	RO	0x17	Extended Capability ID - PCIe extended capability ID indicating TPH capability.
19:16	RO	0x1	Capability Version - PCIe TPH extended capability version number.
31:20	RO	See Description	Next Capability Offset - This field contains the offset to the next PCIe capability structure. See Table 12-14 for possible values of the next capability offset.

12.4.5.2 TPH Requester Capability Register (0x1A4; RO)

Bit Location	Attribute	Default Value	Description
0	RO	1	No ST Mode Supported - If set indicates that the Function supports the No ST Mode of operation
1	RO	0	Interrupt Vector Mode Supported - Cleared to indicates that the XL710 does not support Interrupt Vector Mode of operation
2	RO	1	Device Specific Mode - Set to indicate that the XL710 supports Device Specific Mode of operation
7:3	RsvdP	0	Reserved
8	RO	0	Extended TPH Requester Supported – Cleared to indicate that the function is not capable of generating requests with Extended TPH TLP Prefix
10:9	RO	00b	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	RsvdP	0x0	Reserved
26:16	RO	0x0	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 "00000000011" indicates a table size of 4. The value in this field is undefined since the XL710 does not support an ST Table
31:27	RsvdP	0x0	Reserved



12.4.5.3 TPH Requester Control Register (0x1A8; R/W)

Bit Location	Attribute	Default Value	Description
2:0	RW	0x0	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 XL710) 010b – Device Specific Mode 0thers – reserved for future use The default value of 000 indicates No Table mode of operation.
7:3	RsvdP	0x0	Reserved
9:8	RW	0x0	TPH Requester Enable - – Controls the ability to issue Request TLPs using either TPH or Extended TPH. Defined Encodings are: 00b – The XL710 is not permitted to issue transactions with TPH or Extended TPH as Requester 01b – The XL710 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 XL710 is permitted to issue transactions with TPH and Extended TPH as Requester (the XL710 does not issue transactions with Extended TPH). The default value of this field is 00b.
31:10	RsvdP	0x0	Reserved

12.4.6 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.

The following table lists the PCIe ACS extended capability structure for PCIe devices.

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)	

12.4.6.1 ACS Extended Capability Header (0x1B0; RO)

Bit Location	Attribute	Default Value	Description
15:0	RO	0x0D	PCI Express Extended Capability ID - PCIe extended capability ID indicating ACS capability.
19:16	RO	0x1	Capability Version - PCIe ACS extended capability version number.
31:20	RO	See Description	Next Capability Offset - See Table 12-14 for possible values of the next capability offset.

12.4.6.2 ACS Capability Register (0x1B4; RO)

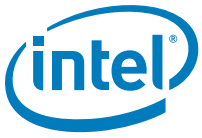
Bit Location	Attribute	Default Value	Description
0	RO	0b	ACS Source Validation (V) – Hardwired to Zero, not supported in the XL710.
1	RO	0b	ACS Translation Blocking (B) – Hardwired to Zero, not supported in the XL710.
2	RO	0b	ACS P2P Request Redirect (R) – Hardwired to Zero, not supported in the XL710.
3	RO	0b	ACS P2P Completion Redirect (C) – Hardwired to Zero, not supported in the XL710.
4	RO	0b	ACS Upstream Forwarding (U) – Hardwired to Zero, not supported in the XL710.
5	RO	0b	ACS P2P Egress Control (E) – Hardwired to Zero, not supported in the XL710.
6	RO	0b	ACS Direct Translated P2P (T) – Hardwired to Zero, not supported in the XL710.
7	RsrvP	0b	Reserved
15:8	RO	0x0	Egress Control Vector Size – Hardwired to Zero, not supported in the XL710.

12.4.6.3 ACS Control Register (0x1B6; RO)

Bit Location	Attribute	Default Value	Description
0	RO	0b	ACS Source Validation Enable (V) – Hardwired to Zero, not supported in the XL710.
1	RO	0b	ACS Translation Blocking Enable (B) – Hardwired to Zero, not supported in the XL710.
2	RO	0b	ACS P2P Request Redirect Enable (R) – Hardwired to Zero, not supported in the XL710.
3	RO	0b	ACS P2P Completion Redirect Enable (C) – Hardwired to Zero, not supported in the XL710.
4	RO	0b	ACS Upstream Forwarding Enable (U) – Hardwired to Zero, not supported in the XL710.
5	RO	0b	ACS P2P Egress Control Enable (E) – Hardwired to Zero, not supported in the XL710.
6	RO	0b	ACS Direct Translated P2P Enable (T) – Hardwired to Zero, not supported in the XL710.
15:7	RsvdP	0b	Reserved

12.4.7 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.



The following table lists the Secondary PCI Express extended capability structure for PCIe devices.

12.4.7.1 Secondary PCI Express Extended Capability

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)			
...	...			
0x1E8	...			

12.4.7.2 Header (0x1D0)

Bit Location	Attribute	Default Value	Description
15:0	RO	0x19	PCI Express Extended Capability ID – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Secondary PCI Express Extended Capability is 0019h.
19:16	RO	0x1	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	RO	See Description	Next Capability Offset – See Table 12-14 for possible values of the next capability offset.

12.4.7.3 Link Control 3 Register (0x1D4)

Bit Location	Attribute	Default Value	Description
0	RsvdP	0b	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	RsvdP	0b	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.
31:2	RsvdP	0x00	Reserved

12.4.7.4 Lane Error Status Register (0x1D8)

The Lane Error Status register consists of a 32-bit vector, where each bit indicates if the corresponding PCI Express Lane detected an error.

Bit Location	Attribute	Default Value	Description
7:0	RW1CS	0x00	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	RsvdZ	0x00	Reserved

12.4.7.5 Lane Equalization Control Register (0x1DC: 0x1F8)

The Equalization Control register consists of control fields required for per Lane equalization and number of entries in this register are sized by Maximum Link Width.

15 **0**

Lane (0) Equalization Control Register
Lane (1) Equalization Control Register
...
Lane (Maximum Link Width - 1) Equalization Control Register

Lane ((Maximum Link Width – 1):0) Equalization Control Register:

Bit Location	Attribute	Default Value	Description
3:0	RsvdP	0000b	Downstream Port Transmitter Preset For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP.
6:4	RsvdP	000b	Downstream Port Receiver Preset Hint For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP.
7	Rsvd	0b	Reserved
11:8	RO	1111b	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	HwInit/RO	111b	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	Rsvd	0b	Reserved

12.5 Virtual Functions

12.5.1 Overview

12.5.1.1 VF to PF allocation

The XL710 supports up to 128 VFsthat can be allocated to LAN ports in granularity of 16 VFs. Under this constraint, 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 16 functions the following parameters are defined in the NVM:

- PCIe* Configuration Space Control 1.SR-IOV enable - should SR-IOV be exposed for this function
- SR-IOV configuration.Next PF First VF - what is the first VF allocated to the next function (out of 128). Can be any number between 0 and 127. VF0 is assumed to be assigned to the first function whose Next PF First VF field is not zero. Next PF First VF for the last PF indicates the last VF exposed.

From these parameters the following parameters are derived in the SR-IOV capability structure (see Section 12.4.4 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]+ 16 - PF# for ARI mode and PF_VT_PFALLOC.FIRSTVF [n]+ 272 - 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.

The total number of enabled virtual functions is reflected in the GLGEN_PCIFCNCNT register.

12.5.1.2 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.

12.5.1.2.1 ARI Mode

The ARI capability allows interpretation of the ”Device” part of the Requester ID as part of the “Function” part . Thus a single device can span up to 256 functions.

The allocation of VFs to PF is flexible, there is no relationship between the PF RID and the associated VFs RID.

Table 12-20. RID Per VF - ARI Mode

VF#/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 15	B,1,7	B,00001,111	PF #15
VF 0	B,2,0	B,00010,000	
VF 1	B,2,1	B,00010,001	
VF 2	B,2,2	B,00010,010	
...	...	...	
VF 127	B,17,7	B,10001,111	Last

12.5.1.2.2 Non ARI Mode

When ARI is disabled, 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, we support only up to 8 physical functions and the RID layout is as follow:

Table 12-21. RID Per VF - Non ARI Mode

VF#/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
VF 0	B+1,2,0	B+1,00010,000	
VF 1	B+1,2,1	B+1,00010,001	
VF 2	B+1,2,2	B+1,00010,010	
...	...	...	
VF 127	B+1,17,7	B+1,10001,111	Last

12.5.1.3 Configuration Space overview

The configuration space reflected to each of the VF is a sparse version of the physical function configuration space. The following table describes the behavior of each register in the VF configuration space.

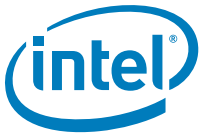


Table 12-22. VF PCIe Configuration Space

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 12.5.2.3.
	6	Status	Per VF	See Section 12.5.2.4.
	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 12.5.2.5.
	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 12.5.3.1.1.
	74	MSI-X table Address	RO	See Section 12.5.3.1.2
	78	MSI-X PBA Address	RO	See Section 12.5.3.1.3
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 Table 12.5.3.2.1.
	AA	PCIe Dev Status	per VF	See Table 12.5.3.2.2.
	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	

Table 12-22. VF PCIe Configuration Space

Section	Offset	Name	VF behavior	Notes
AER Capability	100	AER — Header	RO — 0x15010002	Next = ARI structure.
	104	AER — Uncorr Status	per VF	See Section 12.5.3.3.1.
	108	AER — Uncorr Mask	RO — 0x0	
	10C	AER — Uncorr Severity	RO — 0x0	
	110	AER — Corr Status	Per VF	See Section 12.5.3.3.2.
	114	AER — Corr Mask	RO — 0x0	
	118	AER — Cap/Ctrl	RO	See Section 12.5.3.3.3
	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	0x1A0	TPH - Header	0x1D010017	Next = ACS Structure.
	0x1A4	TPH - Capability	RO - 0x00000005	No table reported.
	0x1A8	TPH - Control	per VF	Same structure as in PF
ACS capability	0x1D0	ACS - Header	RO - 0x0001000D	Next = Last extended capability.
	0x1D4	ACS - Capability	RO - 0x00000000	

12.5.2 Mandatory Configuration Space

The IOV spec defines the configuration space of the Virtual functions as a mirror of the Physical function configuration space with the exception of some fields which 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 XL710 and describes only changes which are not a trivial implementation of the spec.

12.5.2.1 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.

12.5.2.2 Memory BARs Assignment

The IOV spec 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 & BAR3, thus only those are replicated.

The following table describes the BARs and the stride used for the VFs:

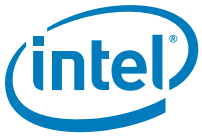


Table 12-23. VF BARs in XL710

BAR	Type	Usage	Requested Size Per VF
0	Mem	CSR space	
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

12.5.2.3 VF Command Register (0x4; RW)

Bit(s)	Initial Value	Rd/Wr	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.
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	0b	RO	RSV: Reserved

12.5.2.4 VF Status Register (0x6; RW)

Bits	Initial Value	Rd/Wr	Description
2:0	0x0	RO	RSV: Reserved
3	0b	RO	IS: Interrupt Status. Hard wired to 0b.
4	1b	RO	NC: New Capabilities. Indicates that XL710 VFs implement extended capabilities. The XL710 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	RSV: 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.

12.5.2.5 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 (i.e. all VFs of a specific PF share the same value)

12.5.3 PCI & 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 12-24. 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

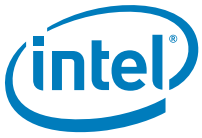


Table 12-24. Extended capabilities List

Address range	Item	Cases where capability does not exist	Next Pointer
0x150 - 0x154	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

12.5.3.1 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 12.3.3.2) 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 per Table 7-139. The maximum value of N is 17 per VF (for the case of up to 32 VFs)
  - 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 12.3.3.2.2] 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 - 8KB 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

12.5.3.1.1 VF MSI-X Control Register (0x72; RW).

Bits	Initial Value	Rd/Wr	Description
10:0	0x004	RO	TS: Table Size (N-1). N varies with the number of virtual functions as depicted in Table 7-139. This field is loaded from NVM. It is reflected in the GLPCI_CNF2.MSI_X_VF_N CSR field
13:11	0x0	RO	RSV: Reserved.
14	0b	RW	Mask: Function Mask.
15	0b	RW	En: MSI-X Enable.

12.5.3.1.2 MSI-X Address Register (0x74; RO)

Bits	Default	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…0x 24 respectively.
31:3	0x000	RO	Table 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 vectors address. The lower three BIR bits are masked off (set to zero) by software to form a 32-bit Qword-aligned offset.

12.5.3.1.3 MSI-X PBA Register (0x78; RO)

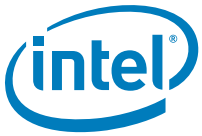
Bits	Default	Type	Description
2:0	0x3	RO	PBA BIR. Indicates which one of a function's BARs, located beginning at 0x10 in configuration space, is used to map the function's MSI-X PBA into memory space. A BIR value of three indicates that the PBA is mapped in BAR 3.
31:3	0x400	RO	PBA Offset. Used as an 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.

12.5.3.2 PCIe Capability Registers

The device control and device status registers have some fields which are specific per VF.

12.5.3.2.1 VF Device Control Register (0xA8; RW)

Bits	Rd/Wr	Default	Description
0	RO	0b	Correctable Error Reporting Enable. Zero for VFs.
1	RO	0b	Non-Fatal Error Reporting Enable. Zero for VFs.
2	RO	0b	Fatal Error Reporting Enable. Zero for VFs.
3	RO	0b	Unsupported Request Reporting Enable. Zero for VFs.
4	RO	0b	Enable Relaxed Ordering. Zero for VFs.
7:5	RO	0b	Max Payload Size. Zero for VFs.
8	RO	0b	Extended Tag field Enable.
9	RO	0b	Phantom Functions Enable. Not implemented in the XL710.
10	RO	0b	Aux Power PM Enable. Zero for VFs.
11	RO	0b	Enable No Snoop. Zero for VFs.
14:12	RO	000b	Max Read Request Size. Zero for VFs.
15	RW	0b	Initiate Function Level Reset. Specific to each VF.



12.5.3.2.2 VF Device Status Register (0xAA; RO)

Bits	Rd/Wr	Default	Description
0	R/W1C	0b	Correctable Detected. Indicates status of correctable error detection.
1	R/W1C	0b	Non-Fatal Error Detected. Indicates status of non-fatal error detection.
2	R/W1C	0b	Fatal Error Detected. Indicates status of fatal error detection.
3	R/W1C	0b	Unsupported Request Detected. Indicates that the XL710 received an unsupported request. This field is separate per PF. However, in case where an error cannot be associated with a PF, this bit is set in all PFs
4	RO	0b	Aux Power Detected. Zero for VFs.
5	RO	0b	Transactions 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	RO	0x00	Reserved

12.5.3.3 AER Registers

The following registers in the AER capability have a different behavior in a VF function.

Note that unlike the PF AER registers, these registers are not sticky since the VF is reset on FLR and on in-band reset.

12.5.3.3.1 Uncorrectable Error Status Register (0x104; RW1C)

Bit Location	Attribute	Default Value	Description
3:0	RO	0x0	Reserved
4	RO	0b	Data Link Protocol Error Status. Hard-wired to 0b
5	RO	0b	Surprise Down Error Status. Hard-wired to 0b
11:6	RO	0x0	Reserved
12	RW1C	0b	Poisoned TLP Status
13	RO	0b	Flow Control Protocol Error Status. Hard-wired to 0b
14	RW1C	0b	Completion Timeout Status.
15	RW1C	0b	Completer Abort Status.
16	RW1C	0b	Unexpected Completion Status.
17	RO	0b	Receiver Overflow Status. Hard-wired to 0b
18	RO	0b	Malformed TLP Status. Hard-wired to 0b
19	RO	0b	ECRC Error Status. Hard-wired to 0b
20	RW1C	0b	Unsupported Request Error Status — when caused by a function that claims a TLP.
21	RO	0b	ACS Violation Status. Hard-wired to 0b
31:21	RO	0x0	Reserved

12.5.3.3.2 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.

Bit Location	Attribute	Default Value	Description
0	RO	0b	Receiver Error Status. Hard-wired to 0b
5:1	RO	0x0	Reserved
6	RO	0b	Bad TLP Status. Hard-wired to 0b
7	RO	0b	Bad DLLP Status. Hard-wired to 0b
8	RO	0b	REPLAY_NUM Rollover Status. Hard-wired to 0b
11:9	RO	0x0	Reserved
12	RO	0b	Replay Timer Timeout Status. Hard-wired to 0b
13	RW1C	0b	Advisory Non-Fatal Error Status.
31:14	RO	0b	Reserved

12.5.3.3.3 Advanced Error Capabilities and Control Register (0x118; RO)

Bit Location	Attribute	Default Value	Description
4:0	ROS	0b	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	RO	0b	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	RO	0b	ECRC Generation Enable. When set, ECRC generation is enabled. Hard-wired to 0b. The PF setting applies to the VF
7	RO	0b	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	RO	0b	ECRC Check Enable.When set Set, ECRC checking is enabled. Hard-wired to 0b. The PF setting applies to the VF
9	RO	0b	Multiple Header Recording Capable. Not supported. hard-wired to 0b
10	RO	0b	Multiple Header Recording Enable Not supported. hard-wired to 0b
11	RsvdP	0b	TLP Prefix Log Present Not supported. hard-wired to 0b
15:12	RO	0x0	Reserved

13.0 Reliability, Diagnostics and Testability

13.1 Reliability

13.1.1 ECC support and ECC error flow

Memories in the XL710 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 XL710 reacts to such an event.

Reporting of ECC errors is as follows:

- The GL_CRITERRMODMASK2 register masks the reporting of ECC errors per block.
  - — If enabled, a correctable error is indicates by the ITR_CAUSE_MEM_0_STATUS.ECC_FIX bit and an uncorrectable error is indicated by the ITR_CAUSE_MEM_0_STATUS.ECC_ERR bit.
- If the ECC_ENA.ECC_ENA is set to 1b, an uncorrectable ECC error sets the ECC_ERR interrupt cause (for all PFs).
- 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.
- 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_EN bit is set to 1b, the XL710 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_EN bit is set to 1b, the XL710 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 XL710 generates a GLOBR reset

13.2 Link loopback operations

Loopback operations are supported by the XL710 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 XL710 and the system (suchas PCIe bus connections, etc.).

Loopback operation is supported as follows:

- Host-side loopback operations:
  — Tx-> Rx MAC Loopback — This loopback is performed on the internal XGMII interface of the MAC core (does not support PCS or analog cores)
  — Tx-> Rx Analog Loopback — This loopback is closed inside the analog module in the serial interface. It supports MAC, PCS and most of the analog block
- Link-side loopback operations:
  — Rx-> Tx Loopback — This loopback is performed in the internal XGMII interface (supports analog and the enabled PCS blocks). Note that 10b/8b encoding is done through this loopback, so IDLE patterns might be different between the received and transmitted data.

For the loopback to be functional, a functional link (with a partner) should be achieved (sync and alignment).

14.0 Electrical/Mechanical Specification

14.1 Introduction

This section describes the XL710 electrical and mechanical characteristics, including:

- Operating conditions
- Power Delivery
- Power Dissipation
- DC/AC specifications
- Package
- Supported devices

14.2 Operating Conditions

14.2.1 Absolute Maximum Ratings

Table 14-1. Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Units
T_{case}	Case Temperature Under Bias	TBD	TBD	°C
$T_{storage}$	Storage Temperature Range	TBD	TBD	°C
Vi	3.3V I/O input Voltage	VSS-0.5	4	V
VCC3P3/VCC3P3_A	3.3V Digital/Analog Periphery Supply Voltage	VSS-0.5	4	V
VCCD/VCCD_A	0.85V Core/Periphery/Analog Supply Voltage	VSS-0.2	1.19	V
ICC3P3	3.3V Periphery Supply Current	-	TBD	A
ICCD	0.85V Core/Periphery/Analog Supply Current	-	TBD	A

Note: 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.

Note: The VCC3P3_A and VCC0P83_A analog power supply pins are expected to be shorted to their digital compatible pins on any functional board implementation. Connecting different power sources on the analog and digital power supply pins is allowed only for debug purposed.

14.2.2 Recommended Operating Conditions

Table 14-2. Recommended Operating Conditions

Symbol	Parameter	Min	Typ	Max	Units
Ta	Operating Temperature Range Commercial (Ambient)[1]	TBD	TBD	TBD	°C
Tj	Junction Temperature	TBD	TBD	TBD	°C
VCC3P3/ VCC3P3_A	3.3V Digital/Analog Power Supply	3.14	3.3	3.46	V
VCCD/VCCD_A	0.85V Digital/Analog Power Supply[2]	0.79	0.85	0.96	V

1. See XL710 TAN (Thermal Application Notes)
2. VCCD/VCCD_A level is changed according to the Silicon-Skew Fuse

Slow/Typical Units; VCCD = 0.92V (Typ)
Fast Units: VCCD = 0.83V 9TypP
Note: During power-up the VCCD= 0.92V (Typ) for all units

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 to stated DC and AC limits is not guaranteed if conditions exceed recommended operating conditions.
- Recommended operation conditions require accuracy of power supply of ±2% relative to the nominal voltage.
- External Heat Sink (EHS) is needed.
- Refer to Chapter TBD for a description of the allowable thermal environment.

14.3 Power Delivery

14.3.1 Power Supply Specification

Table 14-3. External 3.3V Power Supply Specification

VCC3P3/VCC3P3_A (3.3V) Parameters				
Title	**Description**	**Min**	**Max**	**Units**
Rise Time	Time from 10% to 90% mark	0.1	100	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	3.3-5%	3.3+5%	V
Ripple	Maximum voltage ripple (peak to peak)	n/a	70	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

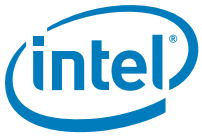


Table 14-4. External VCCD Power Supply Specification

VCCD/VCCD_A (0.85V) 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)	n/a	7120	V/S
Operational Range	Voltage range for normal operating conditions	0.75	0.96	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

Note: Inaccuracy of the Power-supply should be up to 2%

14.3.1.1 Power On/Off Sequence

The following relationships between the rise time of the different power supplies should be maintained at all times when external power supplies are in use to avoid risk of either latch-up or forward-biased internal diodes:

At power-on and after 3.3V reaches 90% of its final value, the VCCD voltage rail is allowed 100 ms to reach it's final operating voltage. Once the VCCD power supply reaches 80% of it's final value the 3.3V power supply should always be above 80% of it's final value until power down.

After 3.3V and VCCD rails reaches 65%-90% of its final value, the VCCA voltage automatically rises. The final value of VCCA is reach only after "power on internal calibration period"

For power down, it is recommended to turn off all rails at the same time and allow voltage to decay.

14.3.2 In-Rush Current

- For the VCC3P3 - the expected in-rush current is 0.5A
- For the VCC0P85 - the expected in-rush current 2.5A

14.4 Power Dissipation

The following tables list the targets for device power. The numbers listed apply to device current and power and do not include power losses on external components.

Power numbers are provides in two modes:

MAX-Power (TDP): Power of the device at Worst-Case operation conditions. power is measured on Fast-Silicon, Nominal-Voltage and Tj=110C (Tj-Max). this power should use for Thermal and Power-Supply design.

Typical-Power: power of the device at nominal operation conditions. power is measured on Typical-Silicon, Nominal-Voltage and Tj-80C.

14.4.1 MAX-Power (TDP)

Table 14-6. MAX-Power with Voltage Scaling

Link Speed	2x40G	1x40G	4x10G	2x10G	1x10G
3.3v Idd [A]	0.45	0.37	0.4	0.33	0.31
VCCD Idd [A]	6.00	5.66	5.88	5.47	5.29
Power [W]	6.47	5.92	6.20	5.63	5.41

Note: Maximum conditions: fast material, maximum operating temperature (TJ = 110C), Digital voltage value, and continuous network traffic at link speed (40G at both 1x40 and 2x40)

14.4.2 Typical -Power

Table 14-7. Typical Active Power

Link Speed	2x40G	1x40G	4x10G	2x10G	1x10G
3.3v Idd [A]	0.32	0.25	0.25	0.22	0.20
VCCD Idd [A]	3.00	2.73	2.90	2.52	2.33
Power [W]	3.82	3.34	3.49	3.04	2.80

Note: Typical conditions: typical material, TJ = 80 °C, nominal voltages and continuous network traffic at link speed.

Table 14-8. Typical Idle Power

Link Speed	2x40G	1x40G	4x10G	2x10G	1x10G
3.3v Idd [A]	0.32	0.25	0.25	0.22	0.20
VCCD Idd [A]	2.48	2.31	2.34	2.23	2.16
Power [W]	3.34	2.95	2.98	2.77	2.65

Notes: Typical conditions: typical material, TJ = 80 °C, nominal voltages and no traffic.

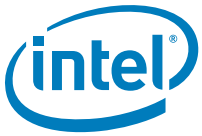


Table 14-9. Typical D3 cold with Wake-Up Enable

Link Speed	Typical material			Fast material		
Link Speed	**10G**	**1G-SGMII**	**100M**	**10G**	**1G-SGMII**	**100M**
3.3v Idd [A]	0.12	0.11	0.11	0.13	0.12	0.12
VCCD Idd [A]	1.10	1.05	11.05	1.29	1.23	1.23
Power [W]	1.41	1.33	1.33	1.62	1.53	1.53

Notes: TJ = 25 °C, nominal voltages, single link up with no traffic and PERST asserted.

Table 14-10. Typical D3 cold with no Wake-Up (link disabled)

	TYP Material	Fast Material
3.3v Idd [A]	0.11	0.14
VCCD Idd [A]	0.44	1.19
Power [W]	0.77	1.45

Notes: TJ = 25 °C, nominal voltages, no link and PERST asserted.

14.5 SVR board connectivity guidelines

The XL710 required two voltage regulators for normal operation, one for the Analog supply (VCCA) and one for the Digital supply (VCCD).

The Voltage Regulator for VCCA is implemented inside the device, and only the power-FET and the filters are need to be placed externally.

The VCCD required an external SVR, while the control on the exact voltage level is implemented inside the device and should be connected to the external SVR.

14.5.1 VCCA Connectivity

The VCCA analog rail should be connected through an external inductor to the switching node of the on die switching voltage regulator: SVR_IND (pin AD17 of the controller chip). This switching voltage regulator takes 3.3V as input and operates at a switching frequency of 1MHz.

14.5.1.1 VCCA SVAR BOM

The following are the BOM (Bill Of Material) guidelines.

- Output Bulk Capacitors (on VCCA)- Total capacitance should be 50uF. It is recommended to use four caps 10uF X7R or two caps 22uF X7R. The total ESR<1mΩ and total Z<1.5mΩ@ F=1MHz.

- Input Bulk Capacitors (on VCC3P3)- Total capacitance should be 40-100 uF. It is recommended to use minimum four caps 10uF X7R or minimum two caps 22uF X7R. The total ESR<<1mΩ and total Z<<1.5mΩ @ F=1MHz.
- Power Inductor: L= 1µH +/- 20%, Irated > 1.6A, Isat > 2.2A, Rdc < 15mΩ. The DCR should be low as possible in order to decrease the inductor losses and achieving higher SVR efficiency. The loss caused by this DCR is P=I^2 R and decreases the efficiency by (I_out^2×RDC)/P_in (%). It is recommended to use Coilcraft inductor XFL4020-102ME or other equivalent. Shielded inductor is recommended better EMI performance.
- Decoupling low ESL capacitors (Optional). For avoiding high frequency noises on input/output there is option to use low ESL caps 10nF or 100nF close to the SVR balls. Using or not depend on the specific board layout and specific system requirements

14.5.1.2 VCCA Bard connectivity

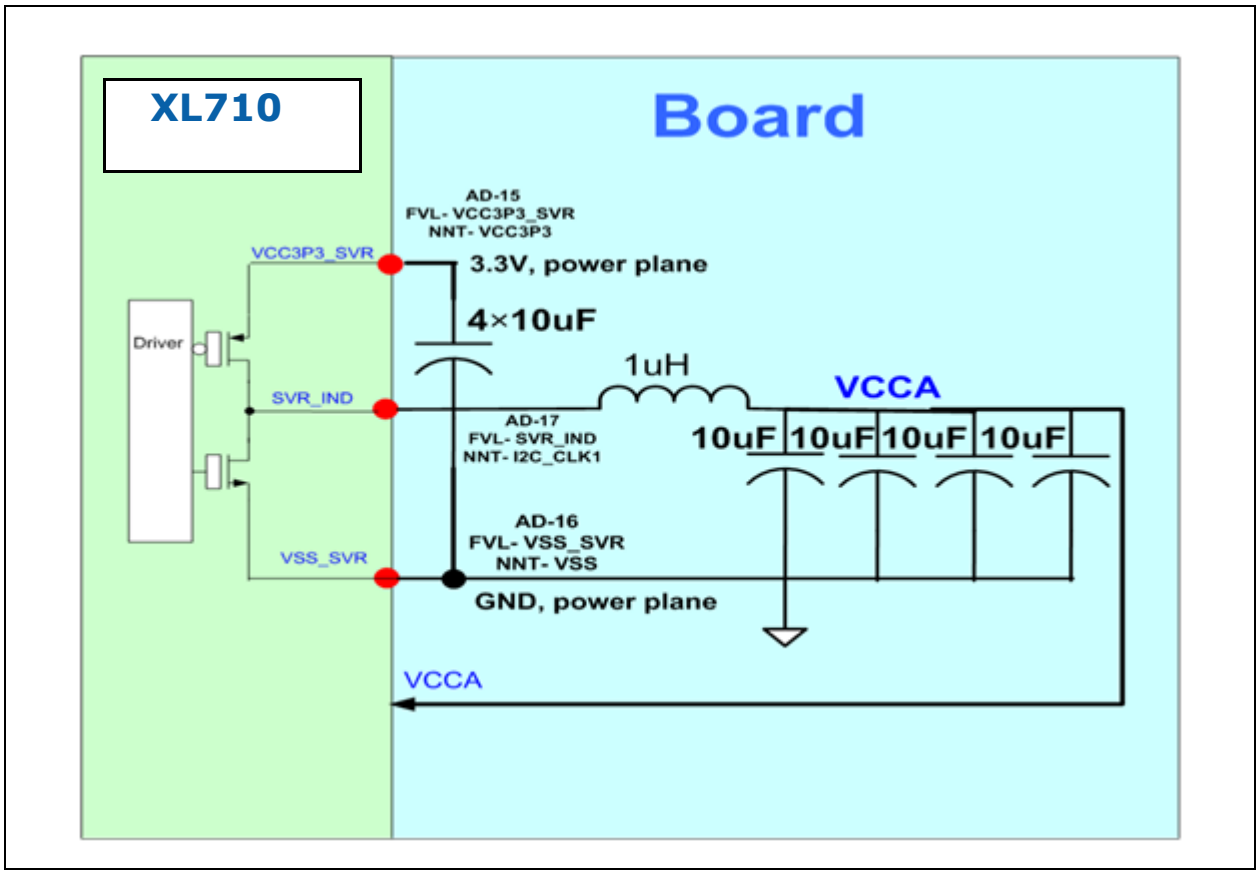


Figure 14-1. XL710 Analog SVR board connectivity schematic illustration

14.5.1.3 VCCA Layout Guideline

All SVR components should be located on the top layer and close as possible to their balls.

The connection between the SVR traces/planes to VCCA, GND and VCC3p3 should be out of SVR area, far from the balls on the other side (See Figure 14-2).

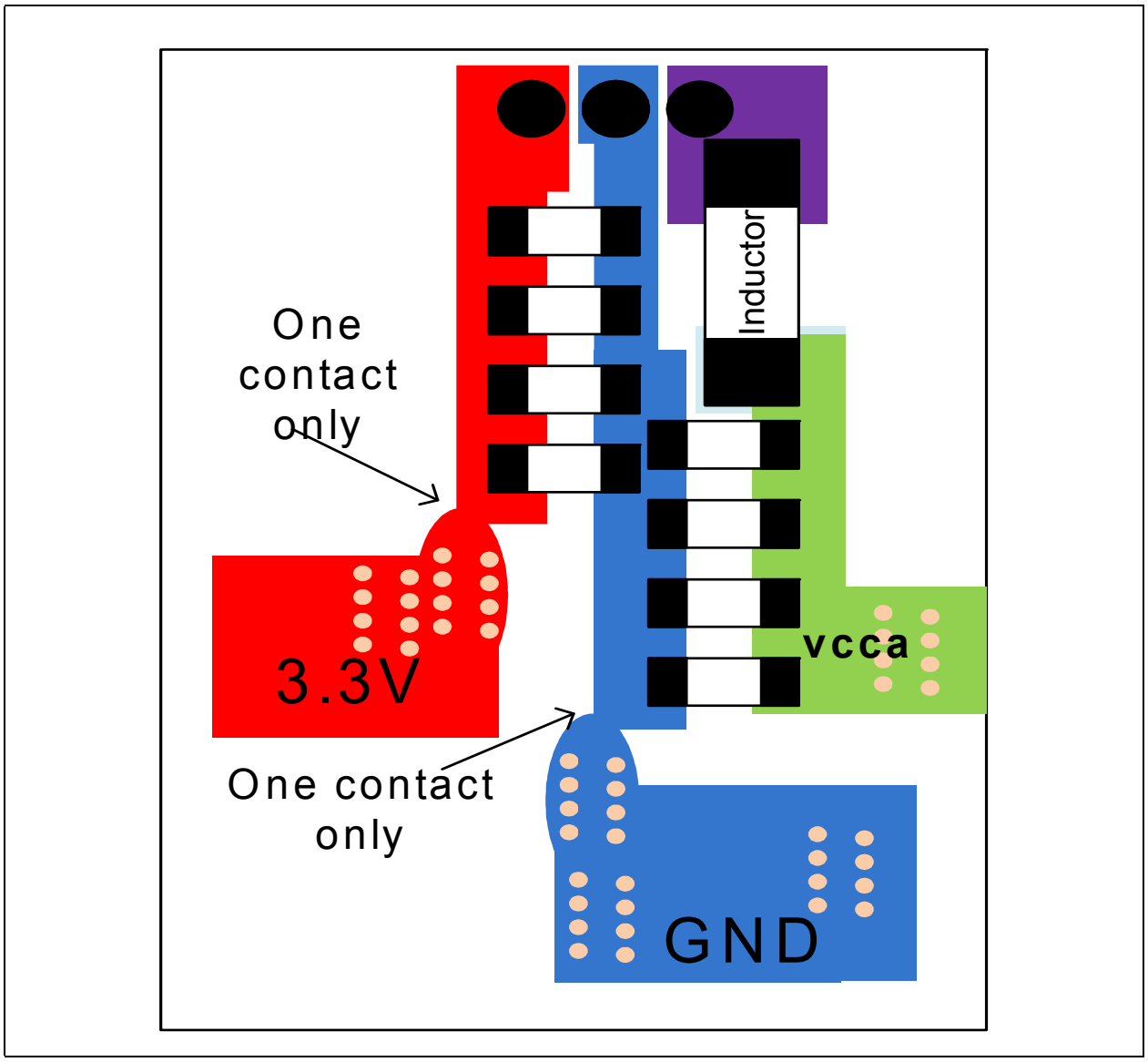


Figure 14-2. Analog SVRLayout Guidelines (VCCA)

14.5.2 VCCD connectivity

To improve power consumption XL710 implements a voltage scaling approach based on process corners. For this to work efficiently the regulator needs to ensure a 2% or tighter regulation of the VCCD power rail.

There are two voltage scaling mechanisms: one through the on die voltage sensing path and one through an output signal that can be used to switch the voltage regulator's output between two pre-defined levels.

SVR for VCCD supply: Accuracy of VCCD should be 2% or tighter, to ensure that the SVR accuracy should be 1% or better.

Reference voltage should be low enough to enable adjusting the output voltage down to 0.75V.

14.5.2.1 External SVR with the On-Die sensing control

In this option the voltage level of the external SVR is controlled by on-die sensing circuit. Voltage of the external SVR should be 0.75V, and the actual voltage inside the chip is between 0.96V - 0.81V, based on the sensing control.

Voltage value at power-Up is 0.92V

SVR with Vref < 0.75:

The feedback divider should be dimensioned such that the voltage-drop across R1 andR2 equals 0.75V. Therefore the resistor values can be calculated using the following equation

$$R1 = \frac{0.75V - Vref}{Vref} xR2$$

Figure 14-3. Resistor values

To minimize the current sourced from the EXT_SVR_SENSE_P pin and minimize the resulting accuracy

issues an extra resistor R4 should be added between the EXT_SVR_SENSE_P pin and the core voltage supply of the external switching voltage regulator.

The value of R4 can be calculate using the following equation:

$$R1 = \frac{Vintvcc - 0.75V}{Vref} xR2$$

Figure 14-4. R4 value

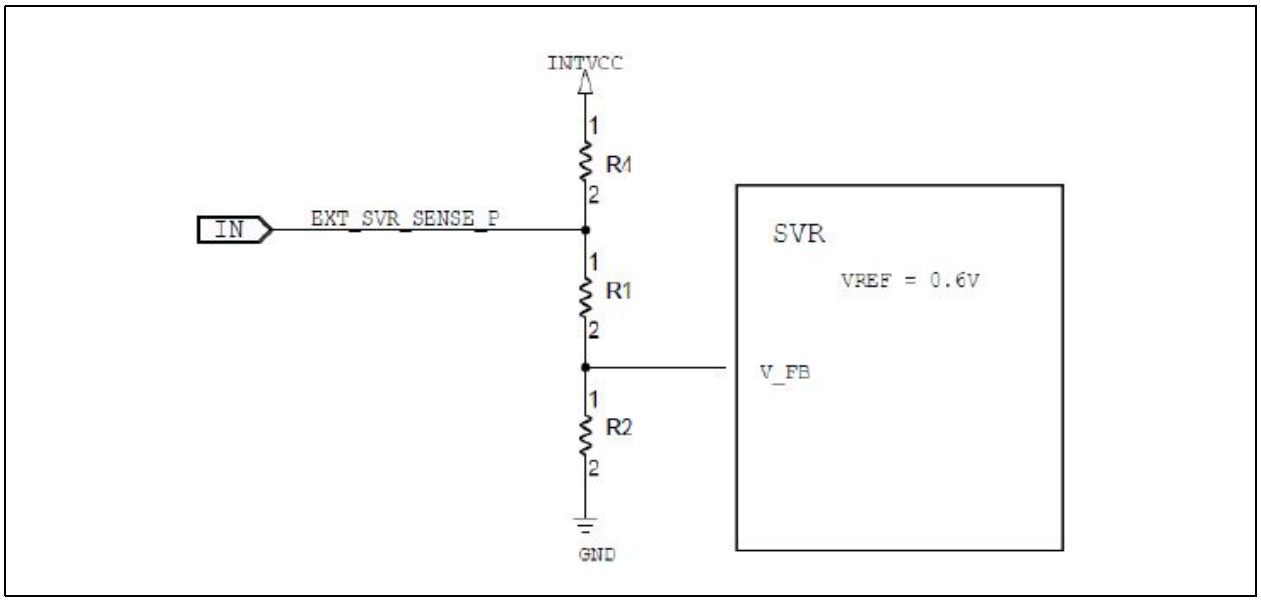


Figure 14-5. VCCD SVR connectivity, Vref < 0.75V: example schematic1 (Single-ended sense signal)

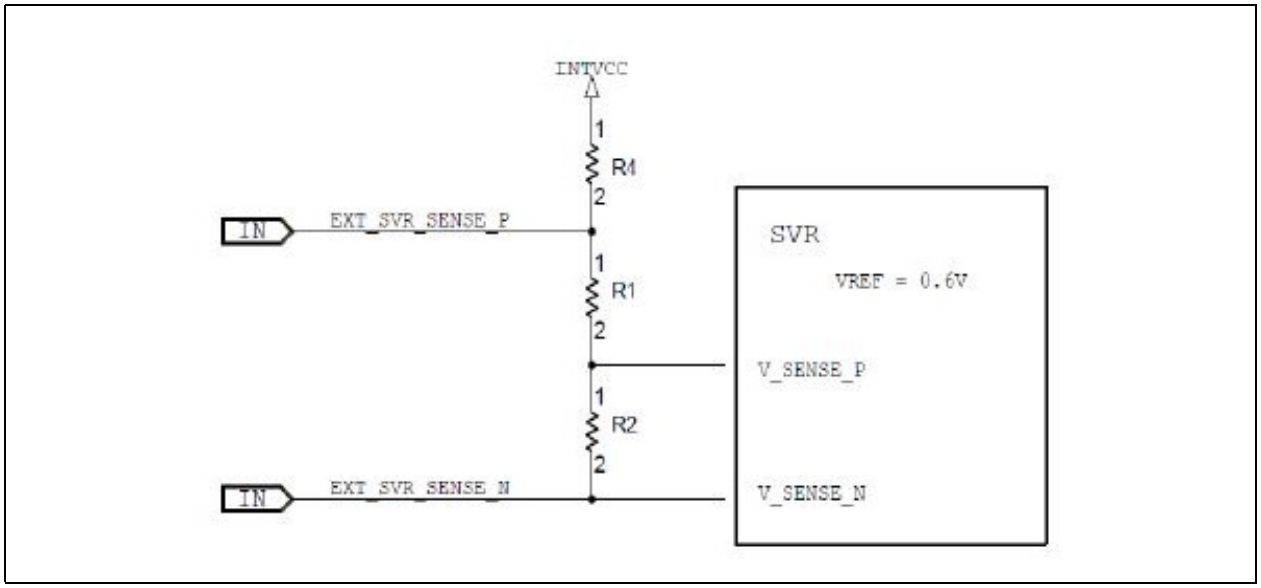


Figure 14-6. VCCD SVR connectivity, Vref < 0.75V: example schematic2 (Differential sense signals)

SVR with Vref > 0.75:

This configuration is only possible if the above mentioned extra resistor R4 is added between the EXT_SVR_SENSE_P pin and the core voltage supply of the external switching voltage regulator.

The feedback divider should be dimensioned such that the voltage-drop across R2 equals 0.75V, the combined voltage-drop across R1 and R2 equals Vref of the regulator.

The resistor values can be calculated using the following equations:

$$R1 = \frac{Vref - 0.75V}{0.75V} xR2$$

Figure 14-7. R1 Value

$$R4 = \frac{intvccV - Vref}{Vref} x(R1 + R2)$$

Figure 14-8. R4 Value

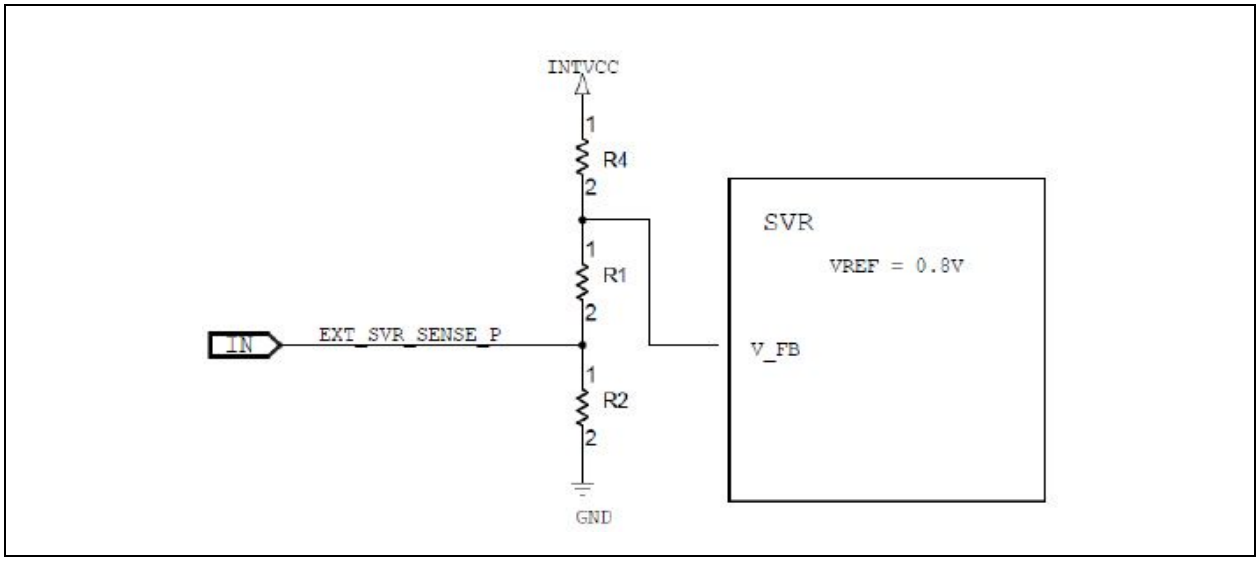


Figure 14-9. VCCD SVR connectivity, Vref > 0.75V: example schematic1 (Single-ended sense signal)

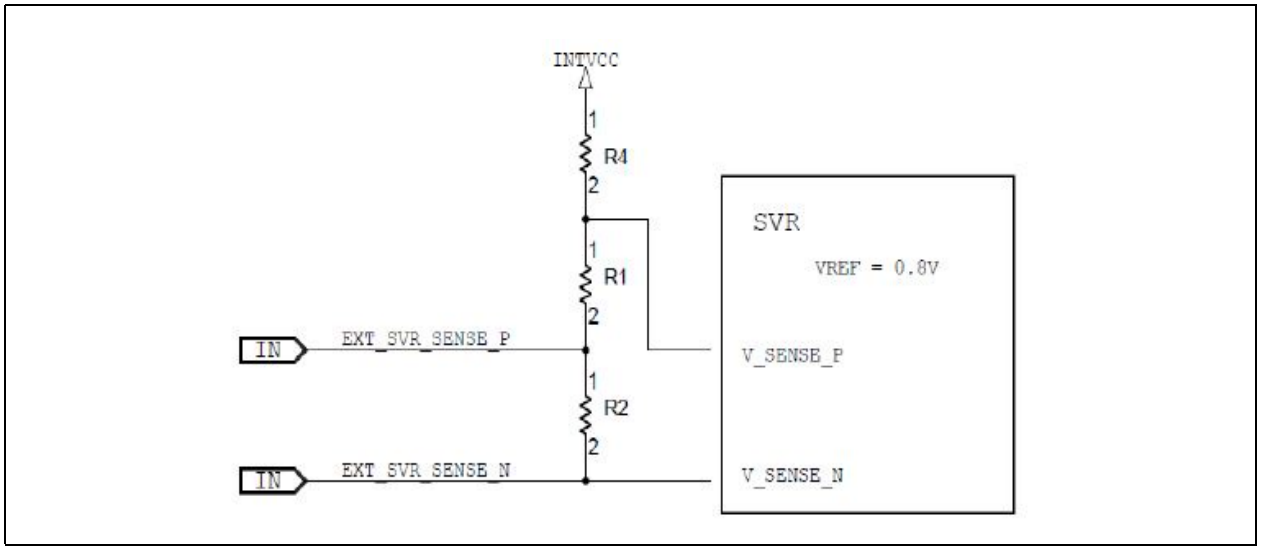


Figure 14-10. VCCD SVR connectivity, Vref > 0.75V: example schematic2 (Differential sense signals)

14.6 DC/AC Specification

14.6.1 Digital I/O DC Specifications

Table 14-11. Digital Functional 3.3V I/O DC Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Units	Note
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	3.45	V	
VIL	Input Low Voltage		-0.3	0.8	V	
Iil	Input Current[1]	VI=3.3V / 0V		10	µA	
PU	Internal pull-up		54	110	KΩ	
Cin	Pin capacitance		1.5	1.7	pF	[1]

1. Input Leakage Current

14.6.1.1 Open Drain I/O DC Specification

This section applies to SMBD, SMBCLK, SMBALRT _N, PE_WAKE_N and MDIO[3:0].

Table 14-12. Open Drain I/O DC Characteristics

Symbol	Parameter	Condition	Min	Max	Units	Note
Vih	Input High Voltage		2.0	3.45	V	
Vil	Input Low Voltage		-0.3	0.8	V	
Ileakage	Output Leakage Current	0 ≤ Vin ≤ VCC3P3 max		10	µA	[1]
Vol	Output Low Voltage	@ Ipullup = 4 mA	0.4		V	
Cin	Input Pin Capacitance		2.9	3.3	pF	[2]
Ioffsmb	Input leakage Current	VCC3P3 off or floating		10	µA	[1]
IOL	Output Current Low		6		mA	
Pull_Up	Total equivalent pull up resistance per OD output		5K		ohm	

1. Device must meet this specification whether powered or un-powered.

2. C load should be calculated according to the external pull-up resistor and the frequency.

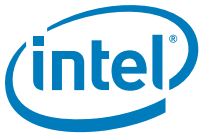
The buffer specification meets the SMBus specification requirements defined at: www.smbus.org.

14.6.1.2 NC-SI I/O DC Specification

Table 14-13. NC-SI I/O DC Characteristics

Parameter	Symbol	Conditions	Min.	Typ.	Max	Units
Bus High Reference	Vref[1]		3.0	3.3	3.6	V
Signal Voltage Range	Vabs		-0.3		3.765	V
Input Low Voltage	Vil				0.8	V
Input High Voltage	Vih		2.0			V
Output Low Voltage	Vol	Iol = 4mA, Vref = $Vref_{min}$	0		0.4	V
Output High Voltage	Voh	Iol = -4mA, Vref = $Vref_{min}$	2.4		Vref	V
Input High Current	Iih	Vin = 3.6V, Vref = 3.6V	0		200	µA
Input Low Current	Iil	Vin = 0V, Vref = $Vref_{min}$ to $Vref_{max}$	-20		0	µA
Input High Current	Ioh		12			mA
Input Low Current	Iol		12			mA
Clock Midpoint Reference Level	Vckm				1.4	V
Cin	Pin capacitance		1.5	1.7	pF	[1]
Leakage Current for Output Signals in High-Impedance State	Iz	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 may have internal mechanisms that determine the operating reference for the sideband interface that are different from the devices overall power supply inputs. Vref is a reference point that is used for measuring parameters such as overshoot and undershoot and for determining limits on signal levels that are generated by a device. In order to facilitate system implementations, a device must



provide a mechanism (e.g. a power supply pin, internal programmable reference, or reference level pin) to allow Vref to be set to within 20 mV of any point in the specified Vref range. This is to enable a system integrator to establish an interoperable Vref level for devices on the sideband interface. Although the NC-SI spec define the $Vref_{max}$ up to 3.6V, XL710 supports the $Vref_{max}$ up to 3.46V (3.3V +5%).

14.6.2 Digital I/F AC Specifications

14.6.2.1 Digital I/O AC Specifications- TBD

Table 14-14. Digital Functional 3.3V I/O AC Electrical Characteristics

Parameters	Description	Min	Max	Condition	Note
F_{max}	Maximum Operating Frequency				
Tor	Output Rise Time				
Tof	Output Fall Time				

Table 14-15. Digital Test Port 3.3V I/O AC Electrical Characteristics

Parameters	Description	Min	Max	Condition	Note
F_{max}	Maximum Operating Frequency				
Tor	Output Rise Time				
Tof	Output Fall Time				

14.6.2.2 SMBus and I^2C AC Specifications

The XL710 meets the SMBus AC specification as defined in SMBus specification version 2, section 3.1.1 (http://www.smbus.org/specs/) and the I^2C specification.

The XL710 also supports a 400 KHz SMBus (as a slave) and meets the specifications listed in the following table:

Table 14-16. Support for 400 KHz SMBus

Symbol	Parameter	Min	Typ	Max	Units
F_{SMB}	SMBus Frequency	10		400	KHz
T_{BUF}	Time Between Stop and Start	1.44[1]			µs
$T_{HD,STA}$	Hold Time After Start Condition. After This Period, the First Clock is Generated.	0.48[1]			µs
$T_{SU,STA}$	Start Condition Setup Time	1.6[1]			µs
$T_{SU,STO}$	Stop Condition Setup Time	1.76[1]			µsS

Table 14-16. Support for 400 KHz SMBus

Symbol	Parameter	Min	Typ	Max	Units
$T_{HD,DAT}$	Data in Hold Time	0.32			µs
$T_{SU,DAT}$	Data in Setup Time	0.1			µs
T_{LOW}	SMBClk Low Time	0.8[1]			µs
T_{HIGH}	SMBClk High Time	1.44[1]			µs

1. The actual minimum requirement has to be less. Many of these values are below the minimums specified by the SMBus

specification

Notes:
1. Table 14-16 applies to SMBD, SMBCLK, SCL0, SDA0, SCL1 and SDA1.

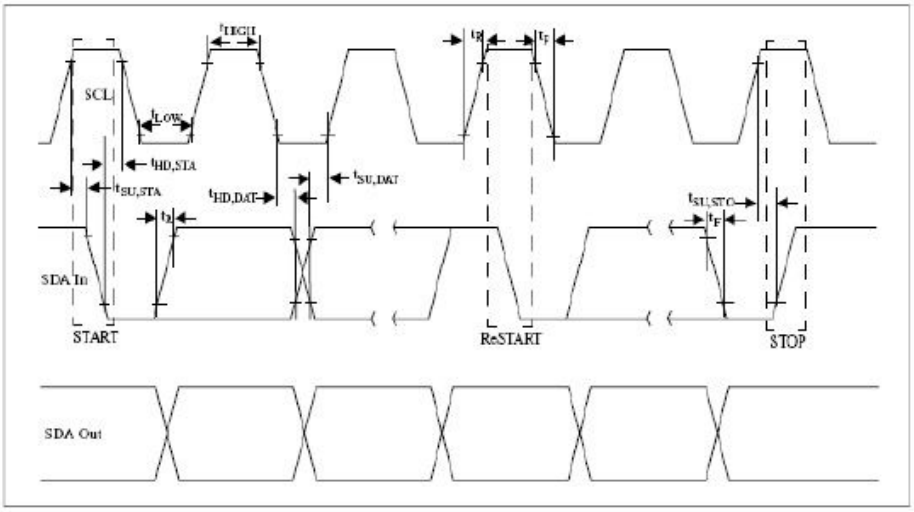


Figure 14-11. SMBus I/F Timing Diagram

14.6.2.3 FLASH AC Specification

The XL710 is designed to support a serial Flash. Applicable over recommended operating range from Ta =

0 ^{o}C to +70 ^{o}C, VCC3P3 = 3.3V, Cload = 16 pF (unless otherwise noted). For Flash I/F timing specifications, see Table 14-17 and Figure 14-12.

Table 14-17. Flash I/F Timing Parameters

Symbol	Parameter	Min	Typ	Max	Units	Note
t_{SCK}	FLSH_SCK Clock Frequency	0	12.5	15	MHz	[2]
t_{RI}	FLSH_SO Rise Time		2.5	20	ns	
t_{FI}	FLSH_SO Fall Time		2.5	20	ns	
t_{WH}	FLSH_SCK High Time	20	50		ns	[1]
t_{WL}	FLSH_SCK Low Time	20	50		ns	[1]
t_{CS}	FLSH_CE_N High Time	25			ns	
t_{CSS}	FLSH_CE_N Setup Time	25			ns	
t_{CSH}	FLSH_CE_N Hold Time	25			ns	
t_{SU}	Data-in Setup Time	5			ns	
t_{H}	Data-in Hold Time	5			ns	
t_{V}	Output Valid			20	ns	
t_{HO}	Output Hold Time	0			ns	
t_{DIS}	Output Disable Time			100	ns	
t_{EC}	Erase Cycle Time per Sector			1.1	Seconds	
t_{BPC}	Byte Program Cycle Time		60	100	µs	

Notes:
1. 50% duty cycle.
2. Clock is either 25 MHz or 26.04 MHz divided by 2.
3. Table 14-17 applies to FLSH_SI, FLSH_SO, FLSH_SCK and FLSH_CE_N.

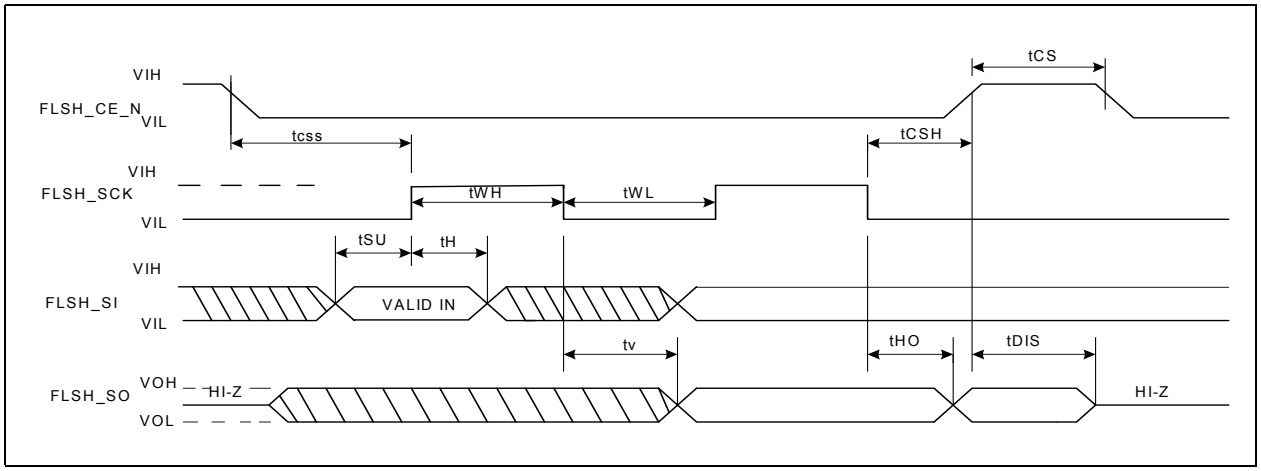


Figure 14-12. Flash I/F Timing Diagram

14.6.2.4 NC-SI AC Specifications

The XL710 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 14-18 and Figure 14-13.

Table 14-18. NC-SI Interface AC Specifications

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Units	Notes
REF_CLK Frequency				50	50+100 ppm	MHz	
REF_CLK Duty Cycle			35		65	%	2
Clock-to-Out (10pF<=cload<=50 pF)	Tco		2.5		12.5	ns	1,3
Skew Between Clocks	Tskew				1.5	ns	
TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER Data Setup to REF_CLK Rising Edge	Tsu		3			ns	3
TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER Data Hold From REF_CLK Rising Edge	Thd		1			ns	3
Signal Rise/Fall Time	Tr/Tf		1		6	ns	4
REF_CLK Rise/Fall Time	Tckr/Tckf		0.5		3.5	ns	5
Interface Power-Up High Impedance Interval	Tpwrz		2			µs	
Power Up Transient Interval (recommendation)	Tpwrt				100	ns	
Power Up Transient Level (recommendation)	Vpwrt		-200		200	mV	
Interface Power-Up Output Enable Interval	Tpwre				10	ms	
EXT_CLK Startup Interval	Tclkstrt				100	ms	

Notes:
1. This timing relates to the output pins timing while Tsu and Thd relate to timing at the input pins.
2. REF_CLK duty cycle measurements are made from Vckm to Vckm. Clock skew Tskew is measured from Vckm to Vckm of two NC-SI devices and represents maximum clock skew between any two devices in the system.
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 14-13). The middle points (50% of Vref) are marked as Vckm and Vm for clock and data, respectively.
5. A serial 33Ohm resistor might be required in case of very short trace on the board.

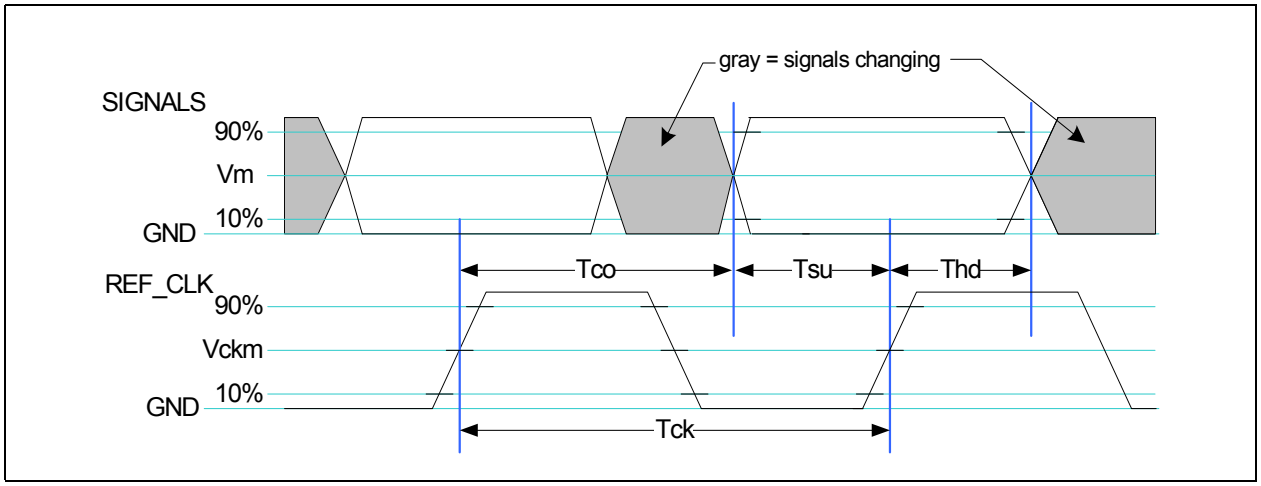


Figure 14-13. NC-SI AC Timing Diagram

14.6.2.5 JTAG AC Specification

The XL710 is designed to support the IEEE 1149.1 standard. The following timing specifications are applicable over recommended operating range from Ta = 0 °C to +70 °C, VCC3P3 = 3.3V, Cload = 16 pF (unless otherwise noted). For JTAG I/F timing specifications, see Table 14-19 and Figure 14-14.

Table 14-19. JTAG I/F Timing Parameters

Symbol	Parameter	Min	Typ	Max	Units	Note
t_{JCLK}	JTCK Clock Frequency			10	MHz	
t_{JH}	JTMS and JTDI HoldTime	10			ns	
t_{JSU}	JTMS and JTDI Setup Time	10			ns	
t_{JPR}	JTDO Propagation Delay			15	ns	

Notes:
1. Table 14-19 applies to JTCK, JTMS, JTDI and JTDO.
2. Timing measured relative to JTCK reference voltage of VCC3P3/2.

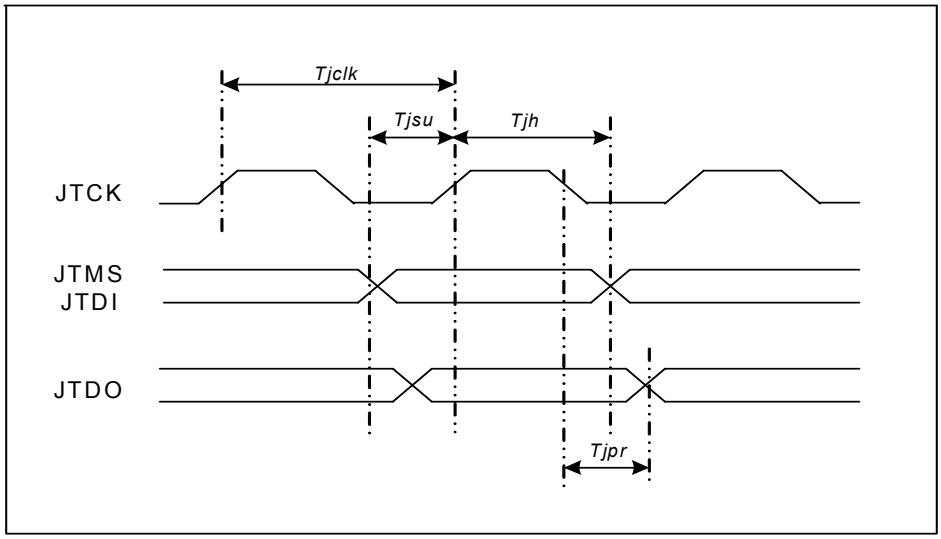


Figure 14-14. JTAG AC Timing Diagram

14.6.2.6 MDIO AC Specification

The XL710 is designed to support the MDIO specifications defined in IEEE 802.3 clause 22. The following timing specifications are applicable over recommended operating range from Ta = 0 ^{o}C to +70 ^{o}C, VCC3P3 = 3.3V, Cload = 16 pF (unless otherwise noted). For MDIO I/F timing specifications, see Table 14-20, Figure 14-15 and Figure 14-16.

Table 14-20. MDIO I/F Timing Parameters

Symbol	Parameter	Min	Typ	Max	Units	Note
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:
1. Table 14-20 applies to MDIO0, MDC0, MDIO1 and MDC1.
2. Timing measured relative to MDC reference voltage of 2.0V (Vih).

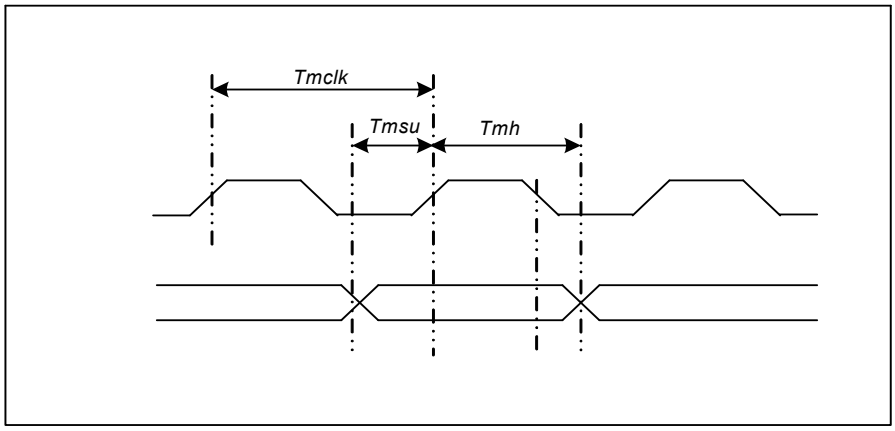


Figure 14-15. MDIO Input AC Timing Diagram

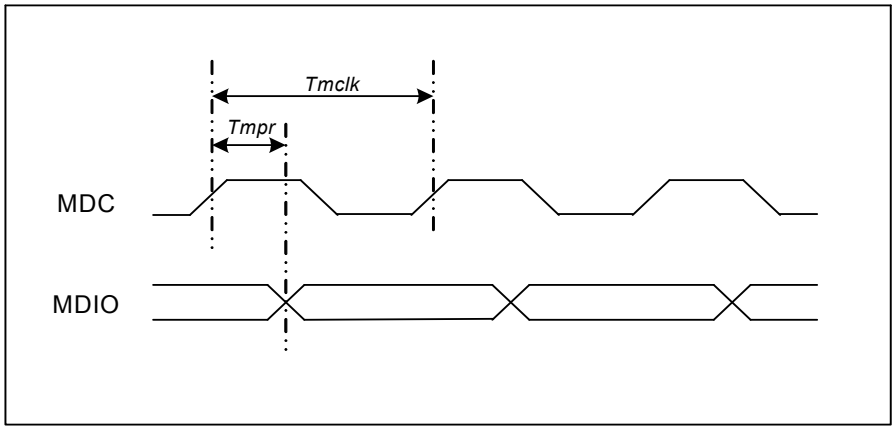


Figure 14-16. MDIO Output AC Timing Diagram

14.6.2.7 Reset Signals

For power-on indication the XL710 can either use an internal power-on circuit, which monitors the VCCD power supply, or an external reset using the LAN_PWR_GOOD pin. The POR_BYPASS pin defines the reset source (when high, the device uses the LAN_PWR_GOOD pad as power-on indication).

Note: The timing between the power up sequence and the different reset signals when using the internal power indication.

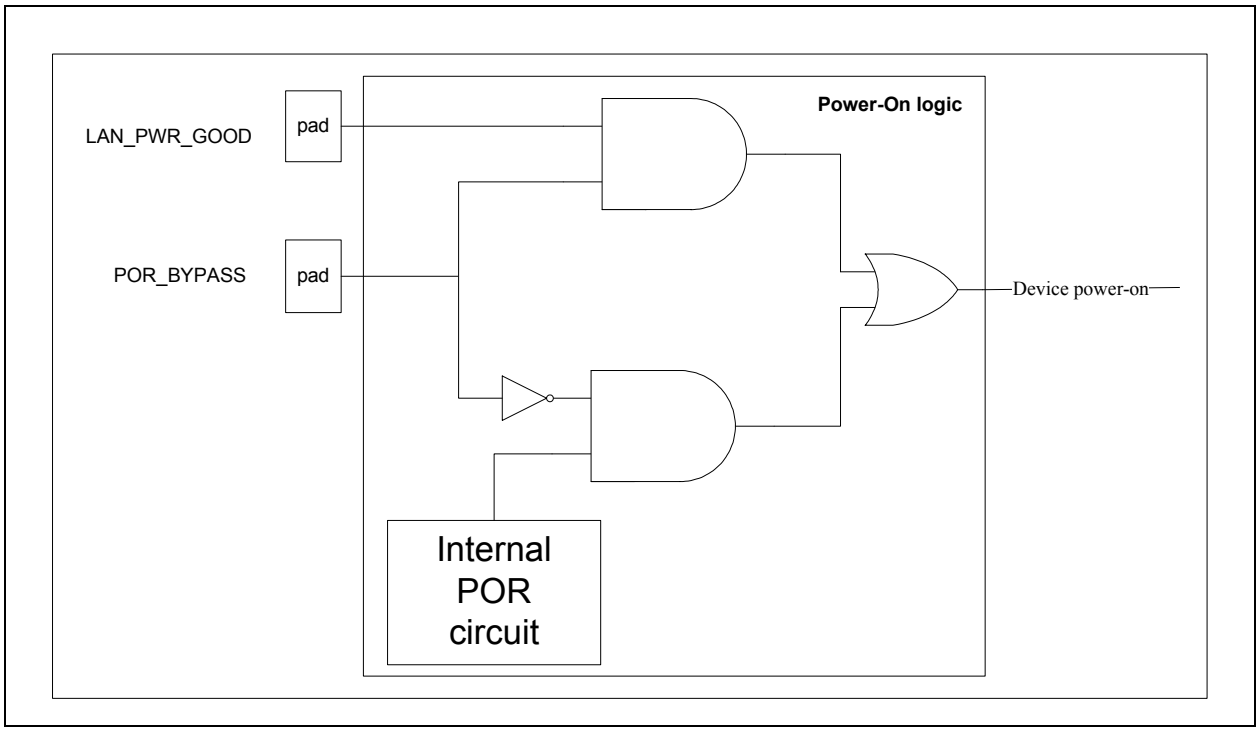


Figure 14-17. Schema of power on logic

14.6.2.7.1 Power-On Reset BYPASS

When asserting the POR_BYPASS pad the XL710 uses the LAN_PWR_GOOD pin as power-on indication. Otherwise the XL710 uses an internal power on detection circuit in order to generate the internal power on reset signal (See diagram below).

Table 14-21 below summarizes the timing for the external power-on signal:

Table 14-21. External Reset specification

Symbol	Title	Description	Min	Max	Units
Tlpgw	LAN_PWR_GOOD minimum width	Minimum width for LAN_PWR_GOOD deassertion	40	N/A	mS
Tpwr_lpg	Power-Ramp to LAN_PWR_GOOD	Time from Power-Up of all power-Rails to assertion pf LAN-Power-Good	40	N/A	mS
Tlpg_prst	LAN_PWR_GOOD to PERST	Timefrom LAN-Power-Good Assertion to PCIe Reset Deassertion	51	N/A	mS
Tpwr_prst	Power-Ramp to PE-Reset	Time from Power-Up of all power-Rails to Deassertion pf PERST	100		mS

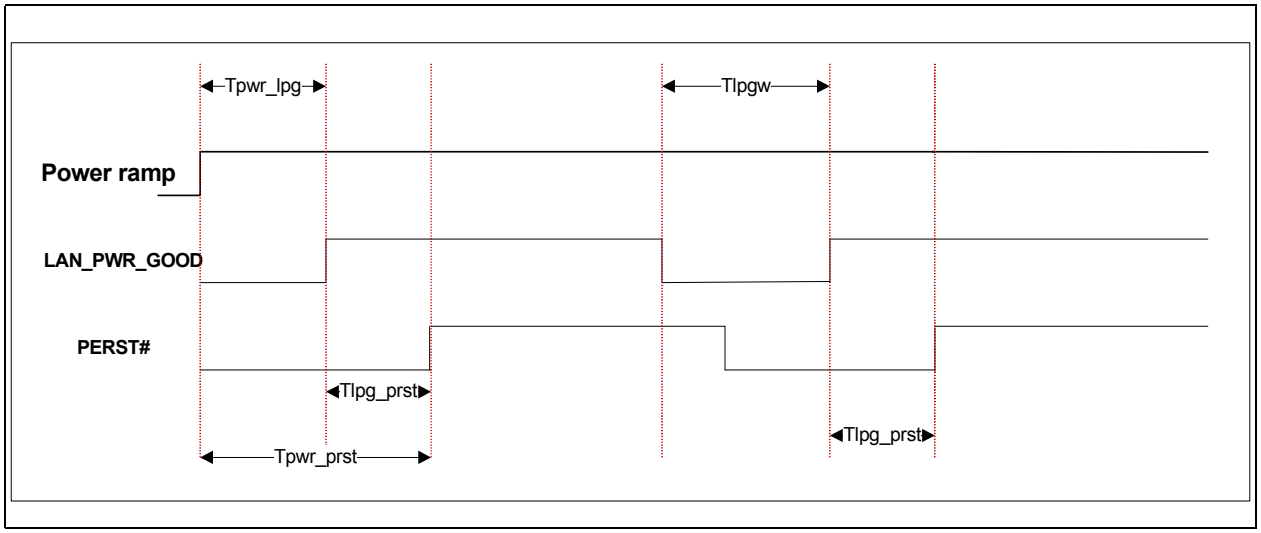


Figure 14-18. LAN-Power-Good timing

14.6.3 PCIe Interface AC/DC Specification

The XL710 PCIe interface supports the PCIe Gen3.0 electrical specification defined in

- PCIe Express Base Specification Revision 3.0
- PCI Express Card Electromechanical Specification, Revision 3.0

14.6.4 Network (MAUI) Interface AC/DC Specification

14.6.4.1 KR Interface AC/DC Specification

The XL710 MAUI interface supports the 10GBASE-KR electrical specification defined in IEEE802.3ap clause 72.

14.6.4.2 SFI+ Interface AC/DC Specification

The XL710 MAUI interface supports the SFI electrical specification defined in the SFI+ MSA (SFF Committee SFF-8431).

14.6.4.3 KX4 Interface AC/DC Specification

The XL710 MAUI interface supports the 10GBASE-KX4 electrical specification defined in IEEE802.3ap clause 71.

14.6.4.4 XAUI Interface AC/DC Specification

The XL710 MAUI interface supports the 10G XAUI electrical specification defined in IEEE802.3ae clause 47.

14.6.4.5 KX Interface AC/DC Specification

The XL710 MAUI interface supports the 1000BASE-KX electrical specification defined in IEEE802.3ap clause 70.

14.6.4.6 KR4 Interface AC/DC Specification - TBD

14.6.4.7 CR4 Interface AC/DC Specification - TBD

14.6.4.8 XLAUI Interface AC/DC Specification - TBD

14.6.4.9 XLPPI Interface AC/DC Specification - TBD

14.6.5 Crystal/Reference Clock Specification

14.6.5.1 Crystal Specification

Figure 14-19 Illustrates the usage of the crystal. This scheme is not a part of the XL710 spec but rather an example that meets the required spec.

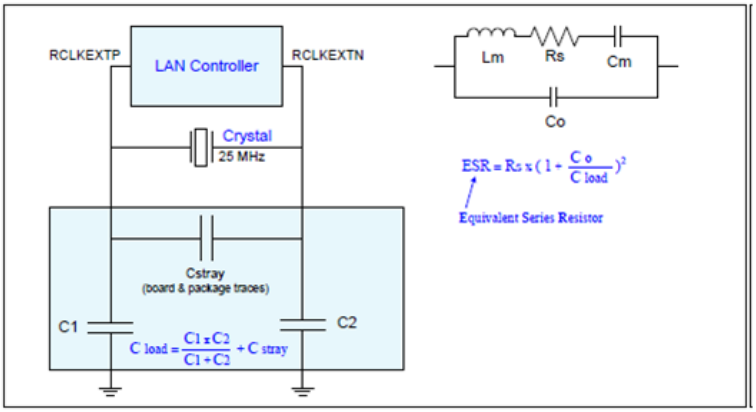


Figure 14-19. Illustrated usage of the Crystal

Table 14-22. Crystal Specifications

Parameter Name	Symbol	Recommended Value	Conditions
Frequency	f_o	25 Mhz	@25 [°C]
Vibration Mode		Fundamental	
Cut		AT	
Operating /Calibration Mode		Parallel	
Frequency Tolerance @25 °C	Df/f_o @25 °C	±30 [ppm]	@25 [°C]
Temperature Tolerance	Df/f_o	±30 [ppm]	
Operating Temperature	T_{opr}	-20 to +70 [°C]	
Non Operating Temperature Range	T_{opr}	-40 to +90 [°C]	
Equivalent Series Resistance (ESR)	ESR	50 [Ω] maximum	@25 [MHz]
Load Capacitance	C_{load}	20 [pF]	
Shunt Capacitance	C_o	6 [pF] maximum	
Pullability From Nominal Load Capacitance	Df/C_{load}	15 [ppm/pF] maximum	
Max Drive Level	D_L	0.5 [mW]	
Insulation Resistance	IR	500 [MΩ] minimum	@ 100V DC
Aging	Df/f_o	±5 [ppm/year]	
External Capacitors	C_1, C_2	20 [pF]	
Board Resistance	R_s	0.1 [Ω]	

14.6.5.2 Clock Generator Specification

Figure 14-20 shows an example for using external oscillator (PECL or CML). This scheme is not part of the XL710 spec but rather an example that meet the required spec. Yet, complete validation was made only using PE77D042-25.0M (PECL Clock Oscillator).

Note: External Oscillator must have a differential-output. The XL710 does not operate with Single-Ended Oscillator.

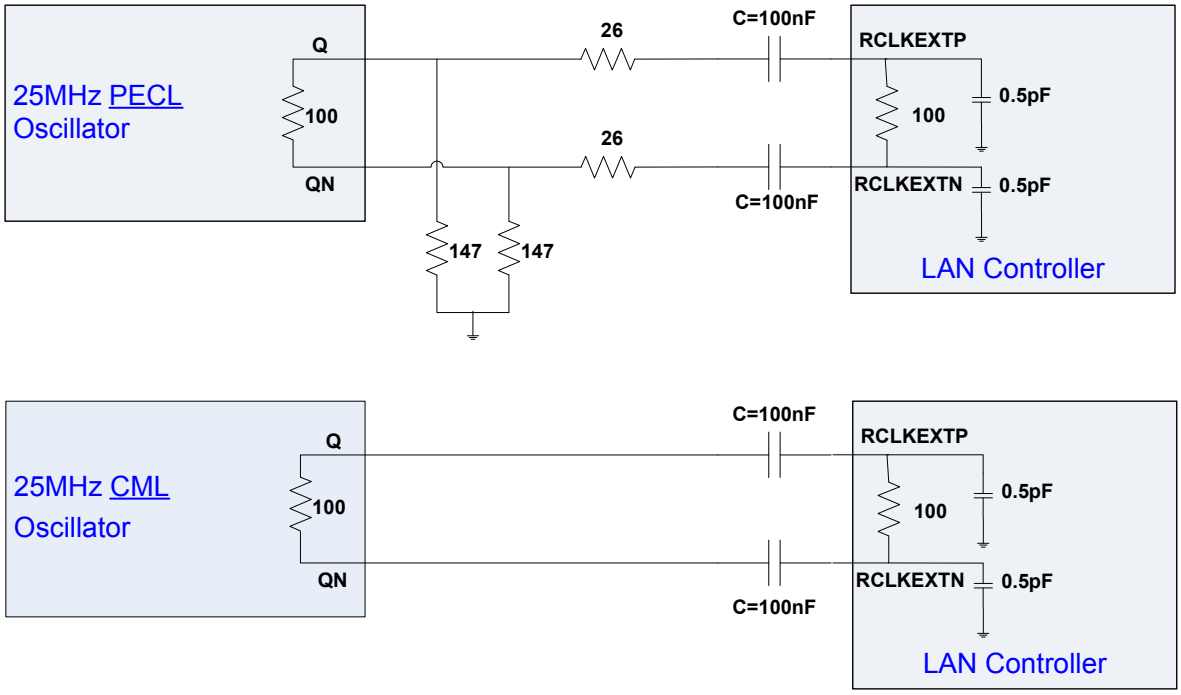


Figure 14-20. Illustrated usage of external oscillator (PECL or CML)

Table 14-23. Input Reference Clock Electrical Characteristics

Sym	Parameter	Min	Typ	Max	Unit	Comments
f	Frequency		25		MHz	
Δf	Frequency Variation	-100		+100	ppm	
DC	Duty Cycle	47.5		52.5	%	
Tr	Rise Time (20% - 80%)	300		1000	ps	
Tf	Fall Time (20% - 80%)	300		1000	ps	
AMP	Differential Pick-to-Pick Amplitude	0.6		1.25	V	
C	AC Coupling		100		nF	

Table 14-23. Input Reference Clock Electrical Characteristics

Sym	Parameter	Min	Typ	Max	Unit	Comments
Cin	Input Capacitance		0.5		pF	
DJ P2P	Deterministic Pick-to-Pick Jitter			10	ps	
p-noise	Phase Noise		-145		dBc/Hz	

14.6.6 BIAS Resistor Connection

XL710 uses a single bias circuit common to both PCIe and MAUI analog blocks.

To properly bias the high speed analog interfaces of XL710, a 4.75kΩ 0.1% resistor need to be connected between the RBIAS (ball M24) to RSENSE (ball N24). The voltage drop measurable on this bias resistor will be 950mV.

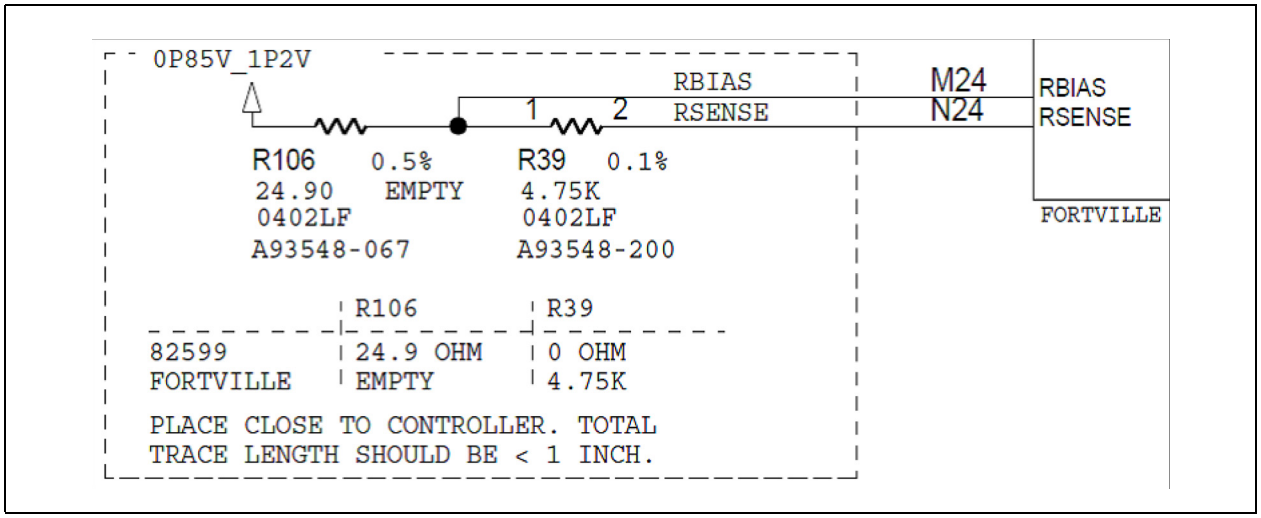


Figure 14-21. XL710 Bias Circuit

14.6.7 Thermal Diode

14.6.7.1 Connectivity

Thermal diode pins should routed with minimal serial resistivity. resistivity should be < 1 Ohm

14.6.7.2 Operation

1.Apply 1mA (maximum) current between the pins THERM_DPL and VSSA
2.Measure the voltage on THERM_DPL, the voltage is inverse proportional to the temperature.

3.Use the TD Formula to get the temperature

14.6.7.3 Thermal-Diode Formula

The temperature of the Thermal Diode can be calculate by using the following formula:

T = Vdiode * Adiode + Bdiode

when T is at [C] and Vdiode is at [V]

Parameters Adiode + Bdiode are different for each Thermal-diode (TD0, TD1) and also depend by the current level that driven to the TD during the measurement.

tables below describe the Parameters Adidode and Vdiode for TD0 / TD1 and different current level

Table 14-24. Parameters for TD0 at Conventional-Mode

Current	Adiodie	Bdiodie	Note
1 [mA]	-558.33398	418.22971	

Table 14-25. Parameters for TD1 at Conventional-Mode

Current	Adiodie	Bdiodie	Note
1 [mA]	-578.43018	432.48579	

14.7 Package

14.7.1 Mechanical

The XL710 is assembled in a 25 x 25 FCGBA package with an TBD-layer substrate.

Table 14-26. Package Specifications

Body Size	Ball Count	Ball Pitch	Ball Matrix	Substrate
25x25 mm^2	576	1 mm	24 X 24	TBD

14.7.2 Thermal

For the XL710's package thermals please refer to Section 16.0.

14.7.3 Electrical

Package electrical models are part of the IBIS files.

14.8 Mechanical Package

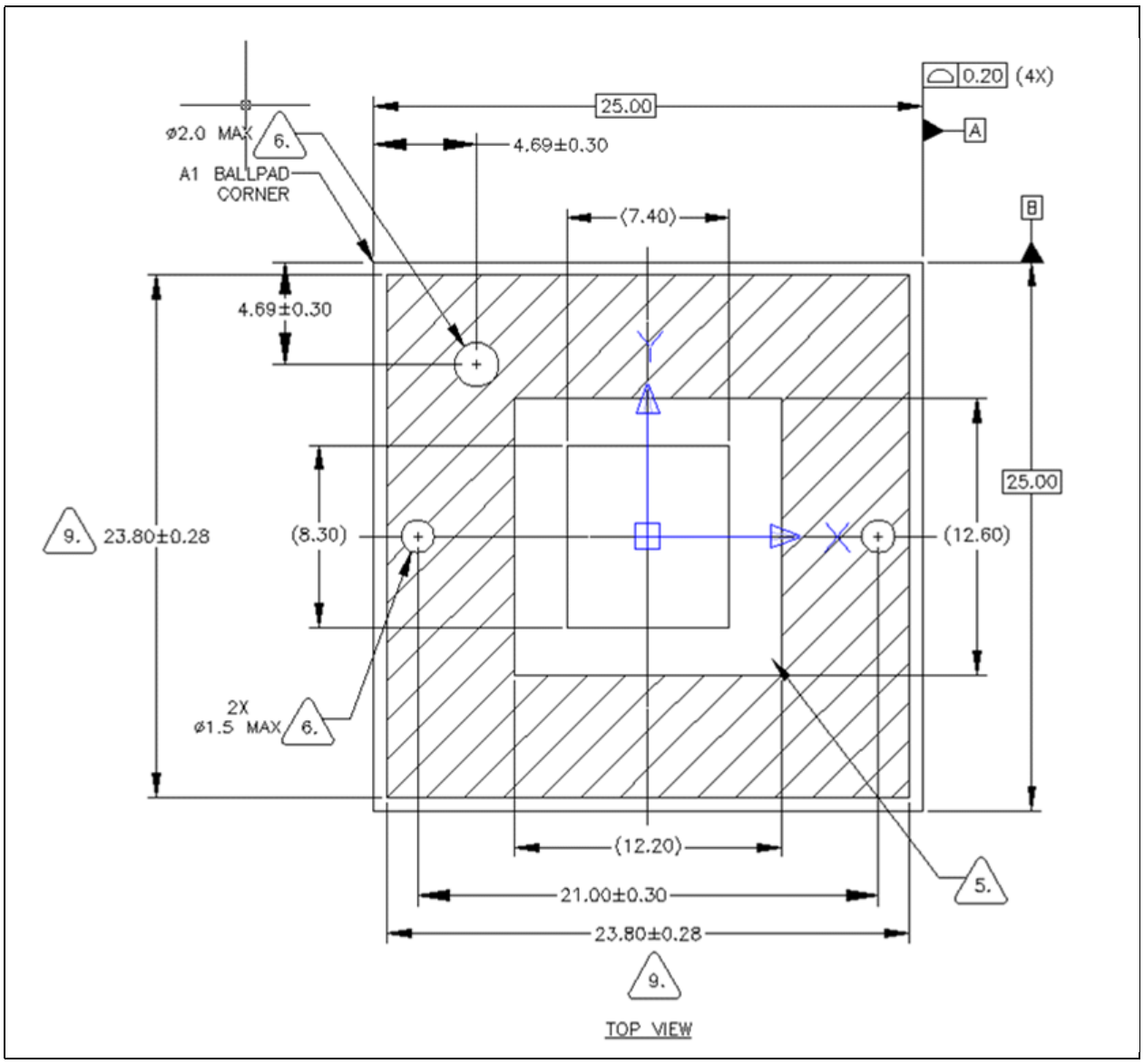


Figure 14-22. Mechanical Package Diagram - Top View

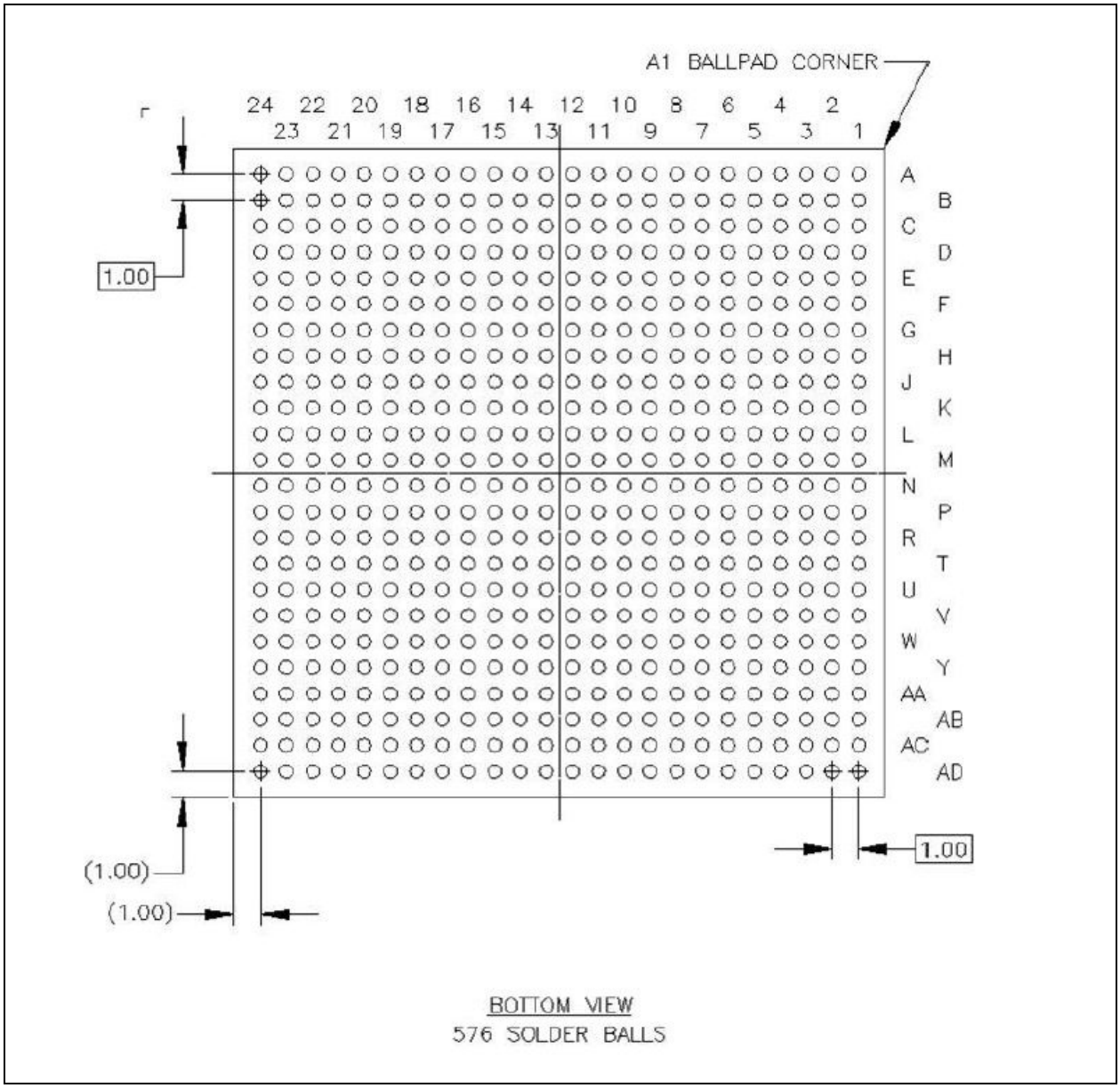
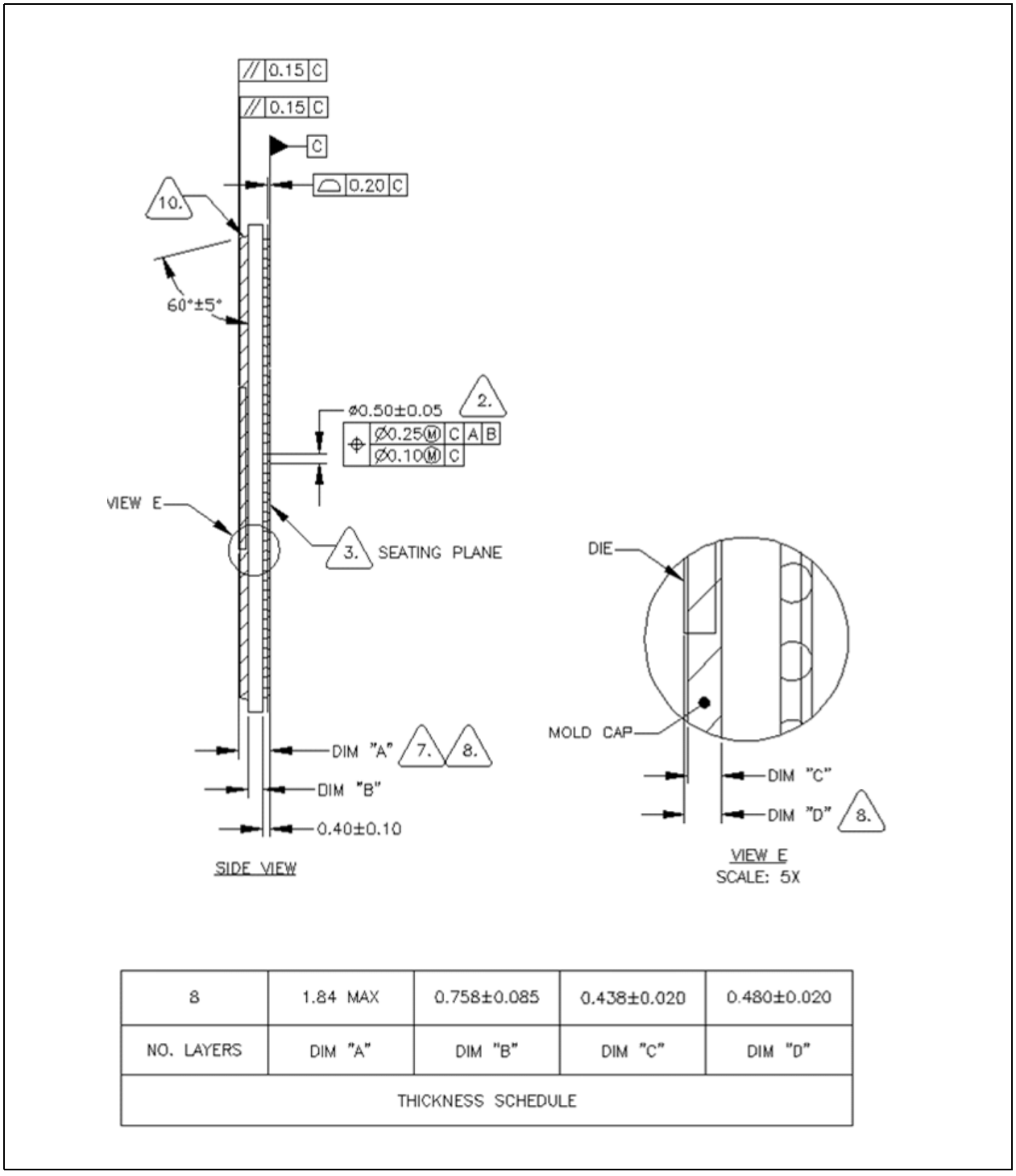


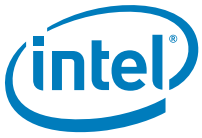
Figure 14-23. Mechanical Package Diagram- Bottom View



8	1.84 MAX	0.758±0.085	0.438±0.020	0.480±0.020
NO. LAYERS	DIM "A"	DIM "B"	DIM "C"	DIM "D"
THICKNESS SCHEDULE				

Figure 14-24. Mechanical Package Diagram- Side View

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



15.0 Design Guidelines

15.1 Connection Considerations and Guidelines

This section provides explicit recommendations for selecting components, connecting interfaces, dealing with special pins, thermal considerations and layout guidance to achieve a successful the XL710 design. The XL710 has several features and interface configuration options that provide the system designer the freedom to architect the design in a multitude of ways.

Note: Before implementing any XL710 (Intel® 40 GbE network controller chip) design based on the following guidelines, please ensure that you have the latest revision of this document. Also, it is strongly recommended that the design schematics and layout are reviewed by your Intel Field Service Representative before designs are taped out.

15.2 82599 Backward Compatibility

The 82599 and the XL710 are footprint-compatible and, for the most part pin, compatible. The pin-out of the the XL710 chip has been defined with a dual design in mind. This enables an easy transition from the 82599 controller chip to the XL710 chip without the need for a board re-design—just a Bill Of Material (BOM) change.

See the Transitioning from the Intel® 82599 to XL710: Dual Layout Design Considerations and Guidelines (CDI#466571) application note for information on creating an 82599 backward-compatible design.

15.3 Naming Conventions

General naming conventions included in this document are as follows:

- Pins named as NCxx are not connected and should be left unconnected in the design.
- Pins named as RSVDxx_NC are reserved and should be left unconnected.
- Pins named RSVDxx_0P85 are reserved and should be pulled up to the VCCD rail.
- Pins named RSVDxx_VSS are reserved and should be pulled low to GND.

In addition to the above mentioned pins needing pull-up or pull-down resistors, some interfaces, when not connected, should also be terminated with pull-up or pull-down resistors. Others must be left open. Unless otherwise specified, the recommended value of pull-up resistors ranges from 3.3 KΩ to 100 KΩ. Recommended pull-down resistor values ranges from 100 Ω to 800 Ω. Please refer to the interface descriptions provided in the sections that follow for more details.

Unless the strapping requirements are followed, the XL710 can enter special test modes and present unexpected behavior.

15.4 PCIe* Interface

The XL710 connects to a host system using the PCIe interface. The interface can be configured into different modes of operation detailed in Section 3.1.4.

When routing these differential signals, note that the channel needs to comply with signal integrity requirements of the Gen 3 interface. For details please refer to the layout recommendations in Section 15.15

15.4.1 Link Width Configuration

The XL710 supports link widths of x8, x4, or x1. The configuration is loaded from the NVM into bits 9:4 in the *Max Link Width* field of the Link Capabilities register (0xAC), the default link width being x8.

During link configuration, the upstream device and the XL710 negotiate a common link width. In order for this to work, the selected maximum number of PCIe lanes must be physically connected to the host system.

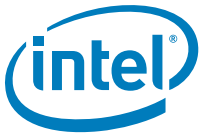
15.4.2 Polarity Inversion and Lane Reversal

To ease routing, designers have the flexibility to use the lane reversal modes supported by the XL710. Polarity inversion can also be used since the polarity of each differential pair is detected during the link training sequence.

Note: When lane reversal is used, some of the down-shift options are not available. For a description of available combinations, see Section 3.1.4.3.

15.4.3 AC Coupling

Each PCIe lane has to be AC-coupled between its corresponding transmitter and receiver; with the AC-coupling capacitor located close to the transmitter side (within 1 inch). Please note that the recommended value of the AC coupling capacitors depends on the link speed. For designs that support PCIe speeds up to Gen2 (5GT/s), a value of 100 nF is recommended. For designs that include support for Gen3 (8GT/s) speeds, a value of 220 nF is recommended As stated in the PCIe Base specification 3.0 for 5.0GT/s speeds or lower the capacitance value should be between 75 nF-265 nF. For 8GT/s speeds, the capacitance value should be between 176 nF-265 nF.



15.4.4 PCIe Terminations

Each PCIe differential pair is terminated on-die; board termination is not required.

15.4.5 Miscellaneous PCIe Signals

The XL710 signals power management events to the system by pulling the PE_WAKE_N signal (pin AA18) signal low. This signal operates like the PCI PME# signal. Note that somewhere in the system, this signal must be pulled high to the auxiliary 3.3V supply rail.

The PE_RST_N signal (pin AD18), which serves as the familiar reset function for the controller chip, needs to be connected to the host system's corresponding signal.

The PCIe interface also requires a reference clock that is usually provided by the host system. For more details, refer to Section 15.8.

15.5 MAUI Interfaces

This section outlines the LAN interface configuration options and lane mapping guidelines for the XL710. Choose the appropriate configuration for your system design.

The XL710 offers a maximum of four 10 Gb/s serial ports, a maximum of two 10 Gb/s parallel ports (for backward compatibility with the 82599), or it can maintain up to two 40 GbE links with an aggregated bandwidth of 40 Gb/s. Consult product marketing for 40 GbE failover support and availability.

The interface categories supported in each of these three cases are slightly different:

- In quad-port 10 GbE mode, supported interface options are SGMII (1 GbE), KX (1 GbE), KR, or SFI.
- In dual-port 10 GbE mode, supported interface options include both 10 GbE serial as well as parallel: SGMII (1 GbE), XAUI, KX (1 GbE), KX4, KR or SFI.
- In 40 GbE mode, the supported interface option is one port of: KX, KR, KR4, CR4, XLAUI or XLPPI.

The XL710's PHY has eight SerDes blocks, all of which support 10 GbE serial (KR/SFI) operation. They are grouped in two groups; Group A and Group B. See Figure 15-1.

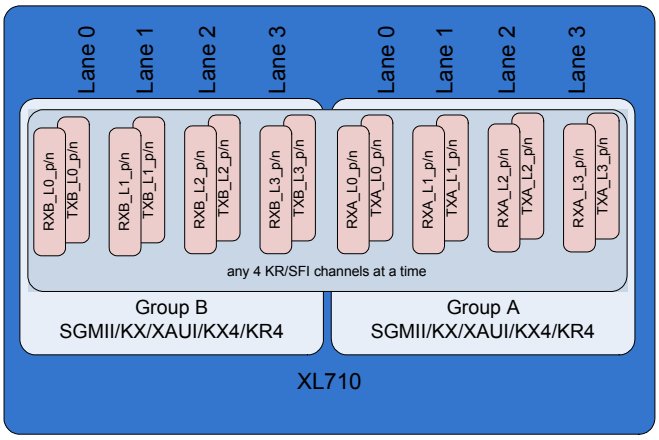


Figure 15-1. Lane Mapping

Figure 15-1 shows the lanes according to their physical location on the package when observed from above (die side).

15.5.1 Lane Mapping

The XL710's ports can be assigned in many different lanes/lane combinations to enable implementation of backward-compatible designs or highly configurable designs.

In Figure 15-1, the eight available SerDes Tx/Rx lane pairs are grouped into two groups: Group A and Group B. For a complete list of these pin-pairs, see Section 2.2.2.

Table 3-62 lists the possible port-to-lane physical lane mapping options for various PHY interfaces.

Ethernet MAC ports are numbered from 0 through 3. When Ethernet Port 0 and Port 1 are configured to operate in 10 GbE parallel mode, ports 2 and 3 are disabled. Similarly when Ethernet Ports 0 and 1 are configured to 40 GbE mode ports 2 and 3 are disabled.

All SerDes differential lanes are 100 Ω terminated on die. Differential MAUI signals need to be AC coupled near the receiver. For recommended capacitor values, consult the relevant IEEE 802.3 specifications and/or the relevant PICMG specifications. Capacitor size should be small to reduce parasitic inductance. Use X5R or X7R, ±10% capacitors in a 0402 or 0201 package size.

Note: SFP+ and QSFP+ board traces generally do not require AC coupling capacitors because they are normally integrated into the pluggable SFP+ and QSFP+ modules.

15.6 Sideband Signals

This section focuses on the various sideband interfaces provided to operate with the externally connected SFP+ and QSP+ modules or PHY chips. Several of these sideband interfaces behave differently based on the use case.As such, the following section focuses on usage models like: SFP+, QSFP, and 10GBASE-T or other solutions using an external PHY.

15.6.1 Management Data Input/Output (MDIO) Interfaces

The XL710 supports up to four MDIO interfaces (one per port) to control external PHY devices. The first two MDIO ports, MDIO0 and MDIO1, are at pin locations: MDC0_SCL0, MDIO0_SDA0 and MDC1_SCL1, MDIO1_SDA1 (pins AC12, AD12 and AB18, AC17). MDIO ports 2 and 3 are located at pins MDC2_SCL2, MDIO2_SDA2 and MDC3_SCL3, MDIO3_SDA3 pins are located at balls AB12, AA12 and Y16, AC18. All four MDIO ports can also be configured to operate in I^2C mode as well. The operating mode of these interfaces is NVM configurable.

In order to manage multi-port PHYs, configure the MDIO interface of Port0 to control a Quad port PHY, or the MDIO interfaces of Port 0 and Port 1 to control dual port PHYs. The PHY configuration registers for each port are mapped into the respective MDIO address space and are accessible to the MAC or the MDIO controller firmware.

Table 15-1. XL710 MDIO Port Mapping

Port 0	Port 1	Port 2	Port 3
MDC0_SCL0 MDIO0_SDA0	MDC1_SCL1 MDIO1_SDA	MDC2_SCL2 MDIO2_SDA2	MDC3_SCL3 MDIO3_SDA3

The XL710's MDIO interfaces use 3.3 V LVTTL signaling. Therefore, interfacing external PHY devices using 1.2 V signaling requires level translators.

When connecting the MDIO interfaces, make sure the appropriate MDIO and MDC signals are connected to the corresponding pins on the PHY chip(s) connected to each MAUI port. Depending on the PHY MDIO interface signaling levels supported by the attached PHY, use appropriate level translators if needed. Also make sure to provide a pull-up resistor to the appropriate voltage(s) on the MDIO signal.

15.6.2 I^2C Interfaces

Since the XL710 offers the option to implement a quad port SFP+ design, a total of four I^2C interfaces are needed. To satisfy this requirement, the MDIO interfaces previously described have the flexibility to operate in I^2C mode.

Table 15-2 lists the mapping of the I^2C interfaces to the SFP+ ports:

Table 15-2. XL710 I²C Interface to SFP+ Port Mapping

Port 0	Port 1	Port 2	Port 3
MDC0_SCL0, MDIO0_SDA0	MDC1_SCL1, MDIO1_SDA1	MDC2_SCL2, MDIO2_SDA2	MDC3_SCL3, MDIO3_SDA3

When implementing SFP+ interfaces, the I^2C interfaces need to be connected to the I^2C pins of the SFP+ connector. Please make sure to provide pull-up resistors to 3.3V on both SCL and SDA pins.

When implementing a dual QSFP+ interface, only Port 0 I^2C and Port 1 I^2C need to be connected to the QSFP+ connector I^2C pins. Make sure to provide pull-up resistors to 3.3V on both SCL and SDA pins in all used or unused ports.

For more details, consult the XL710's reference schematics.

15.7 General Purpose I/O (GPIO) Pins

The XL710 has a total of 30 GPIO pins that can be configured as Software Definable Pins (SDPs), LED drivers, and dedicated hardware functions for connecting to external PHYs or IEEE 1588 auxiliary devices. The GPIO pins can also be associated with any of the physical ports. The following sections show the default configuration for the GPIO pins reserved for specific use and are named by default as SDP, LED or GPIO signals. However, the XL710 offers the flexibility to configure any of the GPIO pins, regardless of name, to different modes and associate them with different ports as described in Section 3.4.3. It is not recommended that one deviate from the default GPIO function allocation specified in this document in order to reduce the amount of support and extra validation needed.

15.7.1 Software-Definable Pins (SDPs)

By default, the XL710 has a total of 16 SDPs. These pins can either operate in native mode or as a software definable pin. In native mode, they serve certain predefined functions and are used as sideband signals for interfacing external PHYs and SFP+ or other modules.

In 82599 compatibility mode (dual port configuration), the XL710 can support the same SDP functions as the 82599 (Table 3-32). In single port mode, all 16 SDPs can be configured for use as Port 0 and in quad port mode, it has to split the total of 16 software definable pins between the four ports. In non-82599 compatibility mode, although the pin locations do not change, their default allocation to the different MAUI ports do. Pins SDP0_0-SDP0_3 (ball locations AD8, AC8, AB8, and AA8) remain allocated to Port 0. Pins SDP0_4-SDP0_7 are redefined as SDP2_0-SDP2_3 (ball locations AD7, AC7, AB7, and AA7) and allocated to Port 2. Pins SDP1_0-SDP1_3 (ball locations AC16, AB16, AB17, and AA17) remain allocated to Port 1. Pins SDP1_4-SDP1_7 are redefined as SDP3_0-SDP3_3 (ball locations AA16, AC15, AB15, and AA15) and allocated to Port 3. The native functions supported by these SDPs also change. Table 15-3 lists the native function of these SDPs in various scenarios.

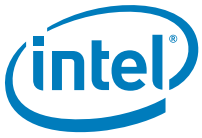


Table 15-3. XL710 Native SDP Functions

Port#	Pin Name	SFP+	QSFP	10GBASE-T Copper PHY
0	SDP0_0	RX_LOS	INTL	PHY_INT
	SDP0_1	TX_FAULT	RESETL	PHY_RST
	SDP0_2	MOD_ABS	MOD_PRESL	TX_DISABLE
	SDP0_3	RS0/RS1	LPMODE	
	SDP2_0		MODSELL	
	MDIO0_SDA0 MDC0_SCL0	SDA SCL	SDA SCL	MDIO MDC
1	SDP1_0	RX_LOS	INTL	PHY_INT
	SDP1_1	TX_FAULT	RESETL	PHY_RST
	SDP1_2	MOD_ABS	MOD_PRESL	TX_DISABLE
	SDP1_3	RS0/RS1	LPMODE	
	SDP2_1		MODSELL	
	MDIO1_SDA1 MDC1_SCL1	SDA SCL	SDA SCL	MDIO MDC
2	SDP2_0	RX_LOS		PHY_INT
	SDP2_1	TX_FAULT		PHY_RST
	SDP2_2	MOD_ABS		TX_DISABLE
	SDP2_3	RS0/RS1		
	MDIO2_SDA2 MDC2_SCL2	SDA SCL	SDA SCL	MDIO MDC
3	SDP3_0	RX_LOS		PHY_INT
	SDP3_1	TX_FAULT		PHY_RST
	SDP3_2	MOD_ABS		TX_DISABLE
	SDP3_3	RS0/RS1		
	MDIO3_SDA3 MDC3_SCL3	SDA SCL	SDA SCL	MDIO MDC

Note: Table entries listed in gray are not SDPs but functionally belong to the sideband signals assigned to the same port.

Note that the TX_DISABLE signal for the SFP+ implementation has been removed from the native SDP functions. This functionality is implemented through I^2C register access. To support modules that don't support this feature, use one or more of the available GPIO pins. When connecting the SDPs to different digital signals, note that these are 3.3V signals and implement level shifting, if necessary.

The XL710 enables all of these SDP pins to be configured as a general purpose interrupt pin. Interrupts can be generated on both the rising and falling edge of the signal. The rising or falling edge detection is implemented by comparing values sampled at the internal clock rate, as opposed to an edge-detection circuit.

15.7.2 Light Emitting Diodes (LEDs)

Each of the LED outputs can be individually configured to select the particular event, state, or activity indicated on that output. In addition, each LED can be individually configured for output polarity as well as for blinking versus non-blinking mode when asserted. For more information on the available states indicated or how to configure these pins from the EEPROM settings, see Section 2.2.6.1.

By default, the XL710 divides the LED pins and uses the same pin locations for LEDs. It divides them up differently based on the total number of ports supported by the implementation. For example, consider the following default configurations:

1. With a dual port implementation, the same configuration is possible as that of the 82599.With a four port implementation, a total of eight available pins are split to four groups and assigned two LEDs each per port. The new names of these pins and their respective locations follow: LED0_0, LED0_1, LED2_0, LED2_1, LED1_0, LED1_1, LED3_0, and LED3_1. They are located at the following pin locations: AD14, AC14, AD13, AC13, AB14, AA14, AB13, and AA13.

Note that the reduced number of LED pins in a four-port configuration presents some design limitations but the configuration flexibility of the GPIO pins allows for intuitive LED implementation.

To prevent damage to the interface and the LEDs themselves, provide separate current limiting resistors for each LED connected. Consider attaching filter capacitors to these outputs if the LEDs are placed close to the board edge and near noisy external interconnects that can cause EMI issues. Adequate separation in routing from noisy signals helps prevent such issues.

15.7.3 GPIO Pins

Six individually controllable GPIOs (ball locations C19, B19, B18, A18, Y12, and Y14) are not assigned for a specific use or port. Uses for these pins include an LED interface or SDP, IEEE1588 connections for auxiliary devices, low speed optical module interfaces, external PHY control, etc. They can also be configured for use as external interrupt sources. Configure the mode and port associations for these pins as described in Section 2.0.

15.7.4 MDIO/I^{2}C/SDP/LED/GPIO Summary

Table 15-4 lists a summary of the various sideband interfaces and software definable pins (GPIOs configurable as LEDs, SDPs etc.). In the descriptions column of the quad-port implementations, each signal name is preceded by a prefix designating the port to which the signal in question belongs (such as P0-SCL). To ease the readability of the table, signal names related to each port are color-coded as follows: Port 0, Port 1, Port 2, and Port 3.

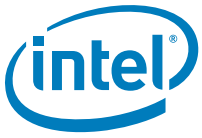


Table 15-4. XL710 MDIO/I²C/SDP/LED Summary

		Description (XL710's pin functions in several use cases)			
Pin Name (default function)	**Ball**	**10 GbE Parallel**	**SFP+**	**QSFP**	**10GBASE-T**
MDC0_SCL0	AC12	MDC0	P0-SCL	P0-SCL	MDC
MDIO0_SDA0	AD12	MDIO0	P0-SDA	P0-SDA	MDIO
MDC1_SCL1	AB18	MDC1	P1-SCL	P1-SCL	MDC
MDIO1_SDA1	AC17	MDIO1	P1-SDA	P1-SDA	MDIO
MDC2_SCL2	AB12		P2-SCL		MDC
MDIO2_SDA2	AA12		P2-SDA		MDIO
MDC3_SCL3	Y16		P3-SCL		MDC
MDIO3_SDA3	AC18		P3-SDA		MDIO
SDP0_0	AD8		P0-RX_Loss	P0-IntL	PHY_Int
SDP0_1	AC8		P0-TX_Fault	P0-ModPrsntL	PHY_Rst
SDP0_2	AB8		P0-Mod_Abs	P0-ResetL	TX_Disable
SDP0_3	AA8		P0-RS0/RS1	P0-LPMode	
SDP2_0	AD7		P2-RX_Loss	P0-ModselL	PHY_Int
SDP2_1	AC7		P2-TX_Fault	P1-ModselL	PHY_Rst
SDP2_2	AB7		P2-Mod_Abs		TX_Disable
SDP2_3	AA7		P2-RS0/RS1		
SDP1_0	AC16		P1-RX_Loss	P1-IntL	PHY_Int
SDP1_1	AB16		P1-TX_Fault	P1-ModPrsntL	PHY_Rst
SDP1_2	AB17		P1-Mod_Abs	P1-ResetL	TX_Disable
SDP1_3	AA17		P1-RS0/RS1	P1-LPMode	
SDP3_0	AA16		P3-RX_Loss		PHY_Int
SDP3_1	AC15		P3-TX_Fault		PHY_Rst
SDP3_2	AB15		P3-Mod_Abs		TX_Disable
SDP3_3	AA15		P3-RS0/RS1		
LED0_0	AD14	LED0_0	LED0_0	LED0_0	LED0_0
LED0_1	AC14	LED0_1	LED0_1	LED0_1	LED0_1
LED2_0	AB14	LED0_2	LED2_0	LED0_2	LED2_0
LED2_1	AA14	LED0_3	LED2_1	LED0_3	LED2_1
LED1_0	AD13	LED1_0	LED1_0	LED1_0	LED1_0
LED1_1	AC13	LED1_1	LED1_1	LED1_1	LED1_1
LED3_0	AB13	LED1_2	LED3_0	LED1_2	LED3_0
LED3_1	AA13	LED1_3	LED3_1	LED1_3	LED3_1
GPIO0	E18				
GPIO1	E16				
NC	C19				
NC	B19				
GPIO2	B18				
GPIO3	A18				

		Description (XL710's pin functions in several use cases)			
Pin Name (default function)	Ball	10 GbE Parallel	SFP+	QSFP	10GBASE-T
GPIO4	Y12				
GPIO5	Y14				

15.8 Reference Clocks

15.8.1 PCIe Reference Clock

The XL710 requires a 100 MHz differential reference clock for the PCIe block. This signal is typically generated on the host system and routed to the PCIe port. For add-in cards, the clock is furnished at the PCIe connector.

This clock must have a frequency tolerance of ±300 ppm and it must be connected to the PE_CLK_p and PE_CLK_n pins of the XL710. The clock pins are AB23 and AB24 respectively. Designers have the flexibility to invert the polarity of this signal if that results in a cleaner routing of the board.

Note: Reference clock specifications are detailed in both the base and CEM specification. Consult both specifications to understand the full set of requirements. Refer to sections 4.3.7 (Base Specification) and 2.1.3 (CEM Specification) in particular.

15.8.2 MAUI Reference Clock

The XL710 requires a reference clock that is being used to generate the high frequency clocks for the MAC and PHY blocks of the controller.

There are many oscillator solutions available. The clock source and distribution network should be designed to meet the necessary frequency and jitter requirements.

To configure the XL710's PLLs correctly, the oscillator select pin OSC_SEL (pin AA9) needs to be pulled low (for example, a 100 Ω resistor).

There are different oscillator solutions with their pros and cons as described in the sections that follow:

15.8.2.1 Fixed Crystal Oscillator

A packaged fixed crystal oscillator includes an inverter, a quartz crystal, and passive components. The device renders a consistent square wave output. Oscillators used with microprocessors are supplied in many configurations and tolerances. Crystal oscillators can be used in special situations (such as when shared clocking among devices). As clock routing can be difficult to accomplish, it is preferable to provide a separate clock source for each device by using clock buffers

15.8.2.2 Programmable Crystal Oscillators

A programmable oscillator can be configured to operate at many frequencies. This device contains a crystal frequency reference and a Phase Lock Loop (PLL) clock generator. The frequency multipliers and divisors are controlled by programmable fuses.

PLLs are prone to exhibit frequency jitter. This contributes to the jitter of the transmitted signal even with the programmable oscillator working within its specified frequency tolerance. If the transmitted signal has high jitter and the receiver PLL's jitter tolerance is marginal, it can lose lock causing bit errors or link loss. Use of programmable oscillators is not generally recommended.

15.8.2.3 Reference Clock Measurement Recommendations

A low capacitance, high impedance probe (C < 1 pF, R > 500 K) should be used for testing. The measurement should be done on the XTAL_OUT (P1) pin. Incorrect probing setup or choice of probe affects the measurement and can lead to inaccurate results (from change in amplitude through clock frequency shift to even preventing the oscillations from starting).

15.9 Bias Resistors

The XL710 uses a single bias circuit common to both PCIe and MAUI analog blocks. To properly bias the high speed analog interfaces, a 4.75 KΩ 0.1% resistor needs to be connected between the RBIAS (ball M24) to RSENSE (ball N24). The voltage drop measurable on this bias resistor is 950 mV.

To avoid noise coupled onto this reference signal, place the bias resistor close to the controller chip and keep traces as short as possible.

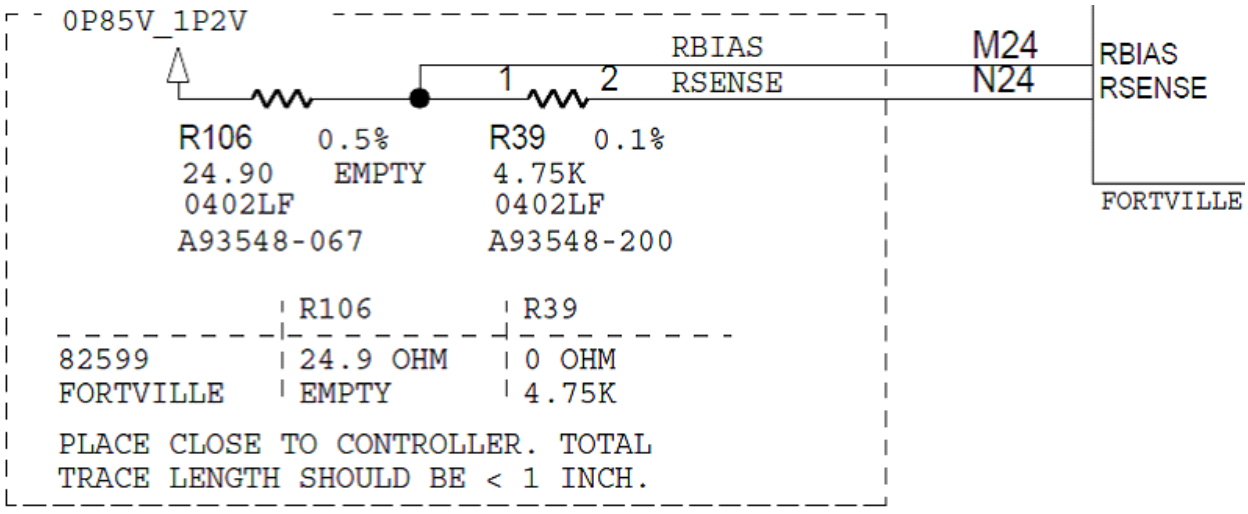


Figure 15-2. XL710 Bias Circuit (82599 Backward Compatible Connection)

15.10 NVM Devices

The XL710 stores its configuration data in externally attached non-volatile memory chips, referred to as NVMs. Optional boot code packages are also stored in these externally attached non-volatile memory chips. These Flash devices are connected through the different Serial Peripheral Interface (SPI) as described in the section that follows.

15.10.1 SPI Flash

The SPI interface is used to connect to Flash devices. Flash devices are used for storing both configuration data and firmware as well as for storing the optional boot code packages.

The pins used to connect the Flash devices are FLASH_CE_N, FLASH_SCK, FLASH_SI and FLASH_SO (balls B7, A8, B6, and A7). Please note that the FLASH_SI pin is an output pin and needs to be connected to the serial input of a Flash device and the FLASH_SO is an input pin and needs to be connected to the serial output of a Flash device. For more information and the electrical characteristics of these pins see Section 14.6.2. Due to the Flash size requirements detailed in the sections that follow designers must make sure that both size requirements can be met by devices with the chosen footprint.

15.10.2 Supported Flash Devices

Depending on the expansion ROM requirements, the XL710 designs require an SPI Flash device ranging from 32 to 64 Mb in size with an address size of 24 bits.

Table 15-5. Proposed Flash Devices

Density	Atmel* PN	Winbond*	SST*	Numonyx*
32 Mb	AT25DF321A	W25Q32BV	SST25VF032B	M25PX32
64 Mb	AT25DF641	W25Q64CV	SST25VF064C	M25PX64

The previous list of proposed Flash devices for the XL710 can change without notice. Contact your Intel representative for more details.

For more information on how to properly attach a Flash device, follow the example provided in the XL710 reference schematics.

15.11 Manageability Interfaces

The XL710 supports the SMBus, NC-SI interfaces, and PCIe (see Section 3.1) for pass-through and configuration traffic between the Management Controller (MC) and the XL710. See Section 10.0, for more information about MC support.

15.11.1 SMBus

The SMBus interface is formed by the SMBCLK, SMBD and SMBALRT_N signals. Since the SMBus specification does not specify the fastest rise time of these clock and data signals, in order to reduce potential PCIe signal integrity issues, the pins dedicated for this interface have been located at E8, E10, and E14.

Optionally, in order to further reduce high frequency content that could potentially couple as noise to other adjacent signals in the package, series resistors (of up to 1.1 KΩ) should be used on the SMBus signals.

If the interface is not being used, the design should provide a pull-up resistor on all of these signals.

Note: Intel recommends connecting the device via the SMBus for NVM recovery. An MC can send the NVM release command to recover a corrupted NVM.

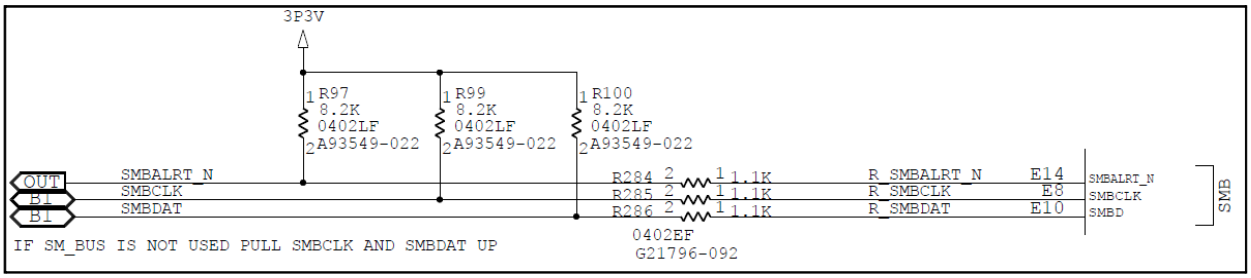


Figure 15-3. SMBUS Connections

15.11.2 NC-SI

The NC-SI provides a 100 Mb/s manageability interface. Management packets can be routed to or from an external management processor. The MC is required to meet the requirements called out in the DMTF NC-SI specification.

The XL710 supports hardware arbitration therefore, two additional pins from the single point to point NC-SI solution, have been defined: NCSI_ARB_IN (pin Y8) and NCSI_ARB_OUT (pin Y10). These pins can be used to implement a hardware arbitration ring. The NCSI_ARB_IN pin of one controller has to be connected to the NCSI_ARB_OUT of another controller to form a ring configuration. A maximum of four network controllers can be connected in this manner and all controllers that share the same NC-SI interface pins must support this feature in order to use hardware-based arbitration

The following sections present examples of single- and multi-drop configurations, with pull-up and pull-down requirements for each.

15.11.2.1 Single-Drop Configuration

The simplest NC-SI configuration is a single drop configuration. This refers to a point-to-point connection between one MC and one network controller device. The following schematic shows an example. Since the hardware arbitration is not applicable for this scenario, the NCSI_ARB_IN and

NCSI_ARB_OUT signals are not used. To enable correct operation, the NCSI_ARB_OUT should be looped back to the NCSI_ARB_IN pin of the network controller or hardware arbitration should be disabled.

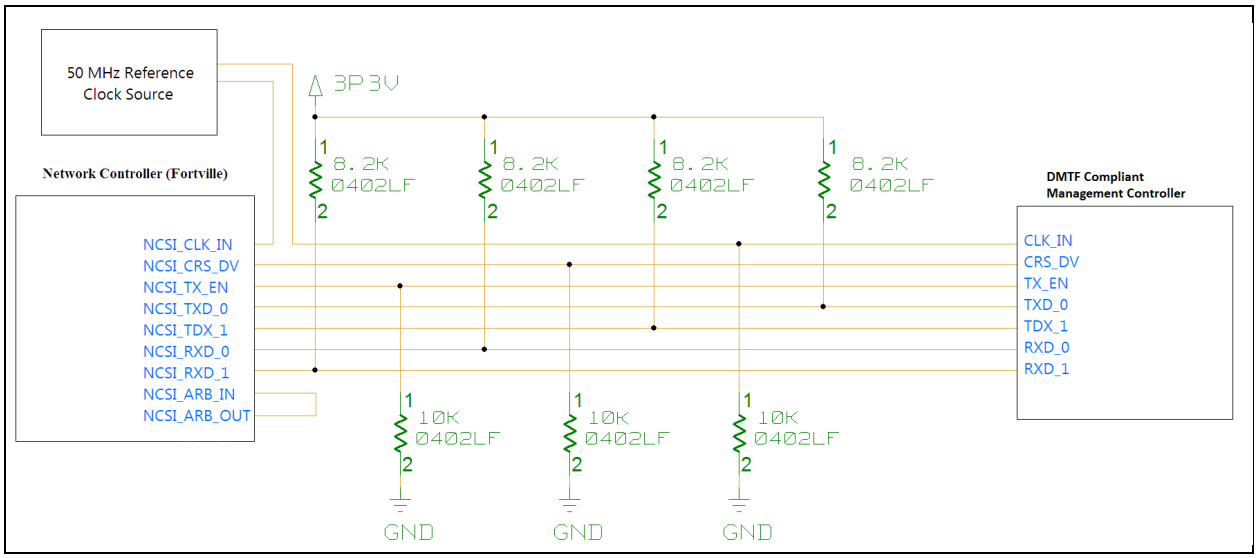


Figure 15-4. NC-SI Connection Schematic: Single-Drop Configuration

15.11.2.2 Multi-Drop Configuration

NC-SI architecture also enables multiple endpoints to be connected to the same management controller. In this configuration, the bus arbitration can also be implemented by hardware using a token ring configuration. The NCSI_ARB_IN pin of one controller must be connected to the NCSI_ARB_OUT of another controller to form a ring configuration.

A maximum of four network controllers can be connected in this manner and all controllers sharing the same NC-SI interface pins must support this feature in order to use hardware-based arbitration. See Section 2.2.3 for multi-drop schematic examples.

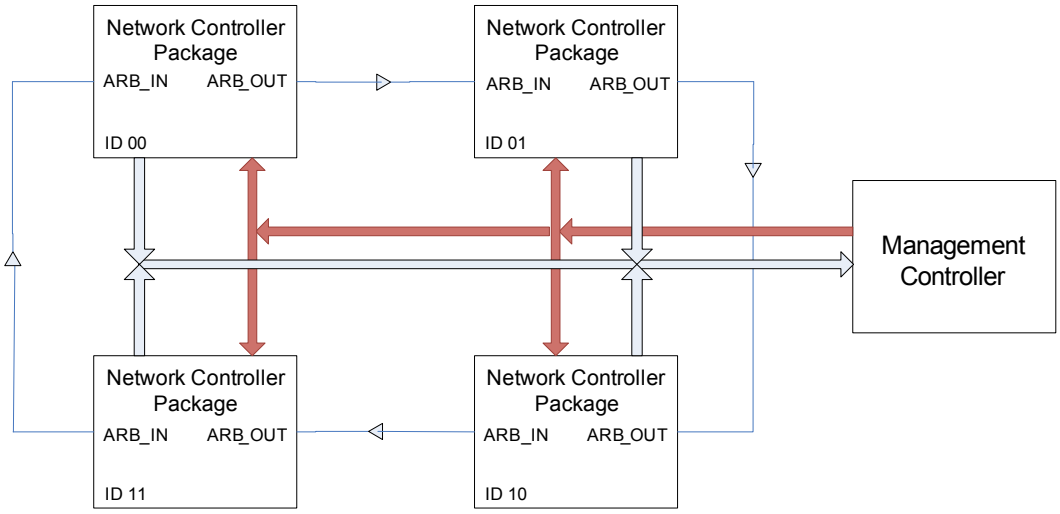


Figure 15-5. Multi-Drop Configuration Example

If the NC-SI interface is not used, pull the NCSI_CLK_IN, NCSI_CRS_DV, and NCSI_TX_EN signals low and the NCSI_RXD_0, NCSI_RXD_1, NCSI_TXD_0, and NCSI_TXD_1 signals high.

15.12 Miscellaneous Signals

15.12.1 Disable Signals

Two signals can be used for disabling Ethernet functions from system BIOS. Use pins PCI_DIS_N (pin AD20) and DEV_DIS_N (AD21). Both are latched at the rising edge of LAN_PWR_GOOD, PE_RST_N or In-Band PCIe Reset.

- **PCI_DIS_N**: If this pin is not connected or is driven high during initialization, then all PCI functions configured from the NVM are enabled. If this pin is driven low during initialization then all PCI functions that are allowed to be disabled as configured in NVM are disabled. When all bits in the corresponding NVM configuration word are set, all PCI functions are disabled when this pin is asserted low.
- **DEV_DIS_N**: If this pin is not connected or is driven high during initialization, all LAN ports configured from NVM are enabled. If this pin is driven low during initialization all LAN ports allowed to be disabled as configured in NVM are disabled. When all bits in the corresponding NVM configuration word are set, all LAN ports are disabled when this pin is asserted low. When a LAN port is disabled, the associated PCI functions are disabled as well.

The PCI_DIS_N and DEV_DIS_N pins should maintain their state during system reset and system sleep states as well as ensure the proper default value on system power-up. For example, a designer could use pin that defaults to 1b (enable) and is on system suspend power, meaning that it maintains it's state in S0-S5 ACPI states.

15.12.2 Power-on Reset

After power is applied, the XL710 must reset. There are two ways to do this:

- Using the internal power on reset circuit (recommended).
- Using the external LAN_PWR_GOOD signal.

By default, the internal power on reset should be used. In this case, the power supply sequencing timing requirement between the 3.3V and VCCD (for the XL710) power rails must be met. If this requirement is impossible to meet, the alternative is to bypass the internal power-on reset circuit by pulling POR_BYPASS (pin D19) high and using an external power monitoring solution to provide a LAN_PWR_GOOD signal (pin A14).

Table 15-6. Reset Context for POR_BYPASS and LAN_PWR_GOOD

POR_BYPASS	Active Reset Circuit	
If = 0b	Internal POR	
If = 1b	LAN_PWR_GOOD	External Reset
	If = 0b	Held in reset.
	If = 1b	Initialized, ready for normal operation.

It is important to ensure that resets for the MC and the XL710 are generated within a specific time interval. The important requirement here is to ensure that the NC-SI link is established within two seconds of the MC receiving the power-good signal from the platform. Both the XL710 and the external MC need to receive power-good signals from the platform within one second of each other.

The MC should poll this interface and establish a link for two seconds to ensure specification compliance.

15.12.3 Connecting the JTAG Port

The XL710 offers a test access port conforming to the IEEE 1149.1-2001 Edition (JTAG) specification. The JTAG interface operates at 3.3V only. To use the test access port, connect the JTCK, JTMS, JTDI, JTDO and JRST pins (balls B16, B13, A13, B15, and B14) to test points accessible by appropriate test equipment.

For proper operation, a pull-down resistor with a value in the range of 470 Ω to 8.2 KΩ range should be connected to the JTCK and JRST_N signals and pull-up resistors with values in the KΩ range to the JTDO, JTMS and JTDI signals.

A Boundary Scan Definition Language (BSDL) file describing the XL710 is currently available for use in your test environment.

All designs supporting the XL710 must provide stuffing options to individually strap pins E15, F21, and Y9 high or low to configure debug capabilities. All designs must also provide accessible test point connections to the JTAG interface to enable system debug.

Note: Intel strongly recommends that each XL710 design provides a header for the JTAG interface to enable quick system debug. If the XL710 is part of a JTAG chain the design should also provide stuffing options to isolate it from the chain. Figure 15-6 shows the required connector

(Samtec TMM-103-01-L-D-SM or equivalent) and pin-out:

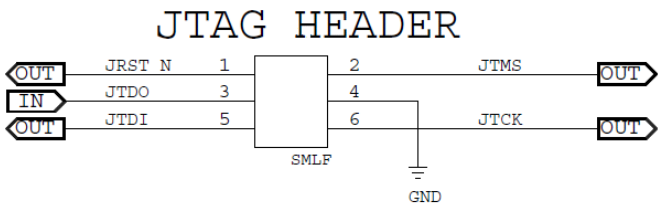


Figure 15-6. JTAG Header

15.13 Power Supplies

The XL710 requires the following power rails: 3.3V for the I/O ring, VCCA for the analog core and VCCD for the digital core. To provide a tight control over voltage levels and improve signal integrity, the VCCA rail is generated by a small on-die switching voltage regulator and is in the 0.9V - 1V range.

For powering the VCCD rail, designers need to provide an external switching voltage regulator. The voltage level needs to be adjusted based on process corner to optimize power consumption. The XL710 provides additional pins for on-die voltage sensing and the necessary margining circuit.

The 3.3V rail needs to be externally derived from the system's main and/or auxiliary voltages.

15.13.1 VCCA Analog Supply

Figure 15-7 shows an example of the necessary connections to implement the on-die regulator.

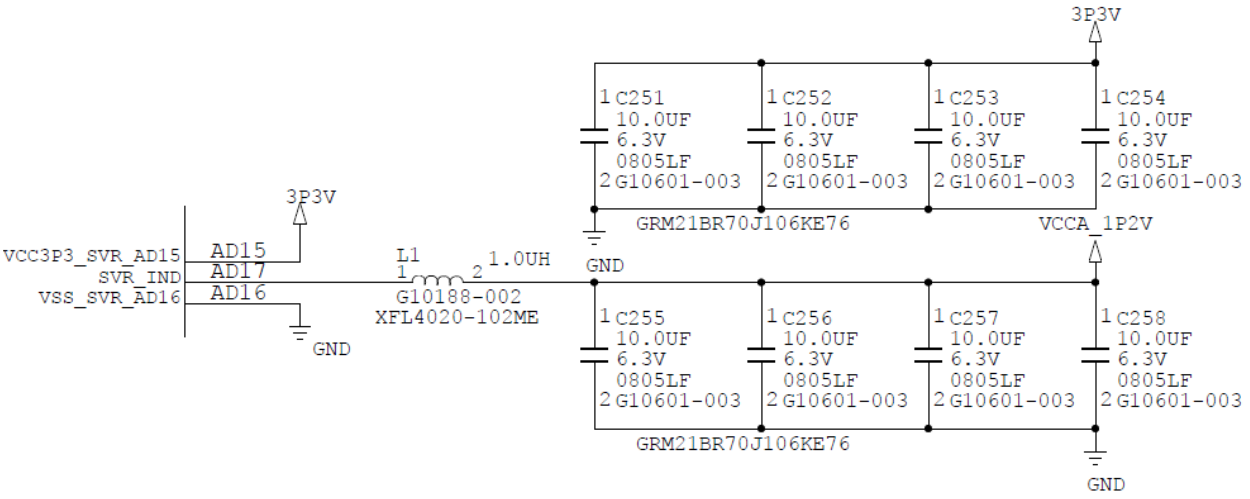


Figure 15-7. Connections to Implement VCCA On-die Regulation.

15.13.1.1 Inductor

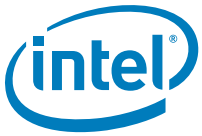
The chosen inductor should meet the following requirements.

- L = 1 uH ±20%
- Irated > 1.6 A
- Isat > 2.2 A
- Rdc < 15 MΩ

When choosing an inductor designers need to take into consideration the inductor package size. To achieve optimal performance, the inductor should be as close as possible to the BGA package; however, certain size limitations might arise due to the geometry of the used heat sink. For a suggested list of inductors refer to Table 15-7.

Table 15-7. Suggested Inductors

Part Number	L [uH]	Irated [A]	Isat [A]	Rdc-typ [mΩ]	Rdc-max [mΩ]
XFL4020-102ME	1	4.5	8	10.8	11.9
IHLP-2525BD-ER1R0M-01	1	9	16	13.1	14.2
IHLM-2525CZ-ER1R0M-01	1	11	22	9	10
IHLP-2525CZ-ER1R0M-01	1	11	22	9	10
MLC1538-102ML	1	12.4	24.5	3.46	3.81
MSS1038-102NL	1	7.3	9		6
SER1052-102ML	1	12.5	16.5		4



15.13.1.2 Output Capacitor

For the on-die switching voltage regulator to operate correctly, the VCCA analog rail should have 40 µF bulk capacitance attached to it. The plane should also have a few 100 nF capacitors attached for high frequency decoupling. The total capacitance of both bulk and high frequency decoupling capacitors, including tolerance, should not exceed 50 µF. It is recommended to use four capacitors 10 µF X7R or two capacitors 22 µF X7R. The total ESR < 1 MΩ and total Z<1.5 MΩ @ F=1 MHz.

15.13.1.3 Input Capacitor

The input of this switching voltage regulator requires 40-100 µF bulk capacitance. It is recommended to use minimum four caps 10 µF X7R or minimum two caps 22 µF X7R. The total ESR<< 1 MΩ and total Z<<1.5 MΩ @ F=1 MHz.

When designing the power source of the 3.3V rail (pins: A10, A15, A19, A6, AD10, AD15, AD19, AD6, E7, L5, P5, Y7) designers should take into consideration the fact that this rail is supplying the input of the VCCA switching voltage regulator (pin AD15). To avoid noise issues due to the input voltage ripple caused by the switching voltage regulator, Intel recommends that designers provide proper isolation between pin AD15 and the rest of the 3.3V power pins. The same principle needs to be applied to the GND pin of the integrated power supply AD16. Refer to the Power Supply Layout Recommendations section for more details.

15.13.2 VCCD Digital Supply

To improve power consumption, the XL710 implements a voltage scaling approach based on process corners. For this to work efficiently, the regulator needs to ensure a 2% or tighter regulation of the VCCD power rail.

15.13.2.1 SVR Controller Selection

Given that the regulator needs to provide an output voltage with a tighter than 2% accuracy, the chosen controller should have a 1% accurate (or better) reference voltage and should be low enough to enable adjusting the output voltage down to 0.75V.

Given the high output current, the design should compensate for the losses presented by the power delivery network. Ideally this should be done by choosing regulators that offer differential remote voltage sensing providing feedback directly from the load. Furthermore, to prevent potential stability issues it is also recommended to use a current mode controller. The LTC3833 and the TPS40180 are good candidates

15.13.2.2 SVR Implementation Example

This section describes a possible SVR implementation designed around the LTC3833 controller (see Figure 15-8).

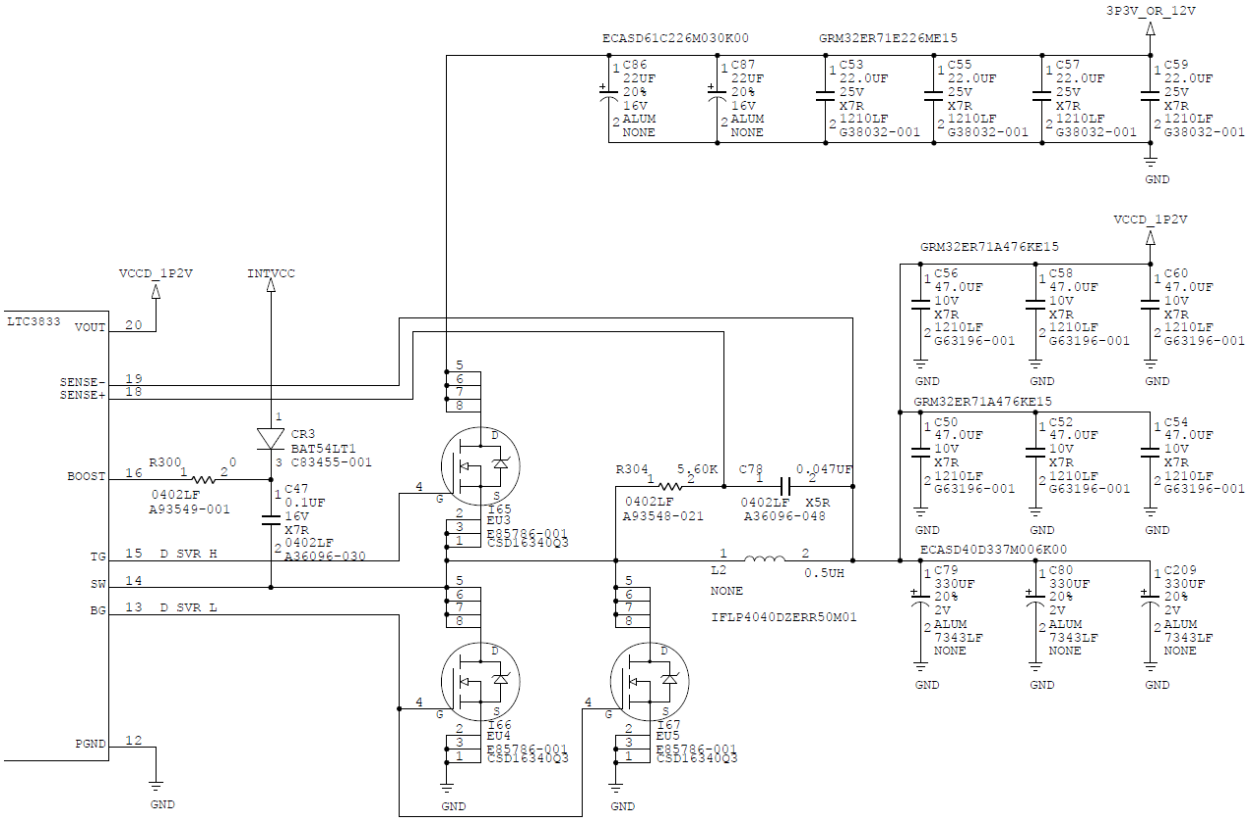


Figure 15-8. LTC3833 SVR Implementation Example

Given that the currently available the XL710 power numbers are only estimates, the power stage of the regulator should be over dimensioned to ensure plenty of margin. Therefore, Intel recommends a regulator design capable of supplying a nominal load of 10 A and peak load of 12 A.

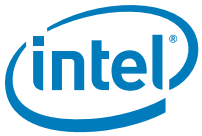
The operating frequency used is 300 KHz to avoid excessive switching losses.

Allowing for about 25% ripple current, the inductor value should be 0.5 µH. It is critical to use an inductor with a low enough DC resistance to avoid over heating. The chosen inductor is the Vishay* IFLP-4040DZERR50M01 with a DC resistance of 0.88 MΩ. If an alternate part is chosen, the core and Rdc losses need to be re-evaluated to ensure thermal stability of the design.

For the top side switching element, the design is using Vishay SIS426DN and for the bottom side switching element the design is relying on 2x Vishay SIS426DN. The TI CSD16340Q3 FETs could be used as alternate parts.

To keep the input voltage ripple under control, the design is relying on a mixture of four Murata* GRM32ER71E226ME15 ceramic capacitors, and two Murata ECASD61C226M030K00 aluminum-polymer capacitors.

To ensure the output voltage ripple is less than 10 mV, the output capacitor tank was chosen to be a mixture of six Murata GRM32ER71A476KE15 ceramic capacitors, and three Murata ECASD40D337M006K00 aluminum-polymer capacitors.



The LTC3833 provides a differential voltage sensing amplifier that should be connected to the EXT_SVR_SENSE_P and EXT_SVR_SENSE_N pins through the previously mentioned feedback circuit. To set the output voltage correctly for the XL710, the value of the resistors are as follows: *R_1=3K, R_2=12K, and R_4=91K* (where R_4 is connected to the INTVCC rail of the LTC3833 controller).

For more information and a complete schematic, refer to the XL710 reference schematics.

15.13.3 Power Supply Sequencing

All regulators need to adhere to the following sequencing rules to avoid latch-up and forward-biased internal diodes: the VCCD rail must not exceed the 3.3 V rail at any moment in time.

The power supplies are all expected to ramp during a short power-up interval (recommended interval 20 ms or faster). Do not leave the XL710 in a prolonged state where some, but not all, voltages are applied.

During the XL710's power on after 3.3V reaches 90% of its final value, the VCCD voltage rail is allowed 100 ms to reach its final operating voltage. Once the VCCD power supply reaches 80% of its final value the 3.3V power supply should always be above 80% of its final value until power down. After 3.3V reaches 60%-100% of its final value, the VCCA voltage automatically rises, independently of VCCD. The final value of VCCA is reached only after a power on internal calibration period.

The use of regulators with enable pins is very helpful in controlling sequencing. Connecting the enable of the VCCD regulator to 3.3V ensures that the VCCD rail ramps after the 3.3V rail. This provides a quick solution to power sequencing. Alternatively, power monitoring chips can be used to provide the proper sequencing by keeping the voltage regulators with lower output in shutdown until the one immediately above reaches a certain output voltage level.

15.13.4 Power Supply Filtering

Provide several 1 µF high-frequency bypass capacitors for each power rail. If possible, place the capacitors close to the load, possibly on the back side of the board directly underneath the BGA package and adjacent to power pads. For a list of recommended mix of bypass capacitors, refer to Table 15-8.

Traces between decoupling and I/O filter capacitors should be as short and wide as practical. Long and thin traces are more inductive and reduces the intended effect of decoupling capacitors. Vias to the decoupling capacitors should be sufficiently large in diameter to decrease series inductance. Alternatively, multiple vias connected in parallel can be used to connect the bulk capacitors to the different power planes.

Table 15-8. Minimum Number of Bypass Capacitors per Power Rail

Power Rail	Bulk Capacitance		High Frequency Bypass
	Cmin [µF]	Cmax [µF]	1 µF
3.3V	84		8
VCCD	132		50
VCCA	40	50	32

15.13.5 Power Management and Wake Up

In order for the XL710 to detect what type of power is available, designers must connect the MAIN_PWR_OK and the AUX_PWR signals on the board. These digital inputs are located in ball locations, AC9 and AB9, and serve the following purposes:

- MAIN_PWR_OK signals the XL710 that the main power from the system is up and stable. For example, it could be pulled up to the 3.3V main rail or connected to a power well signal available in the system.
- When sampled high at power on reset, AUX_PWR indicates that auxiliary power is available to the XL710, and therefore it advertises D3cold wake up support. The amount of power required for the function, which includes the entire network interface card, is advertised in the Power Management Data register, which is loaded from the NVM.

If wake-up support is desired, AUX_PWR needs to be pulled high and the appropriate wake-up LAN address filters must also be set. The initial power management settings are specified by NVM bits. When a wake-up event occurs, the XL710 asserts the PE_WAKE_N signal to wake the system up. This signal remains asserted until PME status is cleared in the Power Management Control/Status Register.

15.14 Thermal Considerations

This section will be included in the next release of the datasheet.

15.15 Layout Recommendations

This section provides recommendations for designing a board layout for the XL710, including general layout guidance, routing the high-speed interfaces, and interface specific recommendations. Guidelines listed in tables are meant to apply to all high speed buses and low speed buses as indicated in each guideline Applicable Buses row. Consult the Intel® XL710 GbE Controller Checklists for more specific layout details.

15.15.1 High-Speed Signal Routing

Table 15-9. Solid Ground Reference

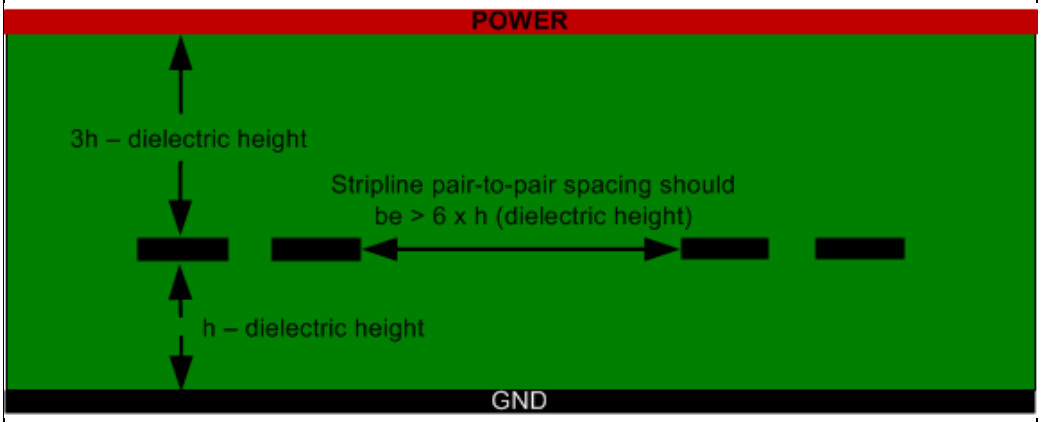
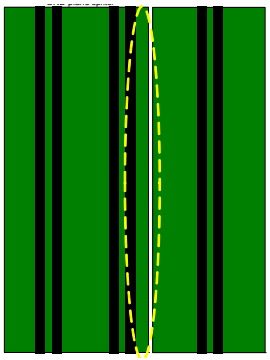
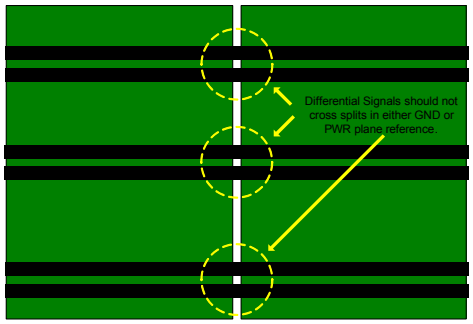
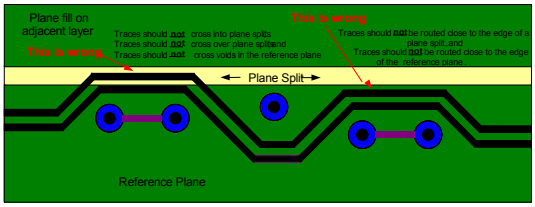
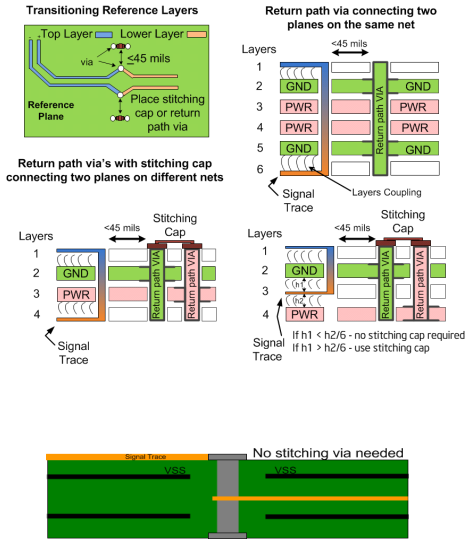
Guideline	Use ground as reference planes. If stripline routing is used and one of the reference planes cannot be ground then use offset stripline, where ground plane separation to signal traces is 1/3 or less than the power plane distance to signal traces. If the differential traces are sandwiched between a power and ground plane, use an offset stripline structure (H to GND, 3H to power), keeping the pairs closer to the ground plane as shown).  Good grounding requires minimizing inductance levels in the interconnections and keeping ground returns short, signal loop areas small, and locating decoupling capacitors at or near power inputs to bypass to the signal return.
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	Reduces coupling between the power bus noise and high speed signals.
Significance	High.

Table 15-10. Clearance From Splits and Edges

Guideline	Signals must not be routed over split ground planes. Signals can be routed over a split power plane for stripline, only if there is a closer solid ground reference plane, and the power plane is at least 3xh away. **(Suggestion to mitigate EMI)** Signal cannot be routed along the edge of a split power plane. 6xh spacing (minimum) from a signal-to-edge of split is required (≥20 mils recommended). **(Suggestion to mitigate EMI)** Limit signal length closeness to plane split to 20 mils or less. (avoid this scenario if possible.) Do not have the signal dwell by a plane edge or route along an edge for any amount of distance. **Note:** (h is closet dielectric height between signal and ground plane). Also, differential signals should be greater than 6 times the dielectric height away from PWR and GND plane splits.    **Warning:** 3xh separation reference rule doesn't protect against potential noisy 12V power plane referencing, since AC noise on 12V plane can be large enough to cause significant xtalk even at 3xh spacing. 12V power plane referencing can only be used if simulation/measurement proves that noise coupling is negligible.
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI
Purpose	Assures the signal integrity of the routing trace. Reduces EMI of the system.

Significance	High. Violation of the rule results in larger noise on the power bus, or EMI failure, or coupling of the power bus noise to the signal traces.						
Min distance signal-to-reference plane edge **MS: microstrip** **SL: stripline**	XAUI	SFI	SGMII/KX	KX4	PCIE 3.0	XLAUI/XLPPI	KR/KR4/CR4
	SL: 5H MS: 6H, non-interleaved or interleaved (Tx and Rx). MS Tx-to-Rx: >9H.	SL: 6H MS: 7H, non-interleaved or interleaved (Tx and Rx). MS Tx-to-Rx: >9H.	SL: 4H MS: 5H, non-interleaved or interleaved (Tx and Rx). MS Tx-to-Rx: >7H.	SL: 5H MS: 6H, non-interleaved or interleaved (Tx and Rx). MS Tx-to-Rx: >7H.	SL: 5H MS: 7H, non-interleaved.	SL: 6H MS: 7H, non-interleaved or interleaved (Tx and Rx). MS Tx-to-Rx: >9H.	SL: 6H MS: 7H MS 7H assumes ≤7 inches on each end of the overall chip-to-chip signal path and stripline in-between.

Table 15-11. Stitching Capacitors/Vias

Guideline	All impedance controlled signals should be routed in reference to a solid plane. If signal traces transition from one reference layer to another reference layer then symmetrically placed stitching capacitors and/or return path vias should be used based on the following: • If the transition is from a power referenced layer to ground referenced layer or from one voltage power referenced layer to a different voltage power referenced layer then stitching capacitors should be used within 45 mils of the transition. • If the transition is from one ground referenced layer to another ground referenced layer or from a power referenced layer to the same net power referenced layer then return path vias should be used within 45 mils of the transition. • At each pair of signal vias, the return path vias should be equidistant to each of the signal vias. Adjacent ground vias must be symmetric with respect to the differential traces. This helps reduce the imbalance that can occur in the different return current paths. 

Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI
Purpose	Provides a clean return path to reduce the probability of having EMI issues, ESD immunity, and IEEE test conformance issues.
Significance	High. If stitching capacitors or return path vias are not used, then it increases the probability of having EMI issues, ESD immunity, as well as some IEEE test conformance issues.

15.15.1.1 AC Coupling Capacitors

Table 15-12. AC Coupling Capacitor Placement

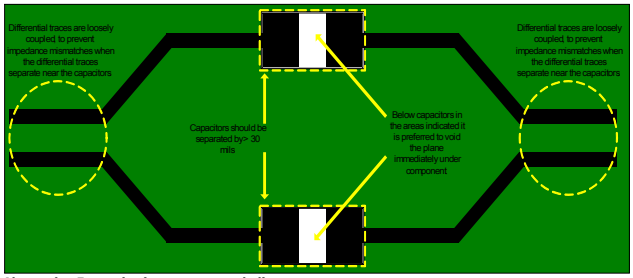
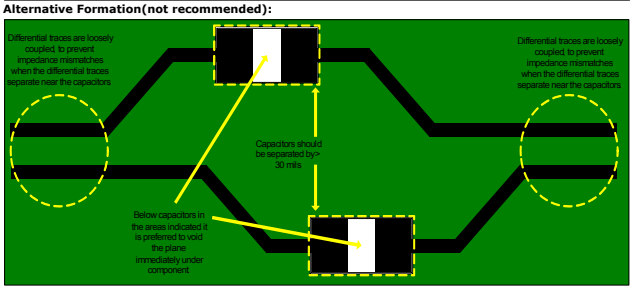
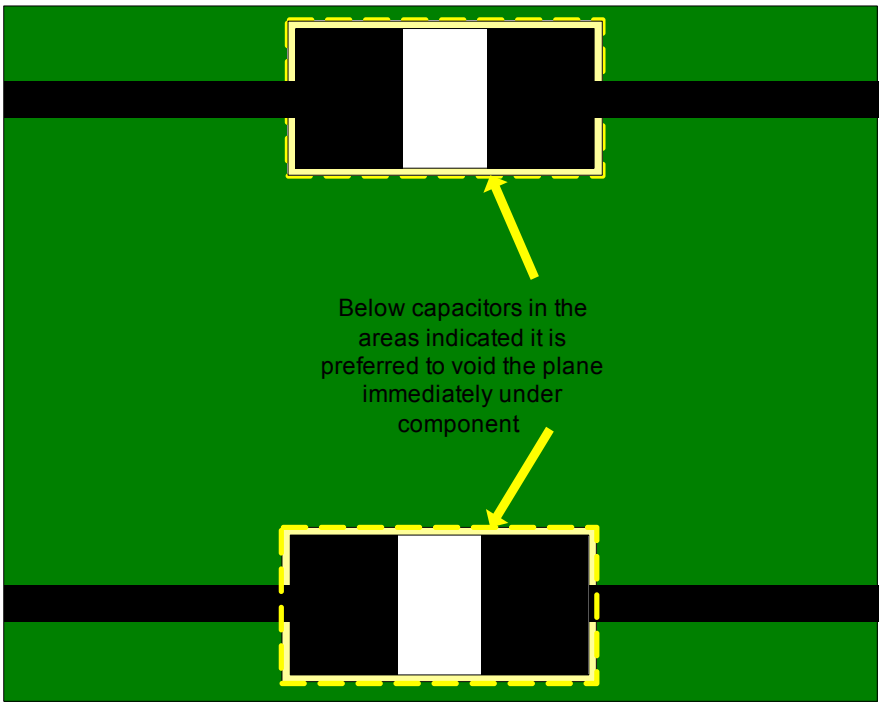
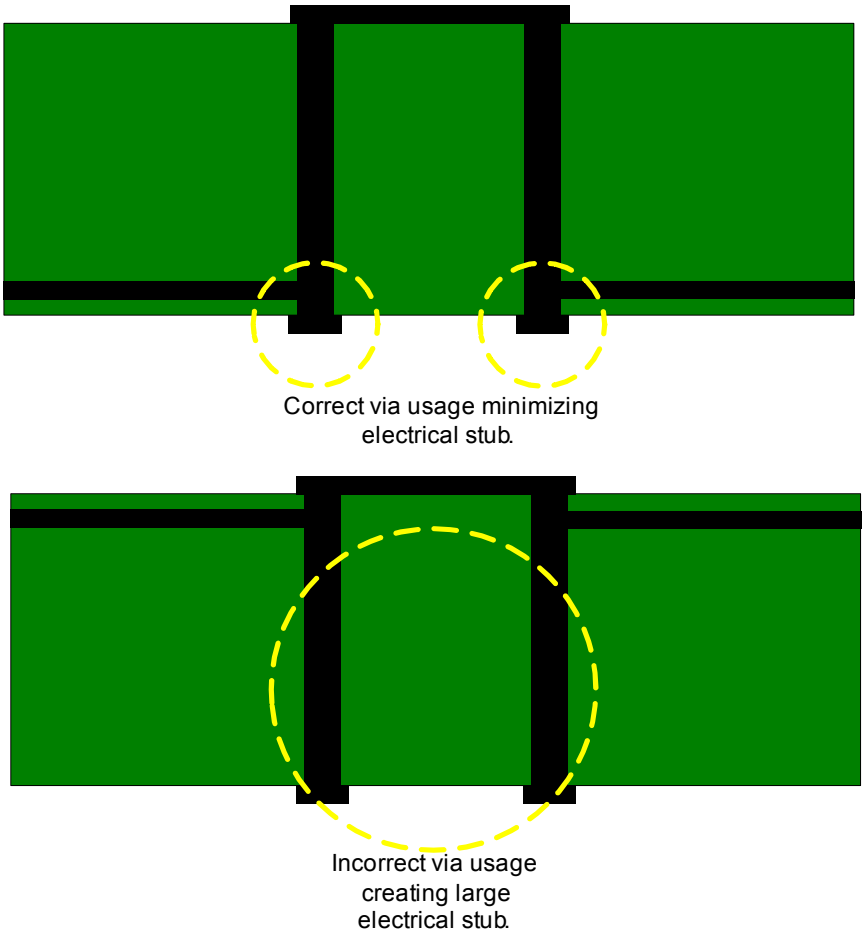
Guideline	Place AC coupling capacitors in the most symmetrical formation as possible. Limit the distance between the capacitor to at least 30 mils. A staggered formation can be used, although not recommended, as an alternative for highly dense platforms with limited space. Grouping discontinuities can help reduce reflections. If the signal needs to change routing layers it is best to do that within 100 mils from the AC coupling capacitors. It is also important that the amount of trace length before and after the capacitor is matched exactly between the two signal lines. For example, if the capacitor on signal D+ is placed 150 mils from the pin of the connector and 2000 mils from the pin of the Rx input on the XL710, the capacitor on the D- signal must also be placed 150 mils from the pin of the connector and 2000 mils from the pin of the Rx input on the XL710. No more than a 5-mil delta should exist between signal pair line segment lengths for either of these trace sections; also, no more than delta should exist for the entire route of the trace. Make sure that decoupled traces follow the nominal single ended impedance, 50 Ω for 100 Ω differential pairs, and 42.50 Ω for 85 Ω differential pairs. Recommended:  **Alternative Formation(not recommended):** 
Applicable Buses	PCIe 3.0/2.0, SGMII/KX, KX4, KR, KR4, CR4, XAUI, XLAUI.
Purpose	Reduces package-to-package parasitic capacitance and common mode conversion.
Significance	Medium. Helps maintain a more consistent transmission line environment. Violation compromises common mode conversion.

Table 15-13. AC Coupling Capacitor Plane Void

Guideline	To reduce shunt capacitance from the AC capacitors' solder pads to the reference plane beneath the solder pads, Intel recommends that designers void the reference plane that is directly under the capacitor. The reference plane void should have the same shape as the capacitor and its solder pads. The size of the reference plane void should be slightly larger than the size of the capacitor and its solder pad. If designers have access to a 3-dimensional field solver, it can and should be used to determine the optimal size and shape for the reference plane void under each capacitor. To prevent noise problems, be careful not to route any adjacent layer traces across the capacitor-shaped voids in the reference plane. 
Applicable Buses	PCIe 3.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	Reduces shunt capacitance created by AC capacitor's solder pads.
Significance	Medium. Helps maintain a more consistent transmission line environment.

15.15.1.2 Vias

Table 15-14. Via Stubs

<table>
<tr><td>Guideline</td><td>Avoid any via stubs if possible; if not possible reduce the size of the stub. The figures that follow show the correct way and topology to be avoided.
 

See Table 15-15 for the total number of vias allowed per interface.</td></tr>
<tr><td>Applicable Buses</td><td>PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.</td></tr>
<tr><td>Purpose</td><td>Reduces the discontinuities (caused by layer transition and via stubs) along the signal path.</td></tr>
<tr><td>Significance</td><td>High</td></tr>
<tr><td>Max Via Stub size(mils)</td><td>
<table>
<tr><td>XAUI</td><td>SFI</td><td>SGMII/KX</td><td>KX4</td><td>PCIE 3.0</td><td>XLAUI/XLPPI</td><td>KR/KR4/CR4</td></tr>
<tr><td>45</td><td>25</td><td>45</td><td>45</td><td>35</td><td>25</td><td>25</td></tr>
</table>
</td></tr>
</table>

Table 15-15. Layer Transitions

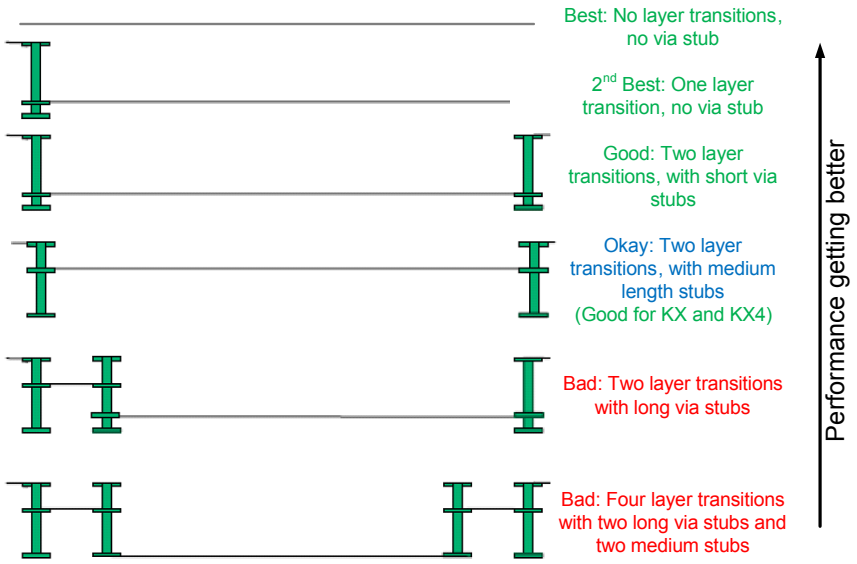
<table>
<tr><td>Guideline</td><td colspan="7">If layer transitions at both Tx and Rx are inevitable, minimize the occurrence of transitions. Route the most risky signals (typically longest channel) first: without layer transition and shortest via stub.
 
</td></tr>
<tr><td>Applicable Buses</td><td colspan="7">PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII.</td></tr>
<tr><td>Purpose</td><td colspan="7">Reduces the discontinuities (caused by layer transition and via stubs) along the signal path.</td></tr>
<tr><td>Significance</td><td colspan="7">High</td></tr>
<tr><td rowspan="2">Total Number of Vias in Path</td><td>XAUI</td><td>SFI</td><td>SGMII/KX</td><td>KX4</td><td>PCIE 3.0</td><td>XLAUI/ XLPPI/CR4</td><td>KR/KR4</td></tr>
<tr><td>6</td><td>2</td><td>8</td><td>6</td><td>4</td><td>2</td><td>6</td></tr>
</table>

Table 15-16. Antipad

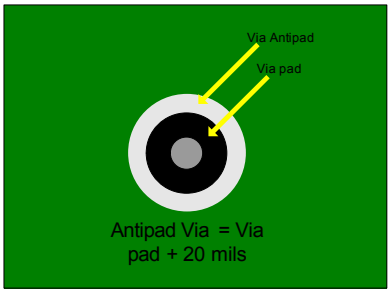
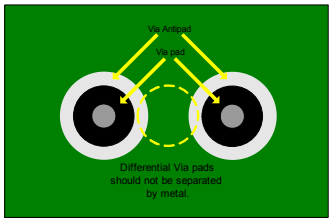
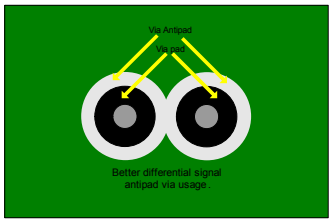
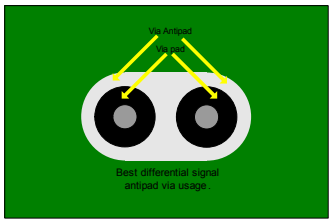
Guideline	The anti-pad diameters should be up to 20 mils larger than the via pad diameters. Clearance between the pad and the surrounding metal should be >= 10 mils. Each time differential signal vias pass through a plane layer, within each differential pair, the anti-pads should overlap or form an oval anti-pad for both vias.  Differential vias should not be separated by metal.  Via anti-pad should overlap.  Best option is having an oval anti-pad. 
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII.
Purpose	Reduces impedance discontinuity.
Significance	Medium. Violation of the rule might result in larger xtalk coupled from vertical vias.

Table 15-17. Inner Via Pads

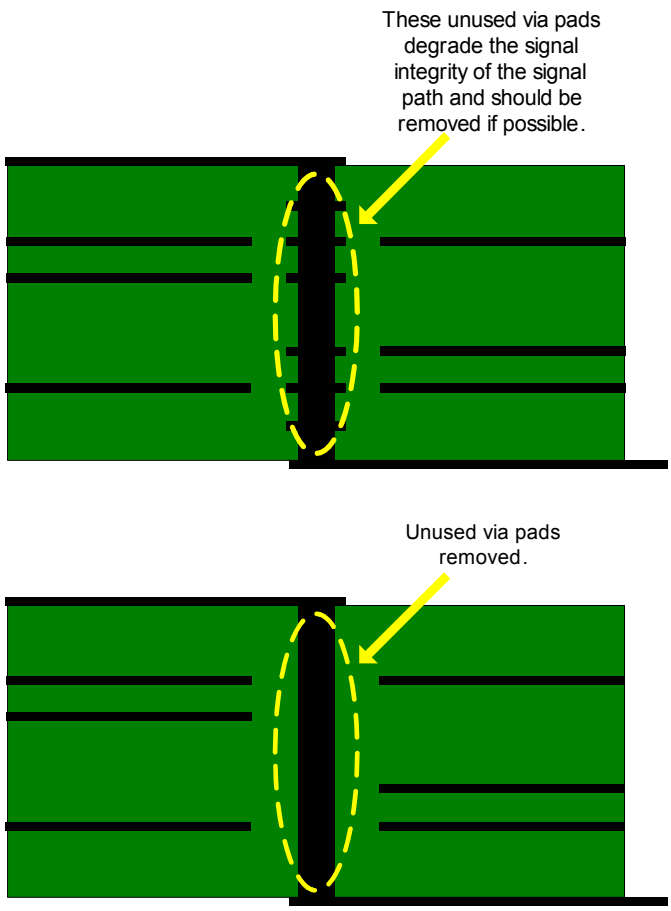
Guideline	If the circuit board fabrication process permits it, it is best to remove signal via pads on unconnected metal layers. 
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	Reduces the via capacitance and improves signal integrity performance.
Significance	Medium. Apply each time possible.

Table 15-18. Differential Via Pitch

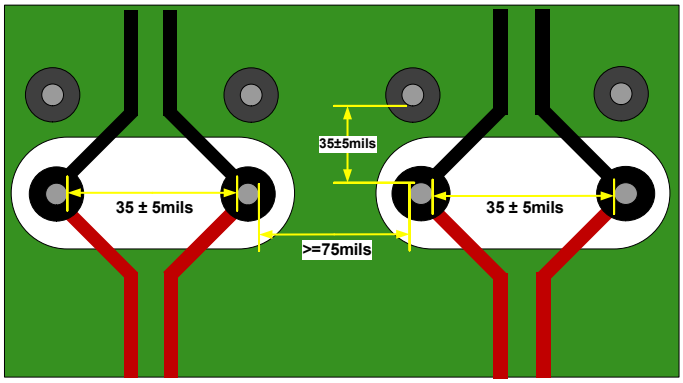
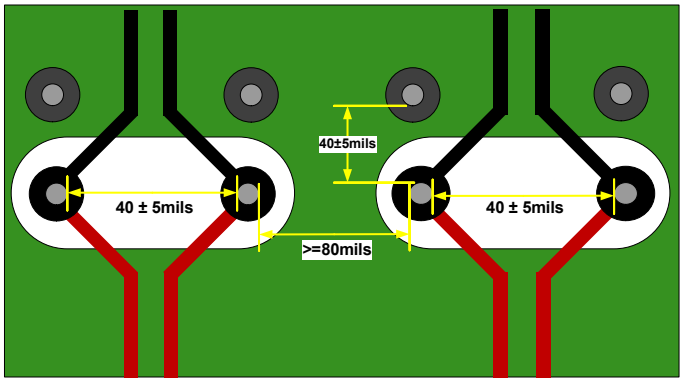
Guideline	The in-pair spacing (barrel edge-to-barrel edge) between signal vias should be in the 30-40 mil range for an 85 Ω differential pair and 35 to 45 mils range for 100 Ω differential pair. The gap between via pairs and other signals vias must be >=80 mils.  **85 Ω transition**  **100 Ω transition** Choose the pitch such that the target differential impedance is maintained on via transition. Simulations are recommended. For instance, using 10/20/40 vias yields a 50 mils center-to-center in pair via spacing, and the spacing center-to-center between other signals should be greater than 80 mils.
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII.
Purpose	Reduces impedance discontinuity and via X-talk at layer changes. This applies to layer changes in open field routing only. This does not apply to BGA or connector routing.
Significance	Medium. Violation of the rule might result in larger xtalk coupled from vertical vias.

Table 15-19. Differential Via Symmetry

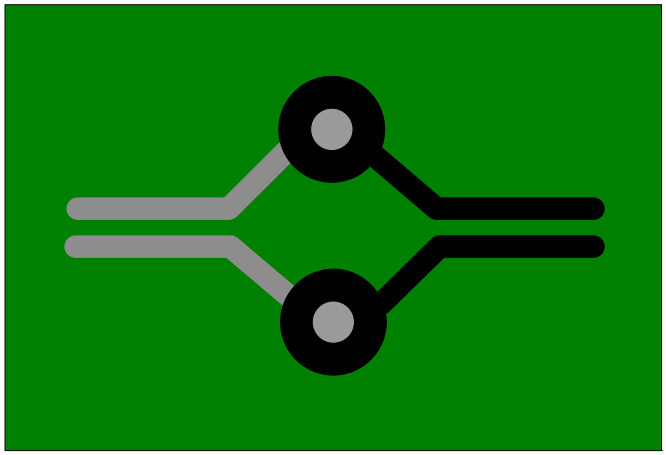
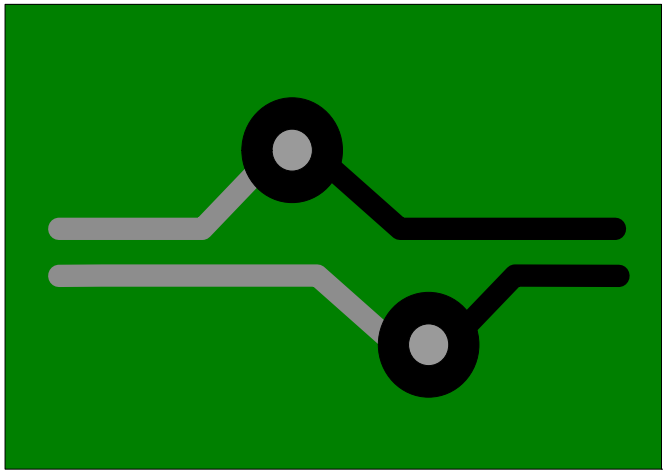
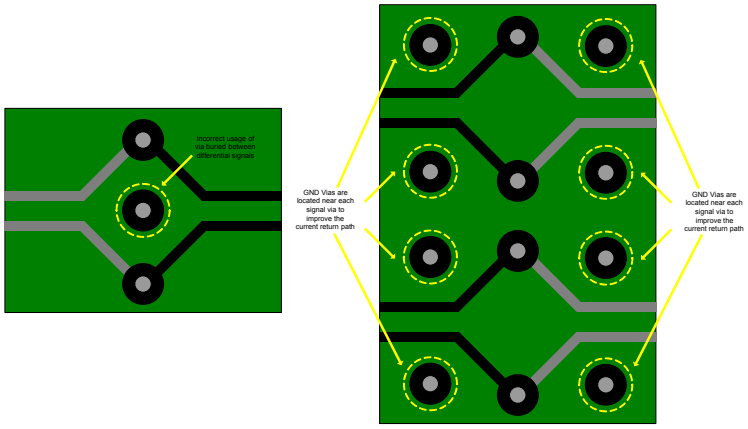
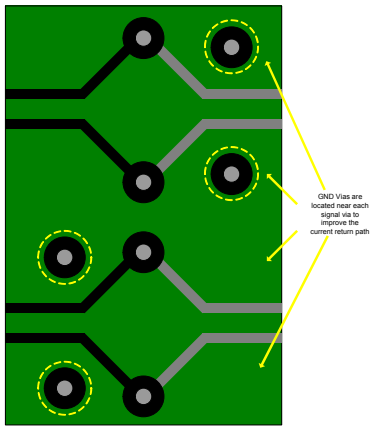
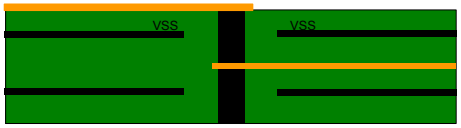
Guideline	It is recommended to place vias symmetric between D+ and D-. In areas where routing density is high, though not recommended the two vias can be slightly offset. But the intra-pair spacing between two vias should be kept close to each other (<=50mil). **Preferred**  **Ok. Avoid** 
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII.
Purpose	Keep the symmetry of the differential via to avoid differential-to-common mode conversion.
Significance	Medium to High. Apply each time possible.

Table 15-20. GND Return Vias

Guideline	Each time differential traces make a layer transition (pass through a pair of signal vias), there must be at least one ground via located near each signal via. Two ground vias near each signal via is somewhat better than just one; however, it's hard to achieve because the design might end up having lower impedance if not done well. Return GND via and signal via gap should between 30 and 50 mils.  **Incorrect**  **Correct; however, risky if not done well.** As shown, if layer transition keeps the same VSS plane (such as, layer transition from layer 1 to 3), then there is no need for return GND vias. 

Applicable Buses	PCIe 3.0/2.0, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI.
Purpose	Reduces the discontinuities, caused by layer transition and via stubs, along the signal path.
Significance	High.

15.15.1.3 Trace Geometries

Table 15-21. Intra-pair Length Matching Requirement

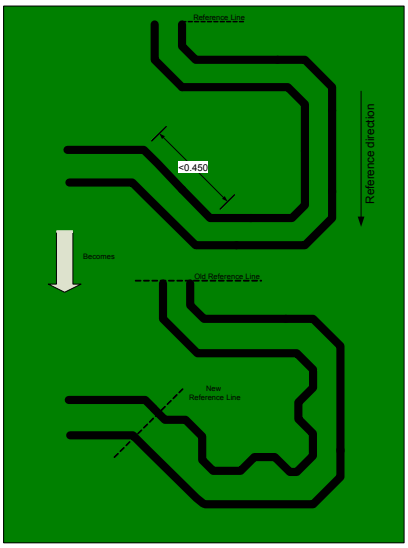
Guideline	If the routing angle exceeds 90° from the reference direction for 450-650 mils, length matching is required. Refer to the end of this table for specific interfaces. 						
Applicable Buses	PCIe 3.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII						
Purpose	Keep symmetry of the differential pair, and avoid differential to common mode conversion.						
Significance	Medium to High. Violation of the rule may result in larger common mode conversion.						
Distance From Source of Skew	XAUI	SFI	SGMII/KX	KX4	PCIE 3.0	XLAUI/ XLPPI	KR/KR4/ CR4
	450	350	600	450	450	350	450

Table 15-22. Intra-pair Matching (No Length Matching Required)

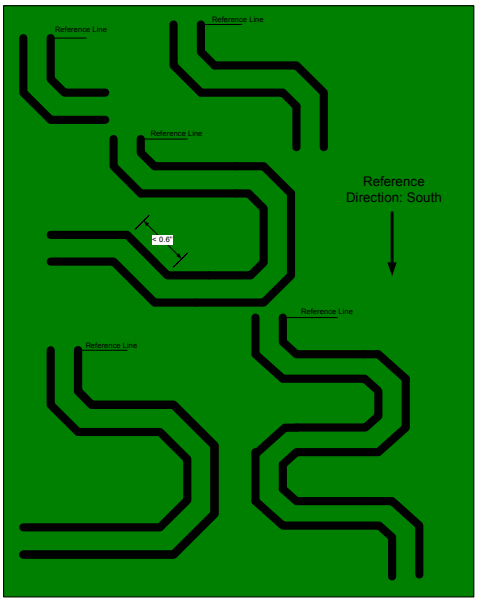
Guideline	As long as the routing angle doesn't exceed 90° from the reference direction for 450-600 mils (depending on the interface; refer to the previous table), no length matching is required. 
Applicable Buses	PCIe 3.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	Keep symmetry of the differential pair, and avoid differential-to-common mode conversion.
Significance	Medium to High. Violation of the rule might result in larger common mode conversion.

Table 15-23. Intra-pair Length Matching (Segment Matching; Running Skew)

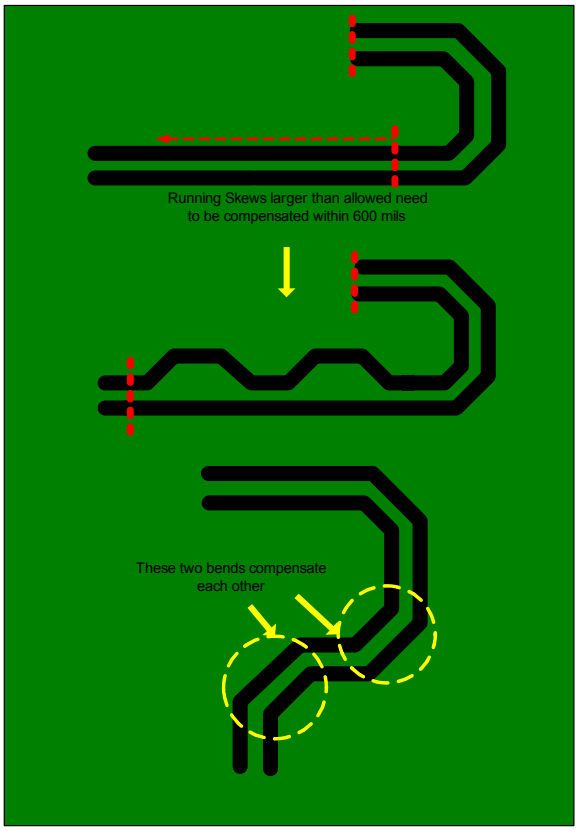
<table>
<tr><td>Guideline</td><td colspan="7">Intra-pair length matching should be enforced per routing layer.
General guidelines:
• Per-layer length matching: <=5 mils between D+ and D-
• Total length matching: <=5 mils between D+ and D-
Keep track of the intra-pair running skew between D+ and D-. Keep the running skew less than 25 mils for multiple board links (in most cases, 25 mils running skew allows for two 45 degree routing bends).
Matching the number of right and left turns and alternating such bends helps to minimize the amount of skew between rising or falling edges of a propagating differential signal pair at any point along the length of the pair.
For skew larger than 25 mils, the miss-match needs to be compensated within 600 mils (see Table 15-22) to either 2 or 5 mils according to the following specifications. The compensation can be done using a bend in the opposite direction, or adding serpentine bumps (wiggles) to the shorter one of the pair. Once the guideline is met, further reduction in the length mismatch of signals within a differential pair to be less than 5 or 2 mils can be done at any location along the routing of the signals.
Refer to the following maximum total intra-pair skew for specific interfaces.
 
</td></tr>
<tr><td>Applicable Buses</td><td colspan="7">PCIe 3.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII.</td></tr>
<tr><td>Purpose</td><td colspan="7">Keep symmetry of the differential pair and avoid differential to common mode conversion.</td></tr>
<tr><td>Significance</td><td colspan="7">Medium to High. Violation of the rule might result in larger common mode conversion.</td></tr>
<tr><td rowspan="2">Max intra pair Skew(mils)</td><td>XAUI</td><td>SFI</td><td>SGMII/KX</td><td>KX4</td><td>PCIE 3.0</td><td>XLAUI/XLPPI</td><td>KR/KR4/CR4</td></tr>
<tr><td>5</td><td>2</td><td>5</td><td>5</td><td>5</td><td>2</td><td>5</td></tr>
</table>

Table 15-24. Serpentines

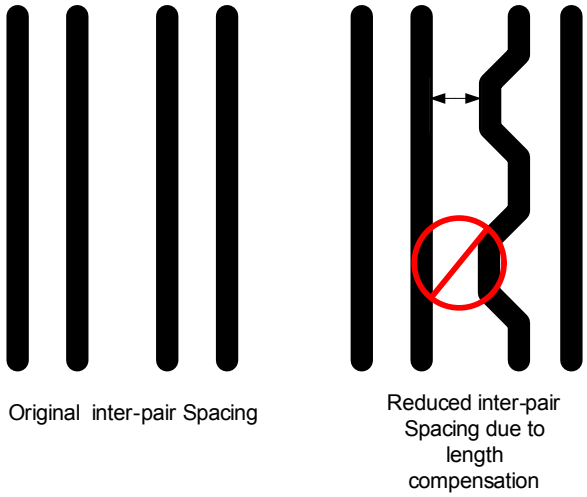
Guideline	The length mismatch compensation techniques should not reduce the inter-pair spacing. 
Applicable Buses	PCIe 3.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	Reduces crosstalk from adjacent traces.
Significance	High. Violation of the rule might result in larger xtalk coupled from nearby traces.

Table 15-25. Matching Within a Bundle

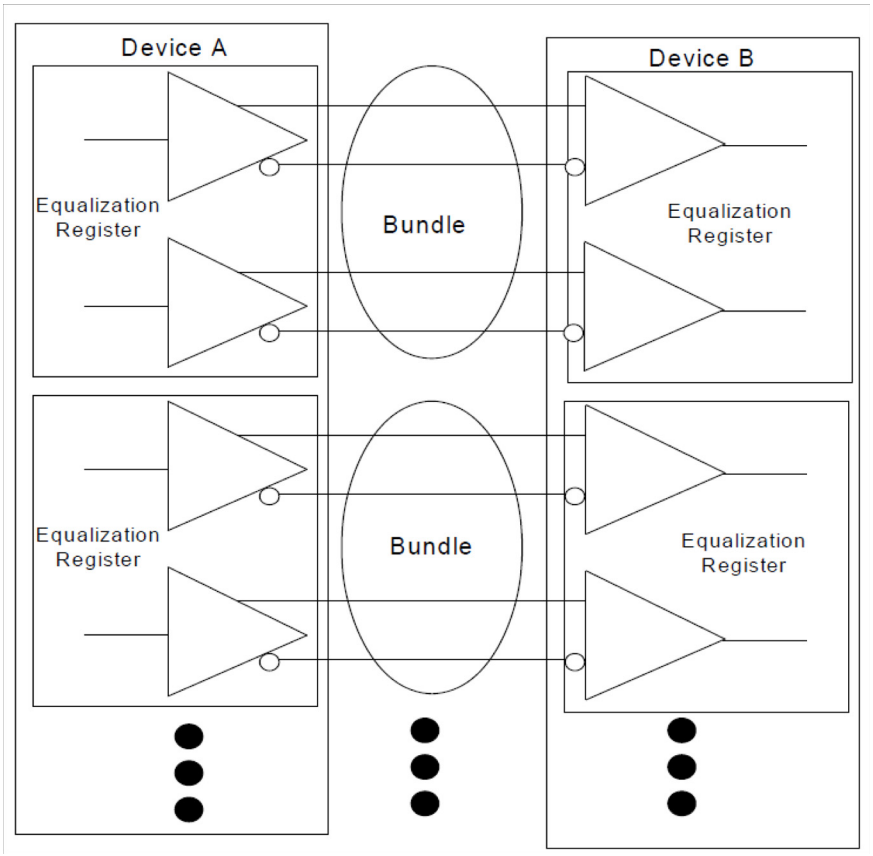
Guideline	Routing bundles are differential pairs that share a common equalization register. Differential pair 0 and 1 is a bundle; differential pair 2 and 3 is a bundle, and so on. Length match pairs in a bundle to within 0.5 of an inch. 
Applicable buses	PCIe 3.0/2.0.
Purpose	Length match within a bundle in order to most effectively use a shared equalization register for that bundle.
Significance	Medium. Violations of this rule might result in sub-optimal equalization register settings.

Table 15-26. Pair-to-Pair Length Matching Serpentine

<table>
<tr><td>Guideline</td><td colspan="7">When doing pair-to-pair length matching, follow this recommendations for doing serpentines.
Self-spacing or width: >=5xh for strip-line; >=7xh for micro-strip-line.
Note: h is dielectric height between signal and ground plane.

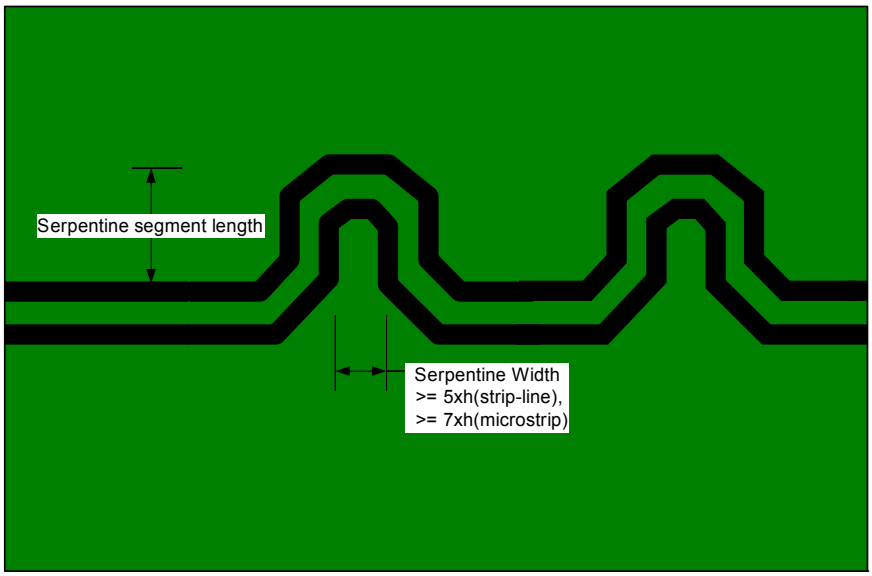
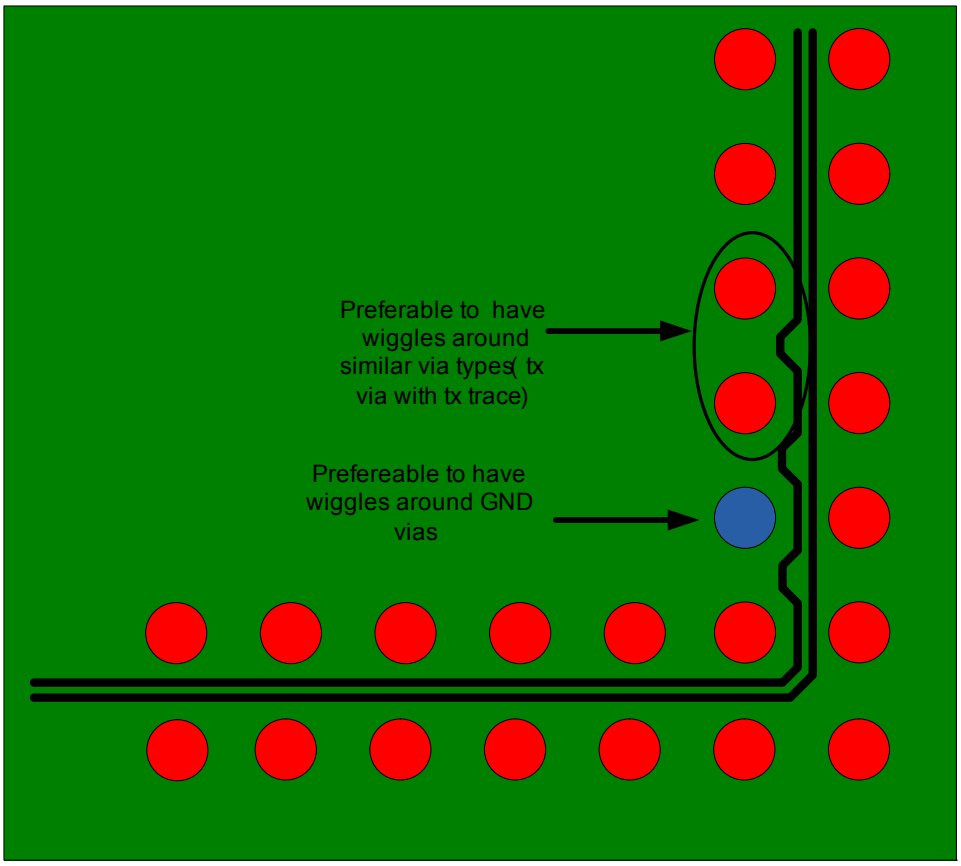
 
</td></tr>
<tr><td>Applicable Buses</td><td colspan="7">PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.</td></tr>
<tr><td>Purpose</td><td colspan="7">Avoid self-coupling between different segments of the serpentine.</td></tr>
<tr><td>Significance</td><td colspan="7">Medium to high. Violation of the rule results in self-coupling between different segments of the routes, and distortion of the signal of the end of the routing. Violation of the rule might also result in slight change of trace impedance.</td></tr>
<tr><td rowspan="2">Pair to Pair Skew(max, mils)</td><td>XAUI</td><td>SFI</td><td>SGMII/KX</td><td>KX4</td><td>PCIE 3.0</td><td>XLAUI/
XLPPI</td><td>KR/KR4/CR4</td></tr>
<tr><td>2000</td><td>N/A</td><td>N/A</td><td>2000</td><td>2000</td><td>Any</td><td>NA/Any/Any</td></tr>
</table>

Table 15-27. Length Matching Through Pin Field

Guideline	Length matching initially should be done outside of the pin field to reduce crosstalk between traces and vias. Length matching in the pin field is generally unnecessary, unless there is not enough room for length matching outside of the pin field. It is preferred to compensate length mismatch within <=400 mils from the edge of the pin field (starting from <= 125 mils from the edge of pin field). When length matching is implemented within the pin field, place the wiggles near ground via or similar via types (tx-via over tx-trace or rx-via over rx-trace). 
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	Reduce the crosstalk between differential trace routing and vertical vias.
Significance	Medium. Violation of the rule might result in larger xtalk coupled from vertical vias.

15.15.1.4 Breakout

Table 15-28. Trace Necking

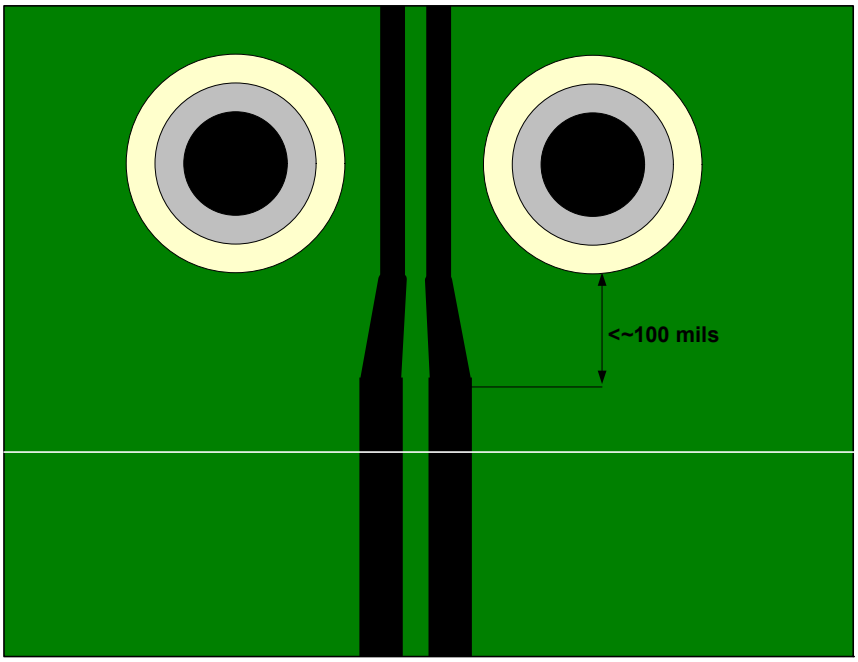
Guideline	When a differential pair escapes the pin field area with narrow traces, it needs to fan out symmetrically to wider trace within 100mils from the edge of pin field. Intel recommends creating a <45 degree angle taper segment between main route and breakout. 
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	Reduces impedance discontinuity and trace loss.
Significance	Medium. Fan out within 100 mils helps to reduce the overall trace loss since narrow traces typically have higher conductor loss. Low. A taper segment helps reduce impedance discontinuity.

Table 15-29. Uncoupled Length

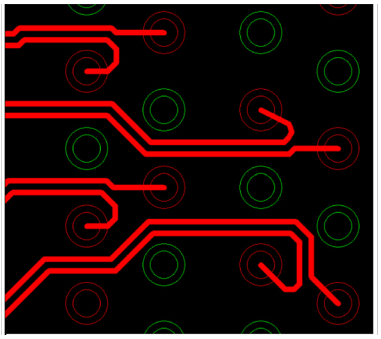
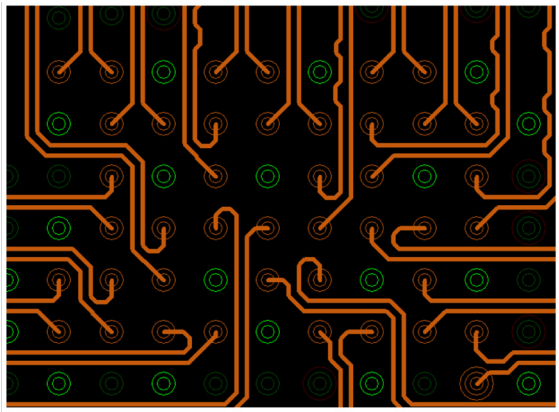
Guideline	Differential signals are preferred to be routed symmetric. However, when a differential pair is assigned diagonally at package pin or escapes from connector, one trace of differential traces see a longer uncoupled line. Maintain <=70 mils (35 mils typical) for uncoupled line length for pin field escape and <=100 mils (70 mils typical) for connector escape (measured from the edge of pad). Escape from pin field sees short routes over anti-pads and it should be the same length between D+ and D-. Connector Example  BGA Escape Example 
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	To reduce impedance discontinuity and trace loss.
Significance	Medium to high. Violation of the rule results in larger common mode noise, which impacts the overall system signal integrity performance and EMI performance.

Table 15-30. Routing Through Pin Field

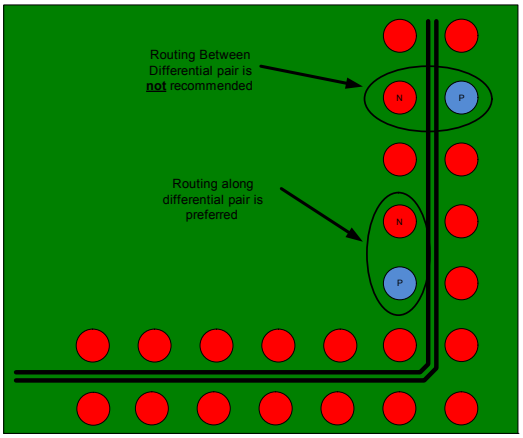
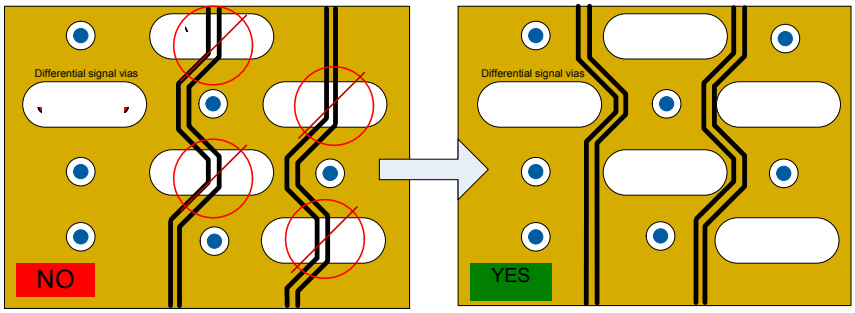
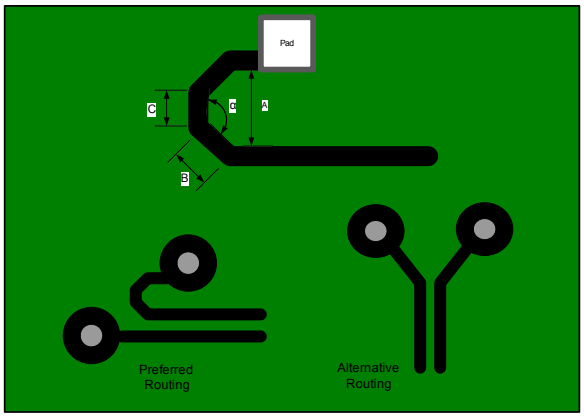
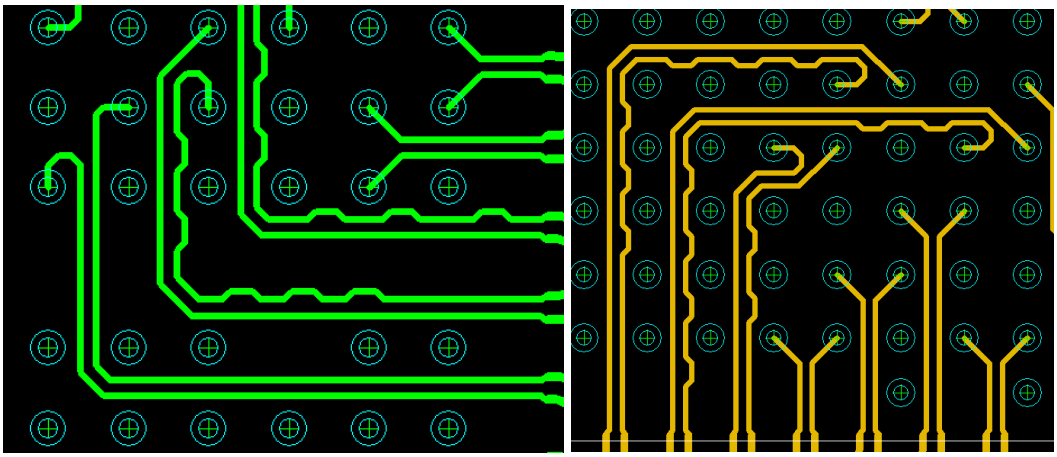
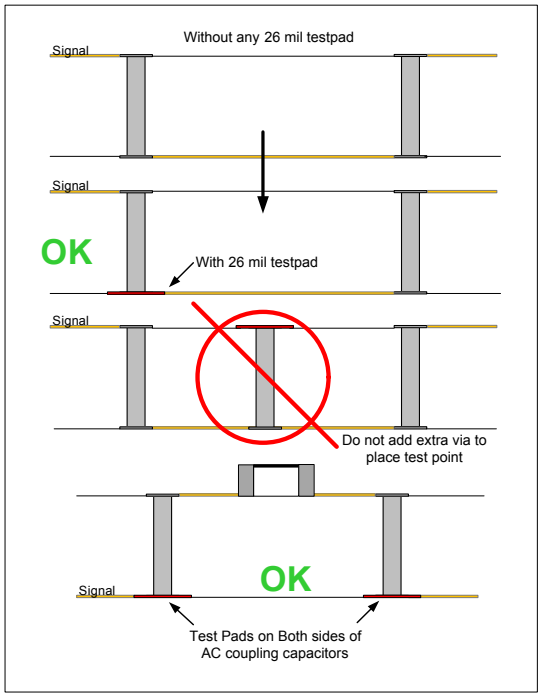
Guideline	1. It is preferred to reduce routing length under the pin field. Maximum length under the pin field is 1.5 inches 2. Avoid routing over pin fields that have a high magnitude of transient currents (such as power delivery pin fields). 3. Length matching initially should be done outside of the pin field to reduce crosstalk between traces and vias. Length matching in the pin field is generally unnecessary, unless there is not enough room for length matching outside of the pin field. 4. It is preferred to compensate length mismatch within <=400 mils from the edge of the pin field (starting from <= 125 mils from the edge of pin field). Route differential trace along one side of a differential via pair.  
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	Reduces the amount of noise picking up from the vertical vias and reduce the mode conversion (from differential model to common mode).
Significance	1. High. Violation of the rule might result in excessive xtalk coupled from vertical vias, causing eye closure. 2. Medium. Violation of the rule results in larger common mode noise, which impacts the overall system signal integrity performance, and EMI performance.

Table 15-31. Pin Field Escaping

Guideline	Refer to the following diagrams for reference. For asymmetric escapes, use bends shown in the diagrams and minimize uncoupled length. A >= 3 x h (dielectric height) α>=135 degrees B, C >= 1.5 x w (trace width) **Note:** h is dielectric height between signal and ground plane.  
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	Reduces impedance discontinuity and mode conversion (from differential model to common mode).
Significance	Medium to high. Violation of the rule results in larger common mode noise, which impacts the overall system signal integrity performance and EMI performance.

15.15.1.5 Test Points

Table 15-32. Test Point or Feature

<table>
<tr><td>Guideline</td><td colspan="7">Intel does not recommend using a via test point or feature on high speed interfaces because they introduce significant impedance discontinuities to the channel.
There is no need for a test point or feature if there is a connector (use a connector pin as test point).
If a test point must be used, place a via test pad on an already present layer transition via. Make sure pad size is not greater than 26 mils in diameter. Make sure a larger test pad is only in one of the outer layers.
Do not add extra vias to implement a test pad.
Via test pads are done for the purpose of a trace connectivity test. Test pad only needs to be implemented on one end of the link.
Test pad can be placed at the vias for the AC capacitor (such as a DC blocking capacitor).
Increase the differential via spacing to accommodate the probe pad spacing.
 
</td></tr>
<tr><td>Applicable Buses</td><td colspan="7">PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.</td></tr>
<tr><td>Purpose</td><td colspan="7">To reduce impedance discontinuity.</td></tr>
<tr><td>Significance</td><td colspan="7">Medium to high.</td></tr>
<tr><td rowspan="2">Via Test Pad Allowed?</td><td>XAUI</td><td>SFI</td><td>SGMII/KX</td><td>KX4</td><td>PCIE 3.0</td><td>XLAUI/XLPPI</td><td>KR/KR4/CR4</td></tr>
<tr><td>yes</td><td>no</td><td>yes</td><td>yes</td><td>yes</td><td>no</td><td>yes</td></tr>
</table>

15.15.1.6 Signal Isolation and Clearance

Table 15-33. Clearance From Vias

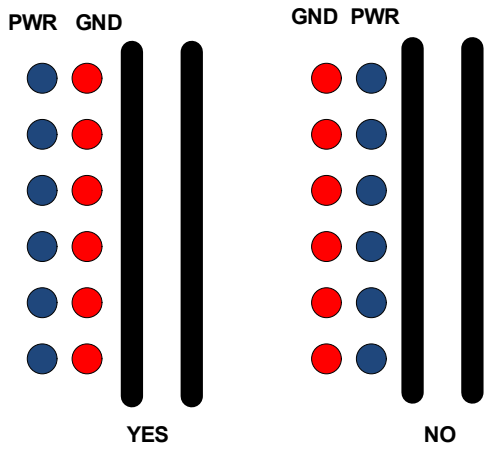
Guideline	Recommendation for minimum signaling impact: 1. 5xh for fast-switching power signals (>1A/ns). *Allowed Violations* - 2 (max) allowed at >3xh if and only if the signal not routed at worst case conditions (such as max length, longest via stub, etc.) This assumes that no ground sharing with high di/dt signals. 2. 4xh for other high-speed signals. *Violations* - 2 (max) allowed at >1xh. 3. 3xh for static, low-current signals or same type of signal – Rx or Tx. *Violations* - 4 (max) allowed at >1xh. 4. >1xh for mounting hole anti-pads. If the signal has violations in several or all the cases previously mentioned, then the maximum count is not applicable and the situation should be reviewed by a signal integrity expert. Other guidelines: Power supply vias should have VSS vias in proximity (<= 50 mils). Intel does NOT recommended routing through a via field (power or other type of signals) except to escape routing from Tx and Rx. If many power supply vias (more than 5) are placed as a via field, then VSS vias should be paired to Vcc vias and VSS vias should be placed between signal traces and power supply via field. **Note:** h is dielectric height between signal and ground plane. 
Applicable Buses	PCIe 3.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII.
Purpose	To minimize impedance discontinuity and assure the signal integrity of the routing trace.
Significance	High. Impact on trace impedance discontinuity and xtalk.

Table 15-34. Routing Clearance From VRM

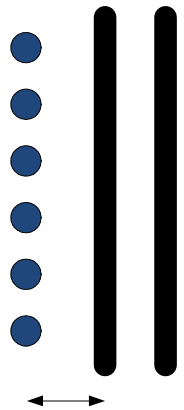
Guideline	High speed differential traces should be routed 10h away from the VRM vias that have high dv/dt switching noise (typically phase nodes in VRM). **Note:** h is dielectric height between signal and ground plane. At least 10h away from edge of anti-pad for phase node vias (with high dV/dt switching noise). 
Applicable Buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.
Purpose	Assures the signal integrity of the routing trace.
Significance	High. Violation of the rule results in noise coupling from the VRM via switching noises.

Table 15-35. Same Layer Clearance

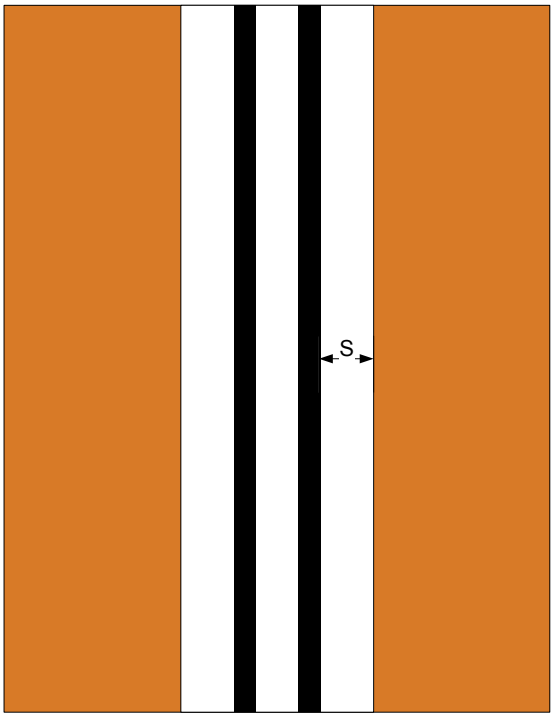
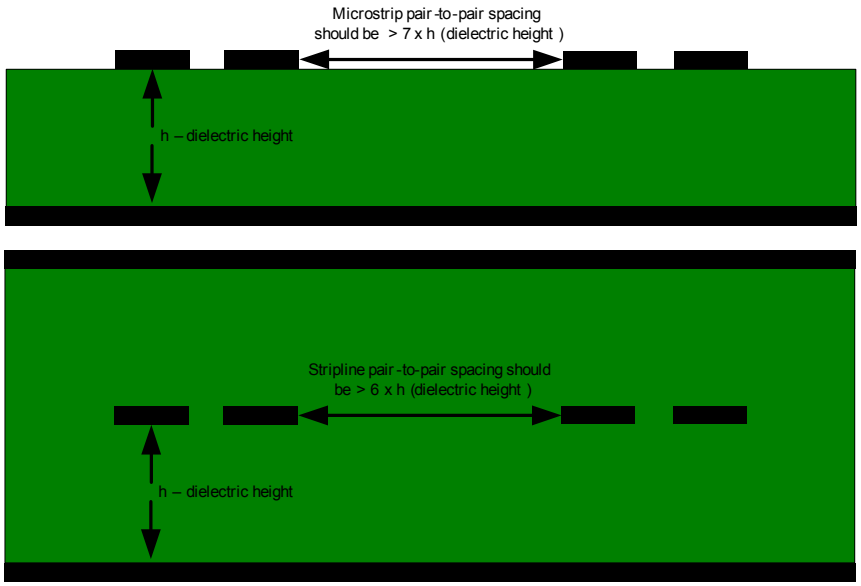
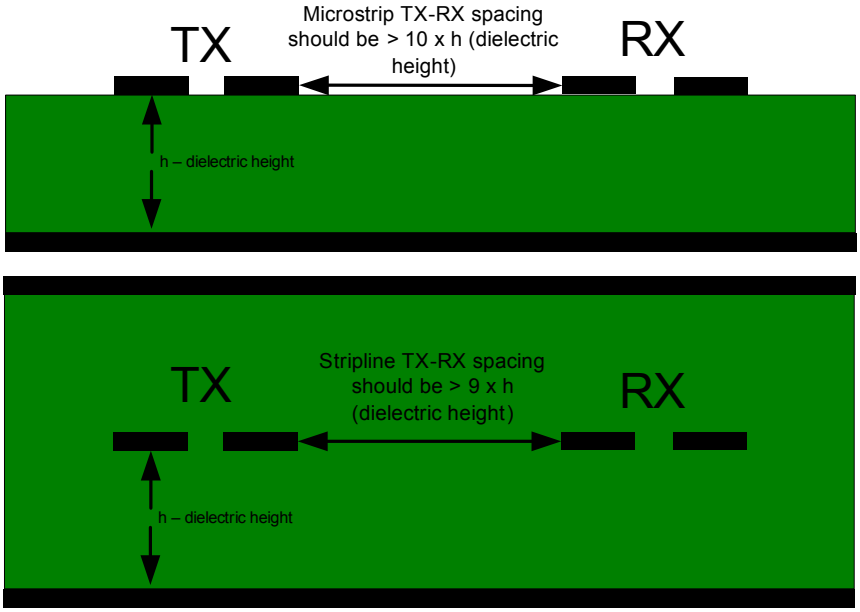
<table>
<tr><td>Guideline</td><td colspan="7">Need to keep S >= 10 x h from the edge of plane to the edge of trace for long signal traces routed in parallel to the edge of planes (or partial planes).
Note: h is dielectric height between signal and ground plane.
 
</td></tr>
<tr><td>Applicable Buses</td><td colspan="7">PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.</td></tr>
<tr><td>Purpose</td><td colspan="7">Reduces the system noise on the power bus.</td></tr>
<tr><td>Significance</td><td colspan="7">High. Violation of the rule results in larger noise on the power bus or coupling of the power bus noise to the signal traces.</td></tr>
<tr><td rowspan="2">Distance from signal to reference plane edge</td><td>XAUI</td><td>SFI</td><td>SGMII/KX</td><td>KX4</td><td>PCIE 3.0</td><td>XLAUI/ XLPPI</td><td>KR/KR4/ CR4</td></tr>
<tr><td>5X</td><td>10X</td><td>5X</td><td>5X</td><td>7X</td><td>10X</td><td>7X</td></tr>
</table>

Table 15-36. Clearance Between Nets

<table>
<tr><td>Guideline</td><td colspan="7">As a general rule, spacing between pairs should be greater than 7 times the dielectric height for microstrip and 6 times the dielectric height for stripline as listed at the bottom of this table.
 

This rule does not apply for Tx-to-Rx pair spacing. See Table 15-37 for Tx/Rx isolation.</td></tr>
<tr><td>Applicable Buses</td><td colspan="7">PCIe 3.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII, NC-SI.</td></tr>
<tr><td>Purpose</td><td colspan="7">Reduces crosstalk between traces.</td></tr>
<tr><td>Significance</td><td colspan="7">High.</td></tr>
<tr><td rowspan="2">Recommended Pair-to-Pair separation</td><td>XAUI</td><td>SFI</td><td>SGMII/KX</td><td>KX4</td><td>PCIE 3.0</td><td>XLAUI/XLPPI</td><td>KR/KR4/CR4</td></tr>
<tr><td>SL: 5H
MS: 6H</td><td>SL: 6H
MS: 7H</td><td>SL: 4H
MS: 5H</td><td>SL: 5H
MS: 6H</td><td>SL: 6H
MS: 7H</td><td>SL: 6H
MS: 7H</td><td>SL: 6H
MS: 7H</td></tr>
</table>

Table 15-37. Tx and Rx Isolation

Guideline	Except in the XL710 and connector break-out areas, routing for Tx and Rx lanes should either be on separate signal layers, or if they are routed on the same layer, then edge-to-edge separation between Tx pairs and Rx pairs should be >= 10 times the thinnest adjacent dielectric height for microstrip and 9h for stripline. 
Applicable Buses	PCIe 3.0, KR, KR4, CR4, XLAUI, XLPPI, SFI.
Purpose	Reduces crosstalk between traces.
Significance	Medium.

15.15.1.7 Connectors

Table 15-38. Ground Plane for Edge Connector Card

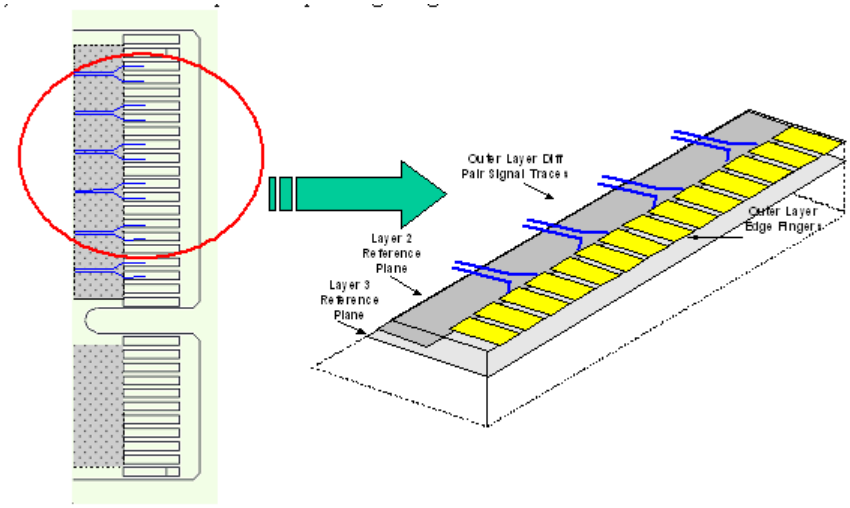
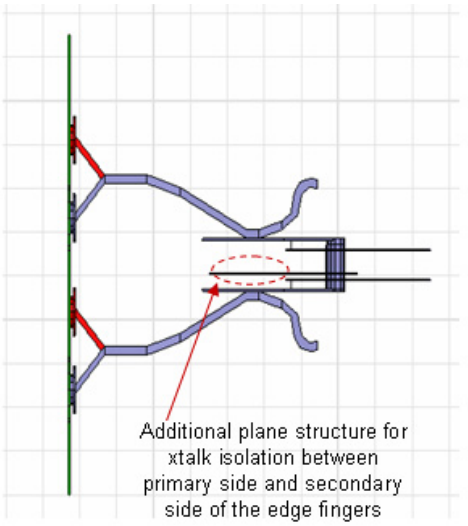
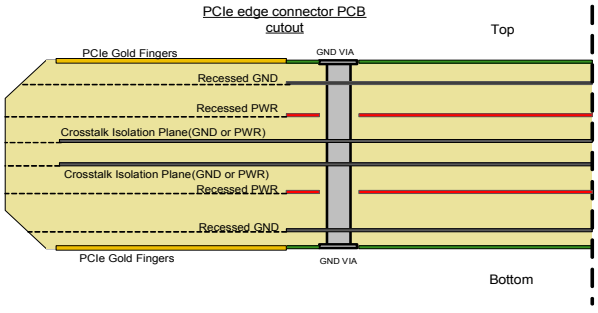
Guideline	If the edge connector is used, then the ground plane underneath (layer 2 in the following diagram as an example) should be recessed up to the tip of the edge fingers. 
Applicable Buses	PCIe 3.0/2.0.
Purpose	Recess the ground plane helps to reduce the insertion loss (S21) by reducing the excessive shunt capacitance between ground plane and the edge fingers.
Significance	Medium. Recessing the ground plane improves the signal integrity performance significantly.

Table 15-39. Crosstalk Control on Connector Card

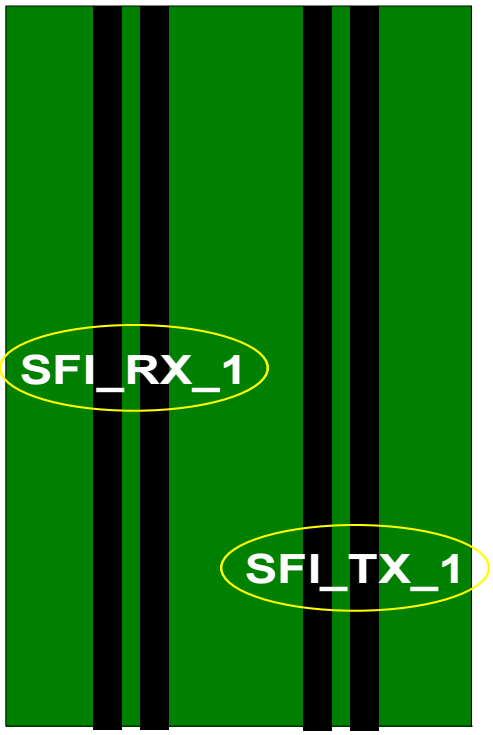
Guideline	Two approaches to control the crosstalk of the edge fingers between primary side and secondary side:
	1. Using the 1:1 S:G ratio pattern.
	`Primary side  s s g g s s g g s s g g`
	`Secondary side  g g s s g g s s g g s s`
	2. If 2:1 S:G ratio is used, then additional plane structures are recommended between primary side and secondary side of the edge fingers, as shown in the following diagrams. Two scenarios are available:
	If the two layers closest to the middle of layer stack up are power planes, then the power plane can be fully extended to the full length of the edge finger.
	If any of the two planes at the middle of stack-up are signal planes, a small section of metal plane can be added on the same signal layer, and shorted to the ground planes by shorting vias.
	 
	 

Applicable Buses	PCIe 3.0/2.0.
Purpose	Controls the crosstalk of the edge fingers between primary side and secondary side.
Significance	Medium. It is strongly recommended for 6.4 Gb/s or above operation.
Implementation	PCIE uses 1:1 S:G ratio.

Table 15-40. Board Impedance

Guideline	Trace impedance targets are: • 85 Ω nominal impedance PCIe • 100 Ω nominal impedance for High speed differential buses Advantages of 85 Ω design vs. 100 Ω design for PCIe: • Less impedance discontinuity (via, socket, PTH, Cpad) • Easy for manufacturing (allow wider trace and/or tighter coupling), and less impedance variation. • Allow for thinner board and/or denser routing Less conductor loss (when wider trace is applied) and keeping density same as 100 Ω. The differential impedance tolerance is dependent on the interface. Refer to the remainder of the table for recommended impedance tolerances.						
Applicable Buses	PCIe 3.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII.						
Purpose	Assures matched (or close to matched) impedance for the link and improve link performance.						
Significance	High. Violation of the rule results in reduced solution space and increased routing difficulty.						
Impedance Tolerance (+-%)	XAUI	SFI	SGMII/KX	KX4	PCIE 3.0	XLAUI/XLPPI	KR/KR4/CR4
	15	10	17	15	15	10	10

Table 15-41. Silk Screen

Guideline	Avoid having silkscreen over high speed signal traces because this causes impedance discontinuities. **Incorrect: No Silkscreen Over High Speed Traces**  This rule is more critical for interface frequencies faster than 2.5GHz.
Applicable Buses	PCIe 3.0/2.0, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI.
Purpose	Reduces reflections due to impedance discontinuities cause by ink's dielectric properties.
Significance	Low to medium. Violation of the rule results in signal degradation.

15.15.1.8 Dielectric

Table 15-42. Use Low Er and Low df Materials

Guideline	Intel recommends using dielectric materials adjacent to critical traces with a low Er (lower dk) and low dissipation factor (df) to enable wider traces, reduce trace insertion loss, and enable the same connector solder-pad geometry and to achieve ~100+ Ω differential. A high Er dielectric layer causes traces to be too narrow and to have higher insertion loss than is desirable. When choosing which dielectric material to use, consider the maximum trace length that must be routed and the cost and availability of the dielectric material.
Valid buses	PCIe 3.0/2.0, KX, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI, SGMII.
Purpose	Allow for wider traces.
Significance	Medium. Violation of this rule might result in increased insertion loss and smaller eye opening.

Table 15-43. Fiber Weave Effect

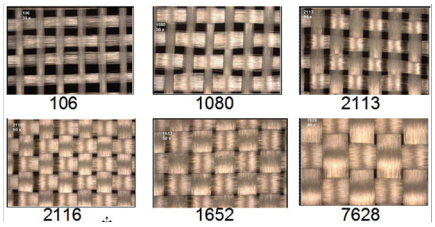
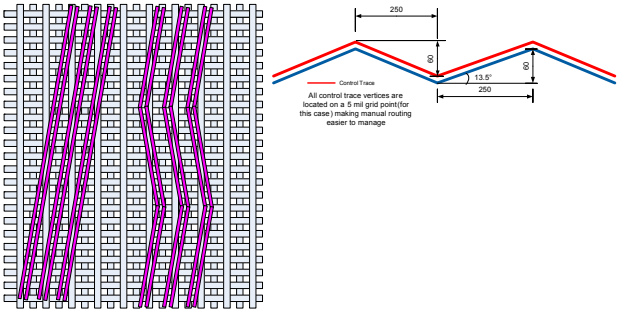
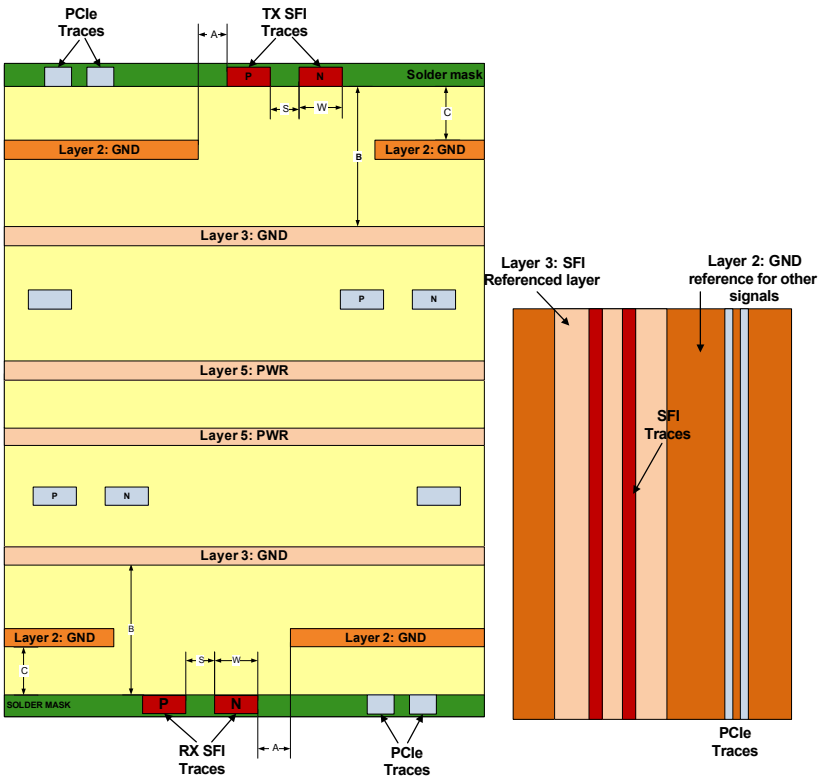
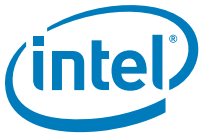
Guideline	Typical Printed Circuit Boards (PCBs), due to their basic construction consisting of woven fiberglass fabric (Er~6) strengthened and bound together with epoxy resin (Er~3.5), present a non-homogeneous medium for signal propagation of differential pairs. There are several weave types, with the weave strands running horizontal and vertical relative to the board edges. For differential pairs with trace width and spacing comparable to the dimensions of the fiberglass cloth, the PCB's non uniform dielectric can give rise to substantial propagation differences between the conductors of the pair. Instead of routing traces parallel to either X or Y axis, long traces with a Root Sum Square (RSS) length >2.25 inches should be routed at an angle of 12-to-35 degree to the weave as shown. The center-to-center pitch of the traces within the differential pairs can be matched to the weave pitch of the dielectric material. If designers plan to use a woven glass/epoxy dielectric material, check with the material supplier to find out the glass weave pitch prior to doing final differential trace routing. $$RSS = \sqrt{(H_1^{\,2} + H_2^{\,2} + \ldots + V_1^{\,2} + V_2^{\,2} + \ldots)}$$ H = horizontal trace segment lengths. V = vertical trace segment lengths. Traces can be routed to include a series of 12-to-35 degree bends relative to the board, with bends separated by several tenths of an inch, to shift the traces in steps by a few millimeters each time. There should be an equal number left turns and right turns along the length of the traces. Also, thicker dielectrics or more plies enable wider traces and diminishes the weave effect. Alternatively, CAD artwork can be rotated with respect to the weave. Some common weave patterns. Using denser weaves diminishes the need for this rule.  Critical traces are carefully routed at >12 degree angle. 
Applicable Buses	PCIe 3.0/2.0, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI.
Purpose	Minimizes signal skewing, impedance discontinuities, and mitigate eye opening reduction to fiber weave effect.
Significance	Medium. Violating might result in violation of AC common mode specifications.

Table 15-44. Dielectric Thickness

Guideline	Dielectric thickness is a factor to achieve target impedance. Having thicker dielectric thickness enables traces to achieve 100/85 Ω impedance while maintaining a minimum trace width to reduce trace resistivity, insertion loss and manufacturing trace width variation. In many cases achieving 100 Ω impedance with wider traces is not feasible as the dielectric thickness is not ideal for high speed interfaces in most platforms. A way to work around this issue is to void the plane right underneath the traces (layer 2 as shown) and use layer 3 as the reference plane. This method should be carefully done ensuring that the **A** distance is large enough to ensure there is no coupling between signals and layer 2. As shown, **Dimension A** must be at least four (4x) times the adjacent dielectric **height C**.  Work around thin dielectrics. Hard to achieve hence not recommended for every design. Consult Intel signal integrity engineers if an alternative method is being on considered in a design.
Applicable Buses	PCIe 3.0/2.0, KX4, KR, KR4, CR4, XAUI, XLAUI, XLPPI, SFI.
Purpose	Allow for wider traces.
Significance	Medium. Violation of this rule might result in increased insertion loss.



15.15.1.9 Recommended Simulations for High Speed Serial Interconnects

KR, KR4, SFI, XLAUI, XLPPI, and CR4 signaling frequencies extend above 5 GHz. Relatively short stubs, small discontinuities, and fairly small in-pair trace length differences can cause signal integrity issues and an undesirable increase in bit errors. Before ordering circuit boards, verify that:

- Planned KR signal trace routing on the circuit board complies with the guidance provided in this document and the interconnect characteristics recommended in IEEE 802.3ap sections 69.3 and 69.4.
- Planned SFI signal trace routing on the circuit board complies with the guidance provided in this document and complies with the interconnect characteristics recommended in the SFF-8431 specifications. Contact your Intel sales representative to get signal integrity support.
- Planned KR4, XLAUI, XLPPI, and CR4 signal trace routing on the circuit board complies with the layout guidance provided in this document and complies with the interconnect characteristics recommended in IEEE 802.3ba. Contact your Intel sales representative to get signal integrity support.
- Made use of Intel's available impedance calculators and that results match within 5% of calculated impedance from other tools. Contact your Intel sales representative to get signal integrity support.
- For most KR and KR4 board traces, if possible, export S-parameters for the planned signal channels, and compare them to the IEEE recommended electrical characteristics. Optimize the signal path until it complies with the IEEE recommendations. IEEE channel characteristics recommendations are for the entire length of the board channels - from the solder-pads for one IC device to the solder-pads for another device, at the far-end of the entire channel path. This end-to-end signal path typically includes two or three circuit boards, connectors, and AC coupling capacitors.
- For unusual routing requirements, which make it difficult to meet the IEEE channel recommendations might still work satisfactorily with the XL710. These are channels that have been optimized by following the layout guidelines recommended within this document but which cannot be improved enough to comply with IEEE 802.3 recommended electrical characteristics. With sufficient notice, Intel engineers can provide assistance. Trace routing should be optimized prior to following steps:
- Request a layout review from your Intel sales representative (must be willing to provide board stack-up information and trace CAD artwork).
- After traces have been optimized, if the IEEE recommended electrical characteristics are still not being met, then end-to-end board channel S-parameter models should be extracted (preferably in the Touchstone* S4p format) for additional investigative simulations by Intel signal integrity engineers. Please request the required S-parameter frequency range, step size etc., before extracting Touchstone S-parameter models.
- For accurate simulation results, pre and post plating information (including surface roughness) should also be included in the extracted channel models.

15.16 General Routing Guidelines

This section is a combination of general recommendations for signal integrity that apply to all interfaces and power delivery.

15.16.1 Board Stackup

PCBs for designs typically have six, eight, or more layers. Although the XL710 does not dictate stackup, the following examples are of typical stack-up options.

1. Microstrip example:

- Layer 1 is a signal layer.
- Layer 2 is a ground layer.
- Layer 3 is used for power planes.
- Layer 4 is a signal layer. Careful routing is necessary to prevent crosstalk with layer 5.
- Layer 5 is a signal layer. Careful routing is necessary to prevent crosstalk with layer 4.
- Layer 6 is used for power planes.
- Layer 7 is a signal ground layer.
- Layer 8 is a signal layer.

Note: Layers 4 and 5 should be used mostly for low-speed signals because they are referenced to potentially noisy power planes that might also be slotted.

2. Stripline example:

- Layer 1 is a signal layer.
- Layer 2 is a ground layer.
- Layer 3 is a signal layer.
- Layer 4 is used for power planes
- Layer 5 is used for power planes
- Layer 6 is a signal layer.
- Layer 7 is a signal ground layer.
- Layer 8 is a signal layer.

Note: To avoid the effect of the potentially noisy power planes on the high-speed signals, use offset stripline topology. The dielectric distance between the power plane and signal layer should be three times the distance between ground and signal layer.

This board stack-up configuration can be adjusted to conform to your company's design rules.

15.16.2 Power Supply

As previously mentioned in Section 15.13, the 1.2V power plane needs to be split in two: a plane supplying the analog domain and one supplying the digital core.

15.16.2.1 VCCA Analog Supply

The analog domain's switching voltage regulator integrated in the XL710 also needs attention. This switching voltage regulator is being powered from the 3.3V rail that serves as the power source for other noise sensitive blocks. The layout needs to provide separation between the noise due to the input voltage ripple and the rest of the 3.3V pins. One of the possible layout topologies is listed in Figure 15-9.

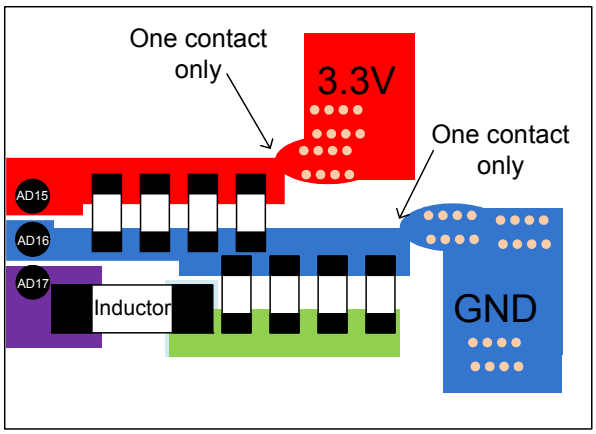


Figure 15-9. Layout Topology Example

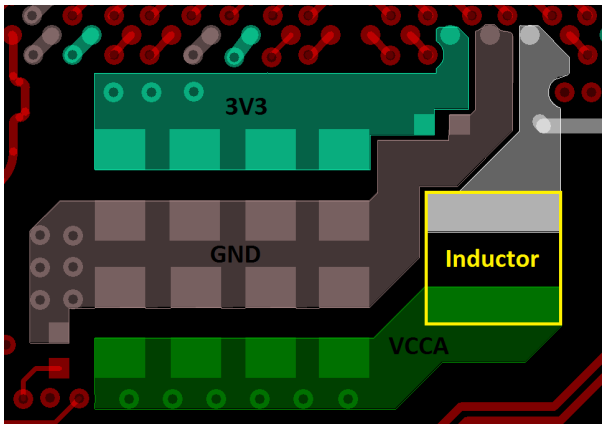


Figure 15-10. VCCA Layout Example

Note the bulk capacitance at the input of the regulator is placed close to the input pin and the rest of the 3.3V circuit is isolated from this copper island by a single contact point.

The same strategy can be used to avoid switching noise from propagating through the ground planes.

15.16.2.2 VCCD Digital Supply

The layout of the digital domain's power supply is also critical to ensure reliable operation.

The placement should focus on minimizing the high current loops in the power stage of the regulator and separating the feedback and compensation circuits from the noisy switching circuit. It is good to provide some separation between the XL710 and the inductor.

The current sense signals should be routed differentially and separated from noisy planes like the switching node to avoid potential noise coupling into the signals and causing stability issues. The RC elements of this circuit need to be placed in the close vicinity of the XL710 (within 300 mils).

Similarly, the voltage feedback signals should be routed differentially and separated from noisy nodes. The resistive feedback divider should be placed in the close vicinity of the XL710 (within 300 mils).

The gate signals should be routed with wide traces and kept far away from the sensitive analog circuits.

The high current path should be routed with wide copper fills. Where layer transitions are necessary an array of vias should be used. The number of adequate vias is a function of the via-size and the peak currents but care should be taken to minimize the swiss-cheese effect of these vias on the affected power planes.

For more details, refer to the appropriate power supply controller's datasheet and design guidelines.

15.16.3 Miscellaneous

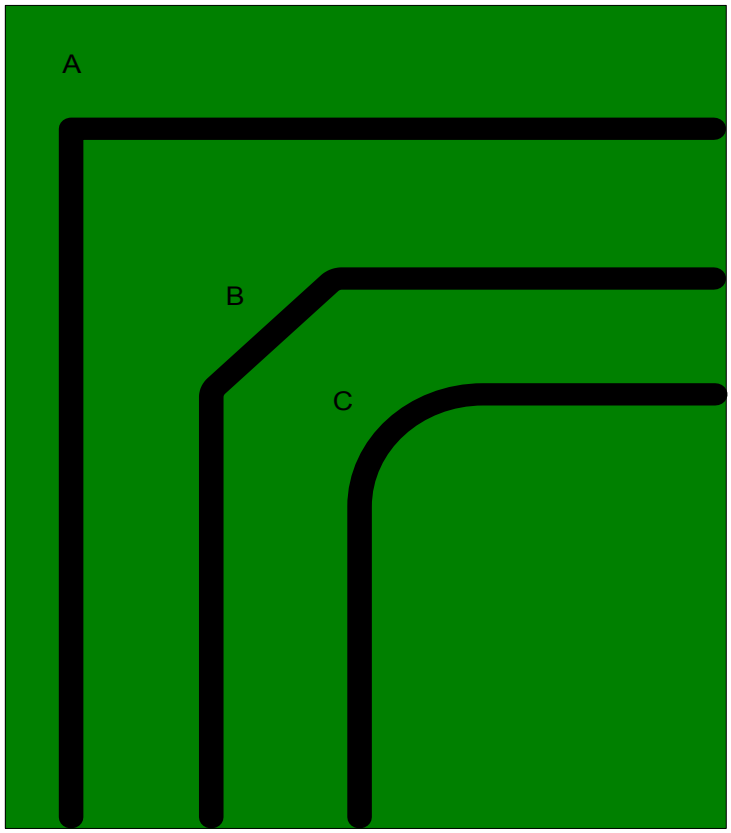
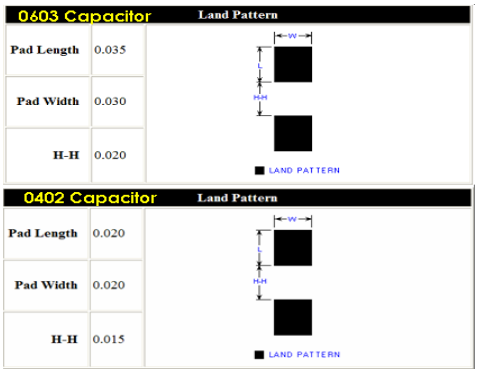
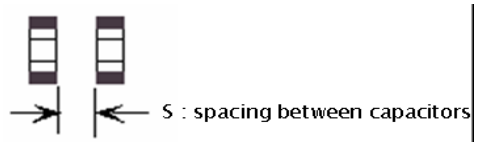
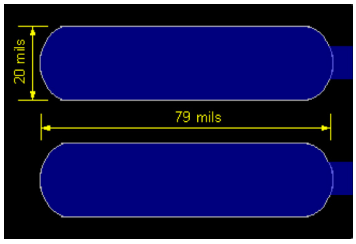
Guideline	Following are three common options for routing 90 degree corners: 1. Right angle routing with no change in trace width. This adds a small excess capacitance to the trace. It is a marginal technique and might be required in some instances such as in pin fields but options (b) or (c) are preferred for best signal integrity. 2. A slight miter or chamfer is added to reduce or eliminate the excess capacitance. This is automatically performed by most routers with simple option settings. 3. Manually routing to transact the orthogonal traces at 45° eliminates the excess capacitance and also minimizes trace length. Moreover, using round corners even further eliminates capacitance and provides shorter length. 
Applicable Buses	All.
Purpose	Reduces length mismatch and discontinuity. Also reduces possibility of acid trap.
Significance	Medium. Large numbers of bends as shown in option (a) could cause accumulated common mode noise effects. Option (b) is good for most cases and option (c) is preferred for **best** signal integrity performance.

Table 15-45. Pad Geometries

Guideline	The following geometry is for reference. Follow the DFM rule if there is any conflict. • 0603 capacitor spacing. 30 x 35 mils PAD. 25 mils spacing (S) – can be up to 30 mils if test pad is required. • 0402 capacitor. 20 x 20 mils PAD. 20 mils spacing (S) – can be up to 30 mils if test-pad is required. For high speed interfaces, it is desired to use 0402 for smaller parasitics.    High speed connector solder pads should either be rounded or chamfered at the corners to eliminate sharp corners, which can cause reflections and concentration of high frequency currents at the corners.
Applicable Buses	All.
Purpose	Provides the reference design for capacitor pad and reduces reflections from sharp corners.
Significance	Medium.
Implementation	Follow the DFM rule if there is any conflict.

15.17 Interface Specific Layout Considerations

15.17.1 NC-SI Layout Requirements

15.17.1.1 Board Impedance

The NC-SI signaling interface is a single ended signaling environment and as such Intel recommends a target board and trace impedance of 50 Ω +L20% and -10%. This impedance ensures optimal signal integrity and quality.

15.17.1.2 Trace Length Restrictions

The recommended maximum trace lengths for each circuit board application is dependent on the number drops and the total capacitive loading from all the trace segments on each NC SI signal net. The number vias must also be considered. Circuit board material variations and trace etch process variations affect the trace impedance and trace capacitance. For each fixed design, highest trace capacitance occurs when trace impedance is lowest. For the FR4 board stack-up provided in direct connect applications, the maximum length for a 50 Ω NC-SI trace would be approximately 9 inches on a -10% board impedance skew. This ensures that signal integrity and quality are preserved and enables the design to comply with NC-SI electrical requirements.

For special applications that require longer NC-SI traces, the total functional NC-SI trace length can be extended with non-compliant rise time by:

- Providing good clock and signal alignment
- Testing with the target receiver to verify it meets setup and hold requirements

For multi-drop applications, the total capacitance and the extra resistive loading affect the rise time. A multi-drop of two devices limits the total length to 8 inches. A multi-drop of four limits the total length to 6.5 inches. Capacitive loading of extra vias have a nominal effect on the total load.

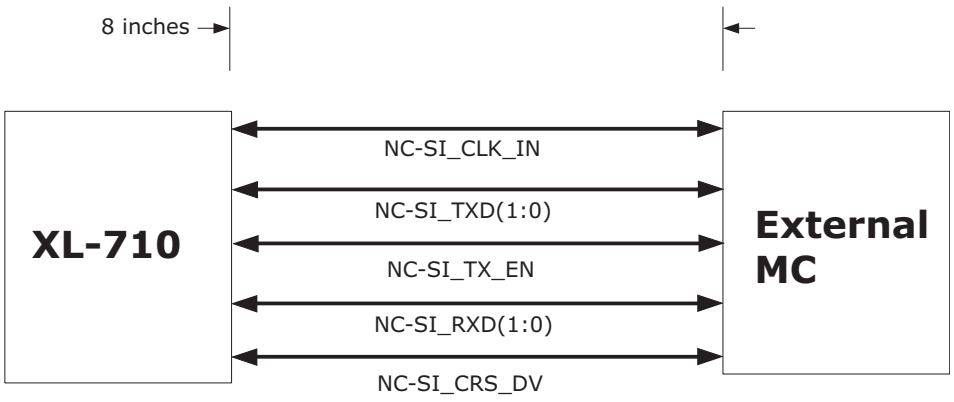


Figure 15-11. NC-SI Trace Length Requirement for Direct Connect

Table 15-46 lists how seven more vias increase the rise time by 0.5 ns. Again, longer trace lengths can be achieved.

Table 15-46. Stack Up (Seven Vias)

Item	Value	Units
Trace width	4.5	mils
Trace thickness	1.9	mils
Dielectric thickness	3.0	mils
Dielectric constant	4.1	--
Loss tangent	0.024	--
Nominal impedance	50	W
Trace capacitance	1.39	pf/inch

Table 15-47 lists the example trace lengths for the multi-drop topology of 2 and 4 shown in Figure 15-4 and Figure 15-5.

Table 15-47. Example Trace Lengths for Multi-Drop Topologies (2 and 4)

Multi-drop Length Parameter Used in Figure 15-12 and Figure 15-13	Segment Length Example Eor Multi-drop Configurations			
	2-drop Configuration		4-drop Configuration	
	Length (Inches)	Trace capacitance (Pf)	Length (Inches)	Trace capacitance (Pf)
L1	2	2.8	1.5	8.8
L2	4	5.6	2	16.1
L3	2	2.8	1	8.1
Total	8	11.1	6.5	35.6

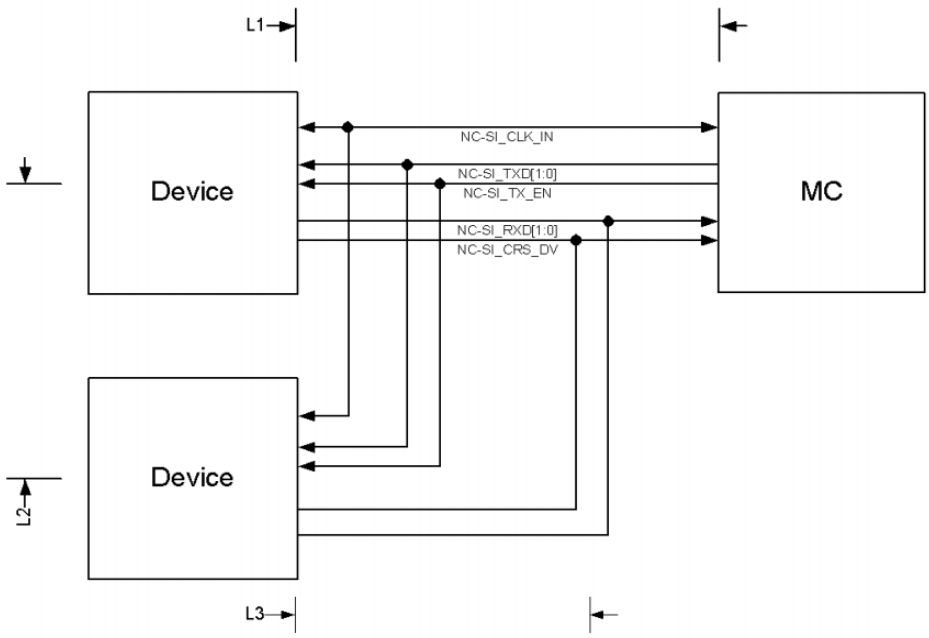


Figure 15-12. 2-drop Topology Example

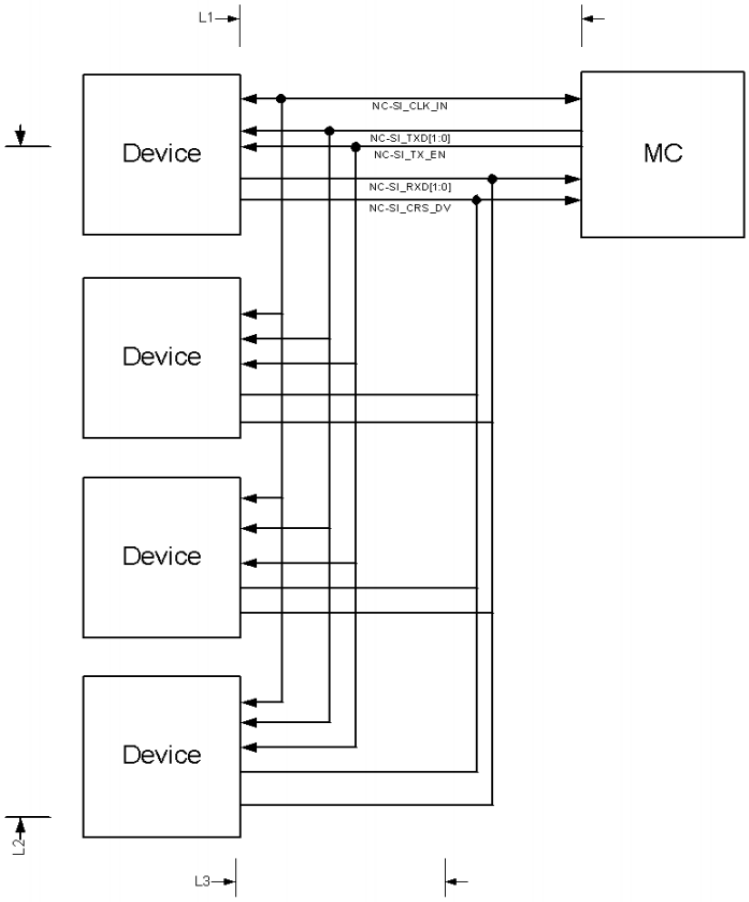


Figure 15-13. 4-drop Topology Example

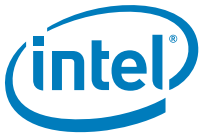


Table 15-48. Compliant NC-SI Maximum Length on a 50 Ω -10% Skew-board with Example Stack-up

Topology	Total Maximum Compliant Linear Bus Size (Inches)	Number of Vias	Approximate Net Trace Capacitance Minus Load Capacitance (pf)
4 multi-drop	6.0	1	8.3
4 multi-drop	5.5	8	8.3
2 multi-drop	8.0	1	11.1
2 multi-drop	7.5	8	11.1
Point to point	9.0	1	12.5
Point to point	8.5	8	12.5

Extending NC-SI to a maximum 11 ns rise time increases the maximum trace length.

Table 15-49. Functional NC-SI Maximum Length on a 50 Ω -10% Skew Board with Example Stack-up (based on actual lab-measured solution)

Topology	Total Maximum Functional Linear Bus Size (Inches)	Number of Vias	Approximate Net Trace Capacitance Minus Load Capacitance (pf)
4 multi-drop	19	1	26.4
4 multi-drop	18	8	26.4
2 multi-drop	20	1	27.8
2 multi-drop	19	8	27.8
Point to point	22	1	30.6
Point to point	21	8	30.6

15.17.2 SFP+ Layout Recommendations

To meet strick TWDPc and DDJ electrical requirements, designers need to optimize SFI trace routing accordingly.

In order to determine the maximum trace lengths allowed for SFP+ in a design refer to the Intel® XL710 SFP+ Differential Trace Calculator and the Intel® XL710 SFI/SFP+ TX HSPICE Channel Checker Simulation Kit. Table 15-50 lists some microstrip example trace geometries and lengths the XL710 supports.

Table 15-50. SFP+ Trace Geometries

Trace Type	Dielectric Material	Dielectric Constant (dk or Er) 2.5 GHz to 5 GHz	Dissipation Factor (df or Loss Tangent)	Dielectric Layer Thickness (or Height) (mm)	Copper Trace Thickness After Plating[1] (mm)	Max SFI TX/RX Trace Length (mm)	Main SFI Routing Trace Width (mm)	Main SFI Routing In-Pair Trace Separation edge to edge (mm)	SFI Breakout Routing Trace Width[2] (mm)	SFI Breakout Routing In-Pair Trace Separation Edge-to-Edge (mm) [2]
Microstrip	Nelco N4000-13 (2-ply 2113)	3.7	0.009	0.1905 (7.0 mils)	0.0508 (2.0 mils)	296.57 (11.676 in)	0.2921 (11.5 mils)	0.4064 (16 mils)	0.1778 (7 mils)	0.127 (5.0 mils)
Microstrip	Nelco N4000-13 (2-ply 2113)	3.86	0.009	0.1524 (6.0 mils)	0.0508 (2.0 mils)	269.69 (10.618 in)	0.2286 (9.5 mils)	0.3683 (14.5 mils)	0.1880[3] (7.4 mils)	0.1905 (7.5 mils)
Microstrip	Panasonic Megtron6 (2-ply 3113)	3.63	0.003	0.2032 (8.0 mils)	0.0508 (2.0 mils)	461.39 (18.165 in)	0.3302 (13 mils)	0.4064 (16 mils)	0.2159 (8.5 mils)	0.1524 (6 mils)
Microstrip	Panasonic Megtron6 (2-ply 1080)	3.4	0.003	0.1524 (6.0 mils)	0.0508 (2.0 mils)	372.71 (14.674 in)	0.2667 (10.5 mils)	0.4572 (18 mils)	0.2032 (8 mils)	0.1905 (7.5 mils)
Microstrip	Isola FR408	3.63	0.013	0.1534 (6.04 mils)	0.0508 (2.0 mils)	233.12 (9.178 in)	0.2540 (10 mils)	0.4318 (17 mils)	0.1905 (7.5 mils)	0.1854 (7.3 mils)
Microstrip	Isola FR406	3.76	0.0186	0.1839 (7.24 mils)	0.0508 (2.0 mils)	200.07 (7.877 in)	0.2921 (11.5 mils)	0.4064 (16.0 mils)	0.1778 (7 mils)	0.1448 (5.7 mils)
Microstrip	Isola FR406	3.76	0.0186	0.1636 (6.44 mils)	0.0508 (2.0 mils)	188.21 (7.410 in)	0.2540 (10.2 mils)	0.3683 (14.5 mils)	0.1905 (7.5 mils)	0.1803 (7.1 mils)
Microstrip	FR4 2116, 2-ply	4.24[3]	0.024[4]	0.2286[4] (9.0 mils)	0.0508 (2.0 mils)	171.34 (6.746 in)	0.3175 (12.5 mils)	0.3759 (14.8 mils)	0.1778 (7.0 mils)	0.1422 (5.6 mils)
Microstrip	FR4 2116, 2-ply	4.24[4]	0.024[4]	0.2286[4] (9.0 mils)	0.04572 (1.8 mils)	171.39 (6.748 in)	0.3200 (12.6 mils)	0.3683 (14.5 mils)	0.1778 (7.0 mils)	0.1372 (5.4 mils)
Microstrip	FR4 2113, 2-ply	4.0	0.021	0.2032 (8.0 mils)	0.04826 (1.9 mils)	53.416 (7.202 in)	0.292 (11.5 mils)	0.356 (14.0 mils)	0.1674 (6.6 mils)	0.1371 (5.4 mils)
Microstrip	FR4 2113, 2-ply	4.05	0.021	0.2032 (8.0 mils)	0.04572 (1.8 mils)	56.718 (7.424 in)	0.305 (12.2 mils)	0.419 (16.5 mils)	0.1701 (6.7 mils)	0.1397 (5.5 mils)
Microstrip	FR4 2113, 2 Ply	4.0	0.021	0.1905 (7.5 mils)	0.04826 (1.9 mils)	53.416 (6.905 in)	0.2667 (10.5 mils)	0.2921 (11.5 mils)	0.2032 (8.0 mils)	0.1524 (6.0 mils)

1. Post-plating copper thickness tolerance ±0.3 mils (0.00762 mm).
2. For SFI traces that are 9 mils wide or less with 12 mils separation or less: Narrow breakout trace widths with smaller in-pair trace separation distances are discouraged. **Narrow SFI traces and/or less in-pair separation should NOT be required when the main trace route is 9 mils wide or less. This is especially true for the SFI Tx trace routes**
3. 2116 FR4 manufactured by different vendors might have different dielectric constants, dissipation factors, different dielectric thicknesses (height), or a combination of these. Check with the appropriate circuit board supplier to obtain the correct properties (at 5 GHz) for the 2116 FR4 that is planned for use. If the 2116 FR4 has different electrical properties or different nominal thickness, the trace widths and trace separation might need to be adjusted.

For the breakout region of the XL710, reducing the differential trace widths is allowed only if the main of the SFI routing trace width used is greater than 9 mils wide AND 100 Ω differential impedance is maintained in the breakout. If the main SFI trace routing uses 9 mils or smaller trace widths, then reducing SFI trace widths in the breakout region is not encouraged or even necessary. Table 15-43 lists recommended breakout region SFI trace and space dimensions for several different dielectric materials. In addition, when implementing narrow breakouts, keep the length of the breakout as short as possible (limiting the length of this narrow section to 100 mils for the SFI Tx pairs and 180 mils for Rx pairs). Figure 15-14 shows two narrow breakout examples and Figure 15-15 shows a Tx/Rx breakout example with 9 mil trace widths. If narrow or a thinner trace breakout is used, keep the narrow traces in-pair skew as low as possible using length matching techniques, this section is more susceptible to signal skew effects than the main SFI route.

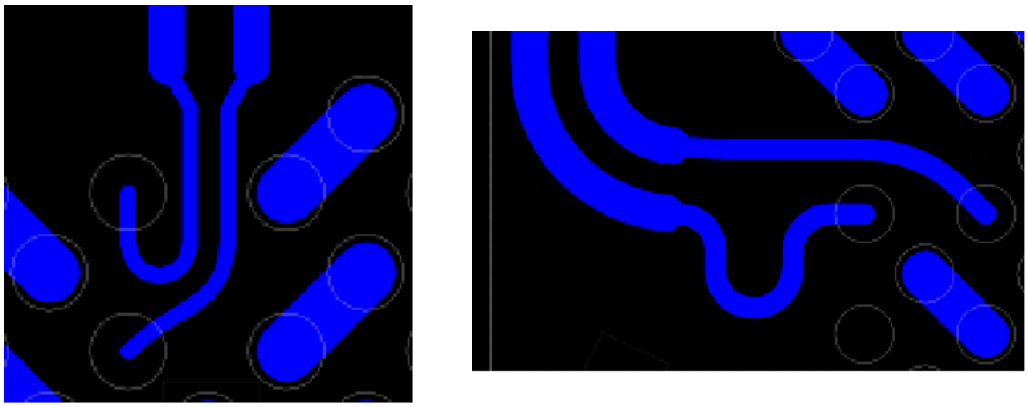


Figure 15-14. Narrow Breakout Routing

Note: Figure 15-14 shows thinner than recommended differential traces used for breakout. This routing is still allowed for this section. Note the breakout length matching.

Figure 15-14 left: Both traces are kept coupled with differential impedance around 100 Ω. This is only achievable in certain cases. Figure 15-14 right: Breakout traces are separated, allowing for wider traces, and each single trace's impedance is kept around 50 Ω.

Added trace loops for in-pair length matching should be located as close as possible to the XL710's BGA solder pads. The space is very limited inside the breakout region so these loops can usually be made just outside the breakout section (see Figure 15-15).

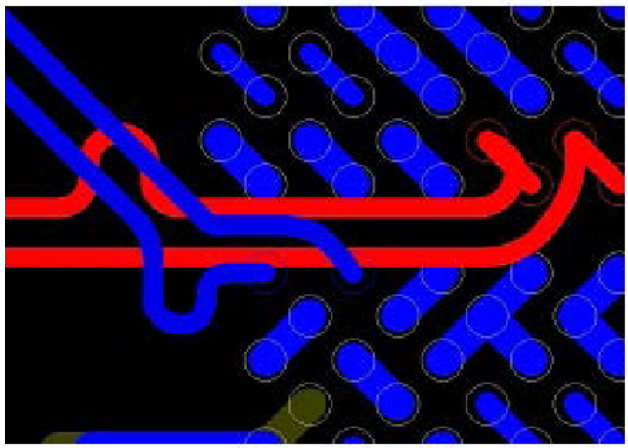


Figure 15-15. 9 Mil Wide Main Traces and Breakout

Note: There is no need for narrower breakout traces. Notice the added length matching trace loop is close to the breakout section in Figure 15-15. Red traces are on the bottom layer. The 82599 is on the top.

Intel strongly recommends against using serpentine traces to match the trace lengths within SFI signal pairs. With serpentines, it is much too easy to accidentally cause unintended signal skew and to increase the signal dispersion (undesirable pulse width and pulse edges spreading). Serpentines can be acceptable for PCIe in-pair trace length matching, if the serpentines are done correctly.

As shown in Figure 15-16, the trace lengths directly depend on the chosen lanes, placement of the XL710 relative to the SFP+ cages and the placement of the SFP+ cages relative to each other. The placement can be defined by three variables as follows:

- Cage Gap—The distance between the cages
- BGA Offset X—Refers to the distance between the cages and the edge of the BGA package
- BGA Offset Y—Refers to the offset of the BGA package relative to the first cage

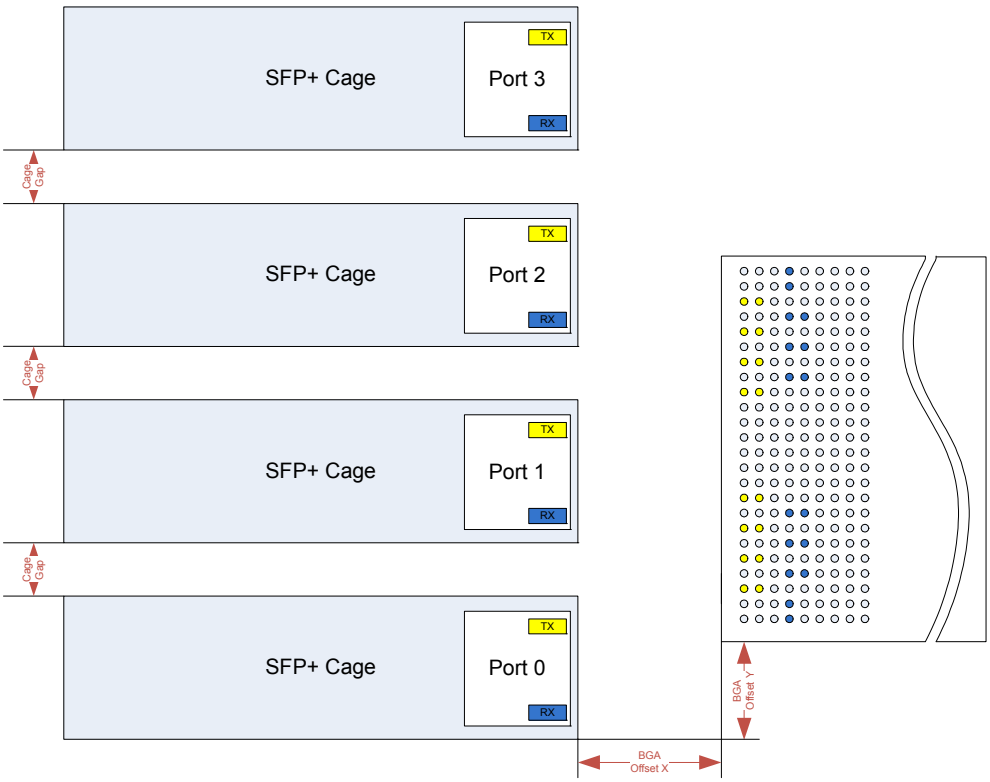
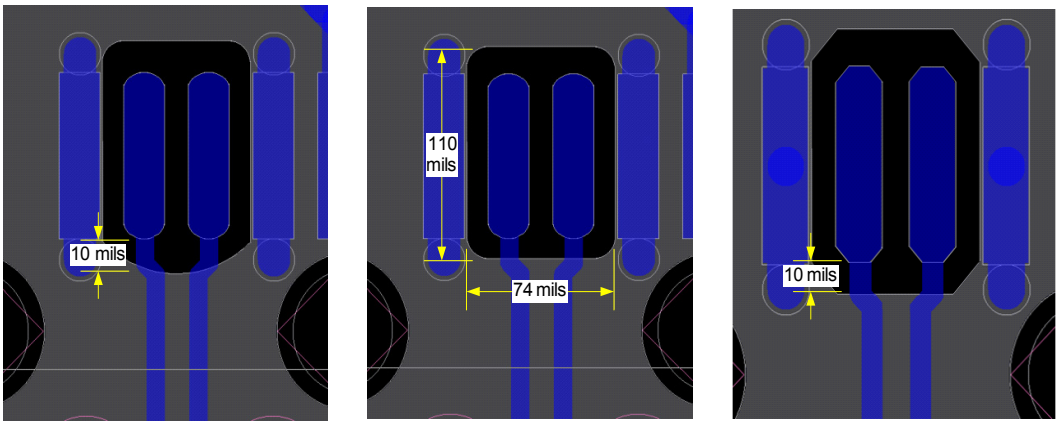


Figure 15-16. Component Placement

Table 15-51. SFP+ Connector Pad Voids

Guideline	Make the voided areas under the connector pads rectangular-like in shape (chamfer or round the corners), and a little larger than the area occupied by each pair of SFP module connector pads. Keep the void width to 20 mils larger than the area occupied by the connector's signal pin solder-pads. The void dimensions should be approximately 110 mils long by 79 mils wide. • Void similar-shaped areas of any inner Vcc power-plane that are under the signal pins or solder-pads of the SFI connector. The voids in power planes should be a few mils larger than the voids in the ground plane on the second layer. The reason for increasing the void dimensions on power planes layers is to prevent power supply noise from coupling onto the SFI signals. • Start plane voids ~10 mils before the SFI traces reach the connector solder-pad. The last 10 mils of the SFI traces should be over the reference plane void. If the void in the planes starts closer to the solder-pads, it causes fringe capacitance, which lowers the solder-pads' effective impedance too much. • Round or chamfer corners of the plane layer (this is especially important at the end of the voids where the SFI traces come in) to reduce possible reflections and EMI issues. If the plane void has sharp corners on the end where the SFI traces enter, it can cause high frequency currents to concentrate at those corners, and can potentially cause reflections and/or radiated EMI. 
Applicable Buses	SFI.
Purpose	Because the SFP+ module connector pads at the end of the SFI traces are wider than the SFI traces, the reference plane area directly under each pair of SFP module signal pin connector solder-pads must be voided in order to avoid excessive capacitance.
Significance	Low.

15.17.3 QSFP+ Connector Layout Recommendations

The same rules that apply to SFP+ apply to QSFP+. However, due to the routing density and different connector dimensions additional special rules need to be followed.

Table 15-52. QSFP+ Connector Pad Voids

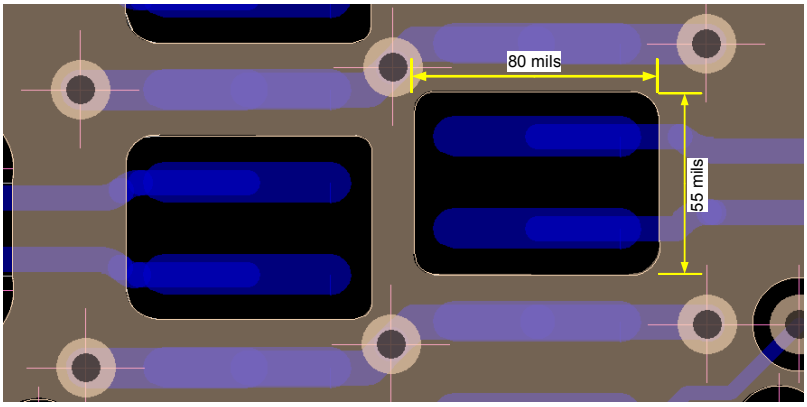
Guideline	Make the voided areas under the connector pads rectangular-like in shape (chamfer or round the corners), and a little larger than the area occupied by each pair of SFP module connector pads. Keep the void width to 10 mils larger than the width occupied by the connector's signal pin solder-pads. The void dimensions should be approximately 80 mils long by 55 mils wide. • Void similar-shaped areas of any inner VCC power-plane that are under the signal pins or solder-pads of the SFI connector. The voids in power planes should be a few mils larger than the voids in the ground plane on the second layer. The reason for increasing the void dimensions on power planes layers is to prevent power supply noise from coupling onto the SFI signals. • Make sure no other signal traces cross under the void. • Start plane voids >5 mils before the SFI traces reach the connector solder-pad. The last 5 mils of the SFI traces should be over the reference plane void. If the plane void starts any closer to the solder-pads, it causes fringe capacitance, which lowers the solder-pads' effective impedance too much. • Round or chamfer corners of the plane layer (this is especially important at the end of the voids where the SFI traces come in) to reduce possible reflections and EMI issues. If the plane void has sharp corners on the end where the SFI traces enter, it can cause high frequency currents to concentrate at those corners, and can potentially cause reflections and/or radiated EMI. 
Applicable Buses	SFI, XLPPI and CR4.
Purpose	QSFP+ module connector pads at the end of the SFI traces are wider than the signal traces, the reference plane area directly under each pair of SFP module signal pin connector solder-pads must be voided in order to avoid excessive capacitance.
Significance	Low to medium.

Table 15-53. QSFP+ Ground Connector Pads

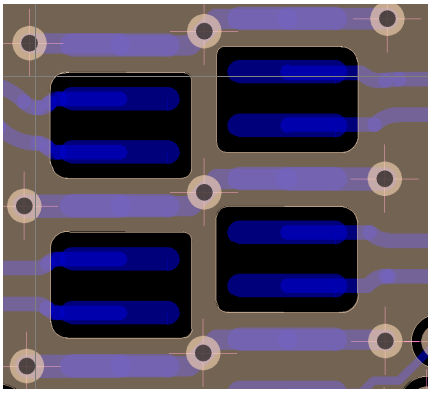
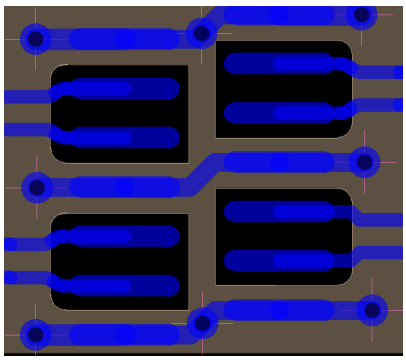
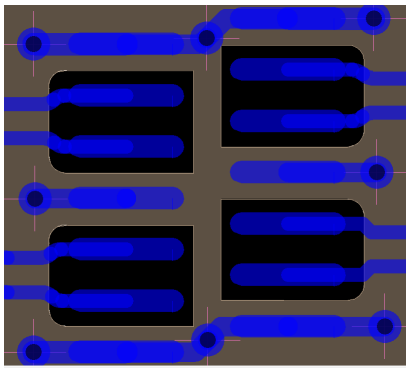
Guideline	The QSFP+ connectors' SMT ground-pin solder pads are usually connected to ground through one single ground via at one end of the solder pad. Simulations have shown that adding another ground via at the opposite end of each ground-pin's solder-pad can prevent undesirable resonances at 15 GHz (inside the working frequency band for most 10 GHz signals). Therefore, adding another ground via is highly recommended to improve signal integrity. In addition, keep the ground vias less than 20 mils from the ends of the ground pads.  **Best** If the center ground via can't be placed, do not connect ground pads together as this might unintentionally cause other frequency resonances inside the signal working frequency band. **Bad**  **Good**  If needed, ground solder-pad vias can be used as well as return-path vias for signal vias.
Applicable Buses	SFI, XLPPI and CR4.
Purpose	Eliminate QSFP+ connector ground solder-pad stubs generated when connecting only one end of the pad.
Significance	Low to medium.

Figure 15-17 shows a reference QSFP+ routing layout example for one port of the XL-710 (port configuration ID 5.01 or XLPPI; same rules apply for SFI QSFP+ configurations).

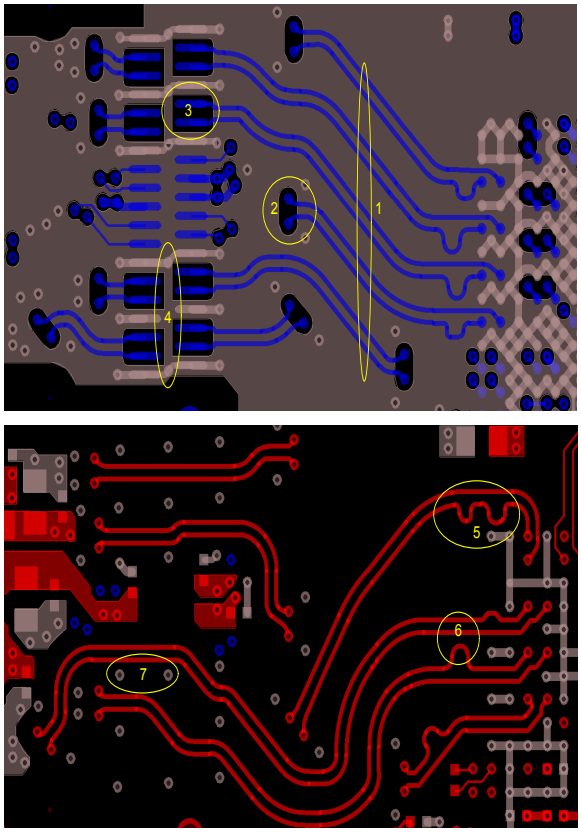


Figure 15-17. Top and Bottom Layers Example for XL-710 QSFP+ Layout

Highlighted sections in Figure 15-17:

1. Distance of 7Xh is kept throughout the route (h being adjacent dielectric height).
2. 100 Ω via transitions. Refer to Table 15-18 for more details.
3. Proper voiding of the plane under connector signal pads is done.
4. Connector ground pads are grounded at both ends of the pad. One of the ground vias is not possible to place on one end of the pad, due to other layer routing, hence no connection between pads is done.
5. Although more than one loop is not recommended, in this case it was needed in order to length match the segment between the pins and via transition.
6. Trace-to-trace spacing can be violated in the breakout regions for very short traces as shown in Figure 15-17.
7. Via-to-signal trace spacing is kept through all designs at 2.5Xh or greater.

15.17.4 KR Re-routing Layout Example

In some specific/custom use cases, there might be a desire to route the Ethernet interfaces to multiple locations on the board. Re-routing high speed serial signals requires careful design. One possible solution that a designer can explore is using AC coupling capacitors to implement the stuffing options. Placed carefully such that two capacitors share one pad that has a via in it, a designer can build a jumper structure that could enable re-routing of the high speed serial signals. See Figure 15-18 for an example.

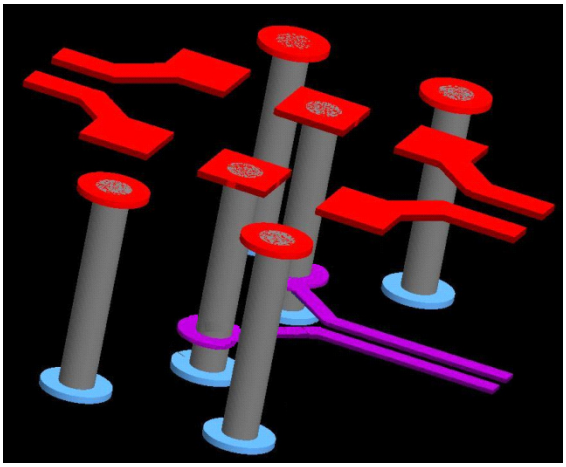
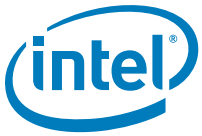


Figure 15-18. Routing High-speed Serial Signals

50 Ω single-ended impedance is necessary for transparent layer transitions from the signal's perspective. To achieve this, designers need to carefully place the return vias and might also need to void the reference plane around the signal vias and underneath the capacitor.

Please use 3D field-solver simulations to determine the appropriate structure for specific stack-ups.

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



16.0 Thermal design considerations

16.1 Introduction

This can be used as an aid when designing a thermal solution for systems implementing the XL710.

Properly designed solutions provide adequate cooling to maintain the XL710 product case temperature Tcase (or junction) at or below thermal specifications. The XL710 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 XL710. Operation outside the functional limits can cause data corruption or permanent damage to the XL710.

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.

16.2 Measuring the thermal conditions

This section provides a method for determining the operating temperature of the XL710 in a specific system based on case temperature. Case temperature is a function of the local ambient and internal temperatures of the XL710. This document specifies a maximum allowable Tcase for the XL710.

16.3 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 may 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 of 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.

16.4 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. Also note that sustained operation at component maximum temperature limit might affect long-term device reliability.

16.5 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.
FCmBGA	Molded 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 package is covered with mold to strengthen it and to protect the capacitors. The die is exposed for better thermal contact.
Junction	Refers to a P-N junction on the silicon. In this document, it is used as a temperature reference point (for example, Θ_{JA} 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).
TA	Local ambient temperature
TJ	Junction temperature: maximum temperature of die active surface, i.e. hot spot.
TC	Case temperature: temperature at geometric center of dies or over mold top surface.

Item	Description
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_{JA} (Theta JA)	Thermal resistance junction-to-ambient, °C/W.
Ψ_{JC} (Psi JT)	Junction to case (top of package) thermal characteristic parameter, defined by (TJ – TC) / TDP. Ψ_{JT} does not represent thermal resistance, but instead is a characteristic parameter that can be used to convert between Tj and Tcase when knowing the total TDP. This parameter can vary by environment conditions like heat sink and airflow.

16.6 Thermal specifications

To ensure proper operation of the XL710, the thermal solution must maintain a case temperature at or below the 110 °C. System-level or component-level thermal enhancements are required to dissipate the generated heat to ensure the case temperature never exceeds that maximum temperature.

Good heat sink and good system airflow is critical to dissipate the XL710's high power. The size and number of fans, vents, and/or ducts, and, their placement in relation to components and airflow channels within the system determine airflow. Good Thermal Interfaces Material (TIM) between the heat sink and die should be applied correctly.

To develop a reliable, cost-effective thermal solution, all of the system variables must be considered. Use system-level thermal characteristics and simulations to account for individual component thermal requirements.

Thermal diodes are available in the XL710 to measure the junction temperature.

Keep the following in mind when reviewing the data that is included in this datasheet:

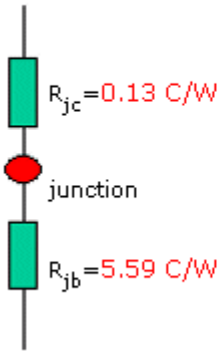
- All data is preliminary and is not validated against physical samples.
- Your system design might be significantly different.
- A larger board with more layers might improve the XL710's thermal performance.

Table 16-1. XL710 thermal specifications

Parameter	Specification	Notes
$T_{J\text{-}MAX}$	110 °C	
$T_{C\text{-}MAX}$	109 °C	At center of die top.
$T_{C\text{-}MIN}$	0 °C	
Ψ_{JC}	0.1 °C/W	Junction to case (top of die) thermal characteristic parameter.
TDP	10 W	

16.6.1 2R model parameters

Case ID: 1274



Board setup parameters

Optimize PCB :	Yes	
Trace Layer :	0.070	20 %
	0.035	90 %
	0.035	90 %
	0.070	20 %
Board Type :	High Conductivity	
Via Number:	24 X 24	
Via Pitch :	1 (mm)	
Via Diameter :	0.4 (mm)	
Via Plate Thickness :	0.05 (mm)	
Via Modeling :	Lumped	

Board Type: High Conductivity (2S2P)

Trace Layers	Thickness(mm)	Coverage(%)
Top Layer	0.070	20
Solid Plane 1	0.035	90
Solid Plane 2	0.035	90
Bottom Layer	0.070	20

16.6.2 Delphi* model parameters

	Top Inner	Bottom Inner	Top Outer	Bottom Outer
Junction	1.05	1.06	99946.47	31.20
Top Inner		1.00	20.05	54.82
Bottom Inner			INFINITY	13.29
Top Outer				2.40

Case ID : 1282
Optimization Method : DOTCOMP
Area Optimization : Enabled
Weight Factor : 0.5
Optimized Top inner size : 8.868 X 8.868 (mm)
Optimized Bottom inner size : 7.831 X 7.831 (mm)
BC file : Default_44.bc

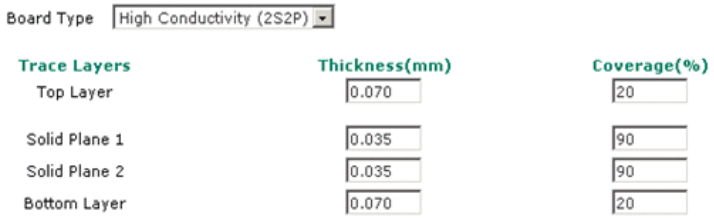
Node-to-BC column mapping:

Node	TI	BI	TO	BO
Column #	1	2	1	2

16.6.3 Still air JEDEC environment

Package Thermal Characteristics in Standard Still Air JEDEC Environment for Reference	
Package	Θ_{JA} (°C/W)
No HS	14.8
With HS 9.5 mm	9.7
With HS 15 mm	8.5

The thermal parameters previously defined are based on simulated results of packages assembled on standard multilayer 2s2p 1.0-oz Cu layer boards in a natural convection environment (still air).



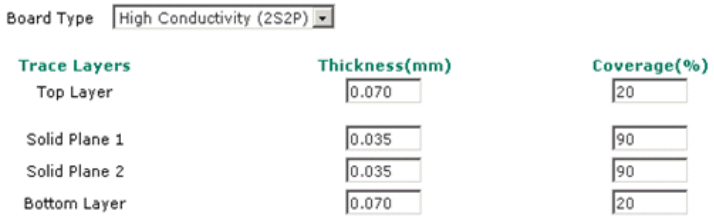
Θ_{JA} is the thermal resistance junction-to-ambient of the package.

The XL710 cannot operate properly without a heat sink or any other good cooling method.

16.6.4 Package thermal characteristics with forced air JEDEC environment (example 1)

The thermal graphs and parameters that follow are based on simulated results of packages assembled on standard multilayer 2s2p 1.0-oz Cu layer boards in a Forced Air JEDEC chamber. A system with the following attributes was used to generate thermal characteristics data:

- Standard JEDEC forced air environment
- Aluminum heat sink
  - 40 x 40 x 2.5 mm base
  - 13 fins Z = 9.9 mm W = 1.1mm
- TIM PCM45, 40u bond line
- 76 mm x 114 mm PCB



The graphs that follow can be used as an aid in determining the optimum airflow and heat sink combination for the XL710.

Again, your system design might vary considerably from the typical system board environment used to generate these figures.

Note: Thermal models are available upon request (Flotherm*: 2-Resistor, Delphi, or detailed). Contact your local Intel sales representative for the XL710 thermal models

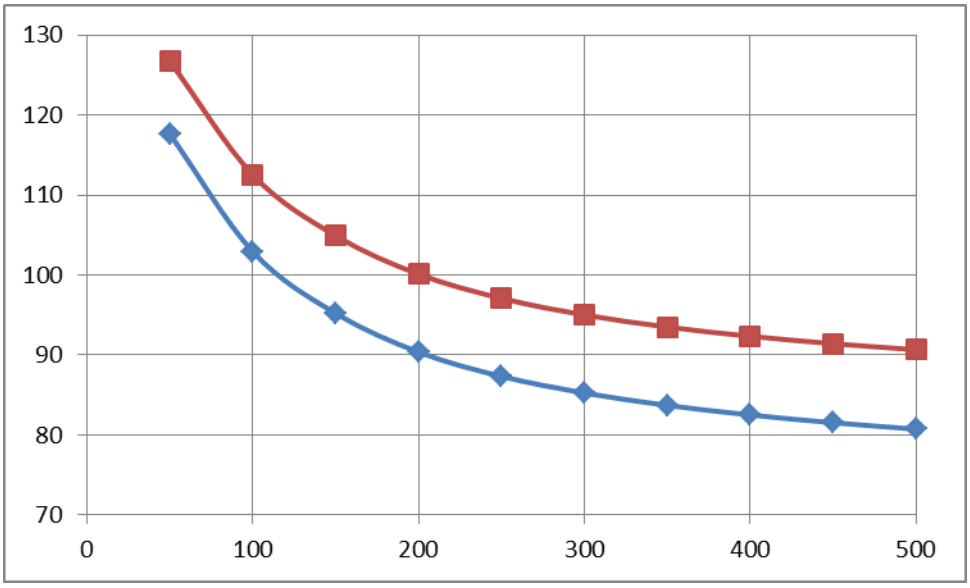


Figure 16-1. XL710 temp vs. air flow @ 10 W (JEDEC card) with 12.4 mm HS

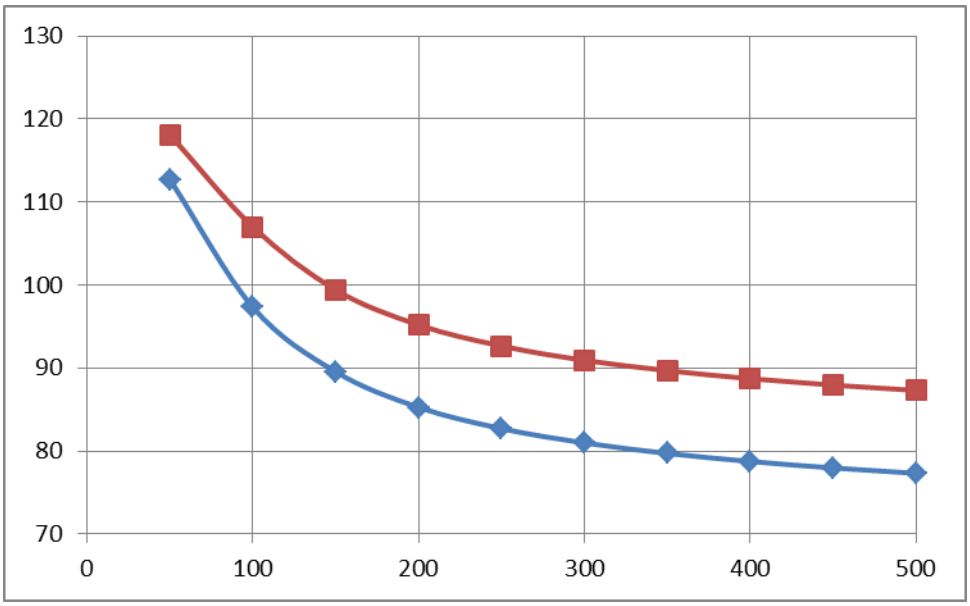


Figure 16-2. XL710 temp vs. air flow @ 10 W (JEDEC card) with 15 mm HS

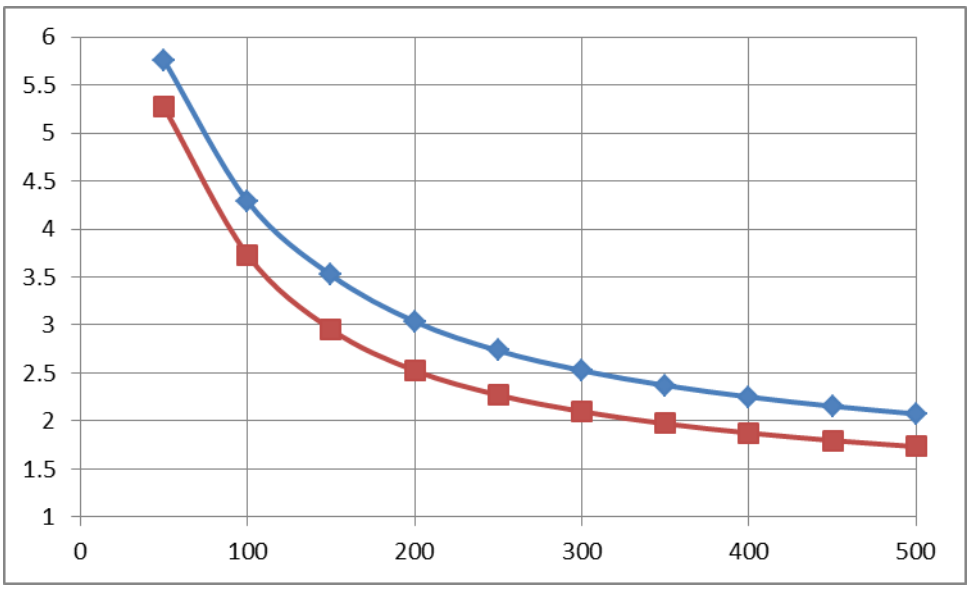
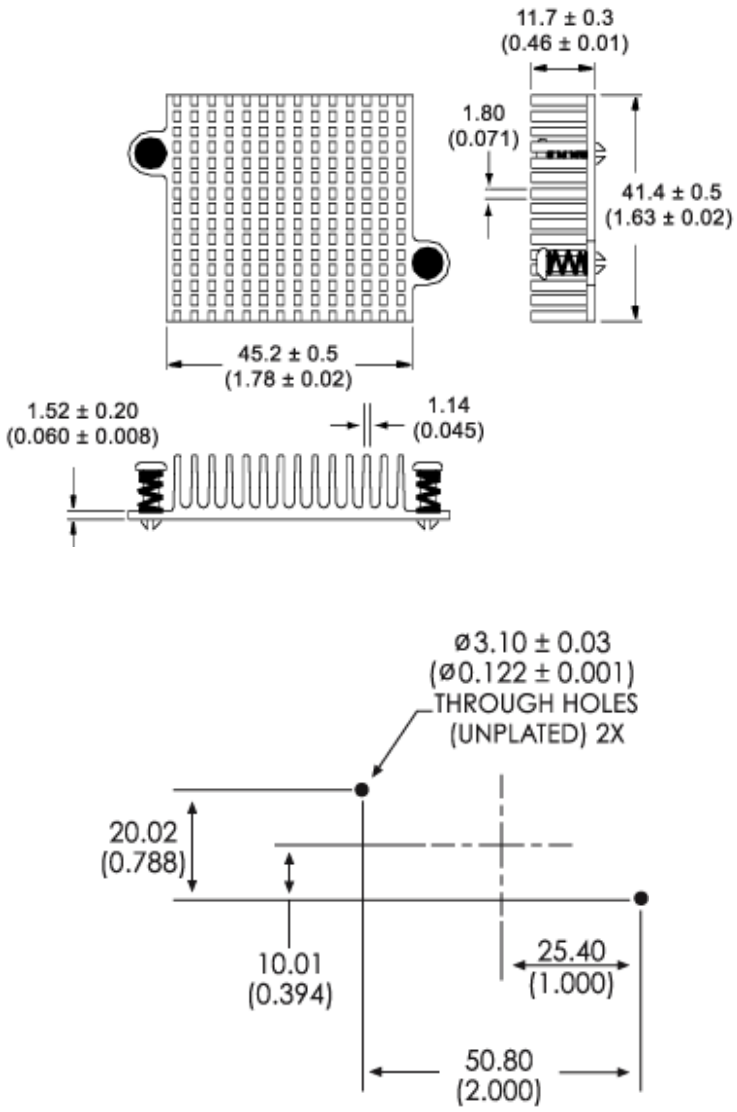


Figure 16-3. XL710 Θ_{JA} (°C/W) vs. air flow @ 10 W (JEDEC card)

16.6.5 Package thermal characteristics with forced air JEDEC environment (example 2)

The thermal graphs and parameters that follow are based on simulated results of packages assembled on standard multilayer 2s2p 1.0-oz Cu layer boards in a forced air JEDEC chamber. A system with the following 10-L4LB-11G heat sink was used to generate thermal characteristics data:

- 10-L4LB-11G heat sink from Aavid Thermalloy



Use the following tables as an aid in determining the optimum airflow and heat sink combination for the XL710.

Again, your system design might vary considerably from the typical system board environment used to generate these figures.

Thermal models are available upon request (Flotherm*: 2-Resistor, Delphi, or detailed). Contact your local Intel sales representative for the XL710 thermal models.

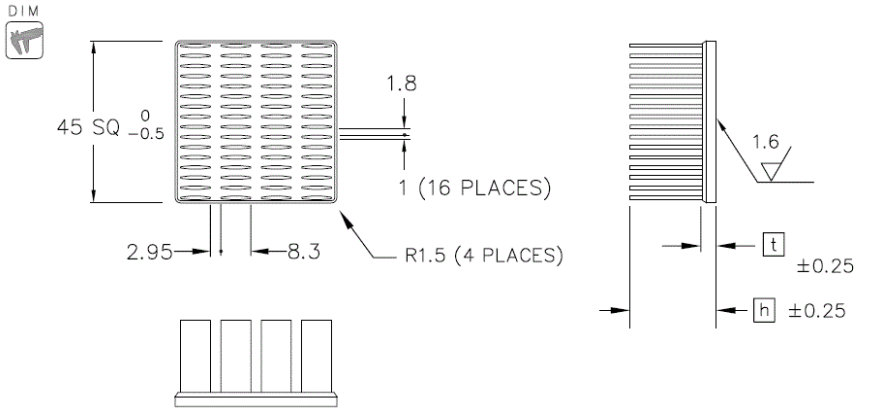
8.6W TDP

Ambient (°C)	Airflow (LFM)								
	0	50	100	150	200	250	300	350	400
45	94	89.61	82.24	76.44	75.52	72.73	70.63	69.08	67.9
50	98.41	94.2	86.99	79.48	80.39	77.65	75.55	74.02	72.86
55	102.9	98.78	91.74	84.36	85.27	82.57	80.5	78.97	77.81
60	107.5	103.4	96.48	89.23	90.14	87.49	85.43	83.92	82.77
65	112	108	101.3	94.13	65.03	92.37	90.34	88.84	87.7
70	116.6	112.6	106	98.99	99.99	97.29	95.28	93.79	92.66
75	121.1	117.1	110.8	103.9	104.8	102.2	100.2	98.74	97.62

10W TDP

Ambient (°C)	Airflow (LFM)								
	0	50	100	150	200	250	300	350	400
45	100.9	96.3	88.13	79.37	80.43	77.24	74.79	73	71.63
50	105.4	100.8	92.84	84.22	85.29	82.15	79.72	77.93	76.58
55	109.9	105.4	97.55	89.07	90.14	87.06	84.64	82.88	81.53
60	114.4	109.9	102.3	93.92	95	91.96	89.58	87.83	86.49
65	118.8	114.5	107	98.8	99.86	96.82	94.47	92.73	91.41
70	123.3	119	111.7	103.6	104.7	101.7	99.39	97.68	96.37
75	127.8	123.5	116.4	108.5	109.6	106.6	104.3	102.6	101.3

16.6.6 Package thermal characteristics with forced air JEDEC environment (example 3)



HEAT SINK
MATERIAL : DATA6063
FINISH : CLEAR ANODIZE

MODEL	HEIGHT [h]	THICKNESS [t]	WEIGHT (grams)
W45−10W	10	2	19.5
W45−15W	15	2	25.0
W45−20W	20	2.5	33.0
W45−25W	25	2.5	38.5

Dimensions : mm

Use the following tables as an aid in determining the optimum airflow and heat sink combination for the XL710.

Again, your system design might vary considerably from the typical system board environment used to generate these figures.

Note: Thermal models are available upon request (Flotherm*: 2-Resistor, Delphi, or detailed). Contact your local Intel sales representative for the XL710 thermal models

8.6W TDP

Ambient (°C) \ Airflow (LFM)	0	50	100	150	200	250	300	350	400
45	99.88	91.86	84.79	79.73	77.37	74.63	72.61	70.91	69.39
50	104.2	96.57	89.53	84.55	82.2	79.52	77.52	75.73	74.36
55	108.5	100.4	94.24	89.4	87.09	84.42	82.44	80.7	79.31
60	112.8	105	98.94	94.252	91.92	89.31	87.34	85.62	84.25
65	117.2	109.8	103.7	99.06	96.81	94.21	92.28	90.56	89.21
70	121.6	114.7	108.4	103.9	101.6	99.09	97.2	95.6	94.24
75	125.9	119.1	113	108.7	106.5	104	102.1	100.5	99.1

10W TDP

Ambient (°C)	Airflow (LFM)								
	0	50	100	150	200	250	300	350	400
45	107.5	99.41	91.09	85.26	82.51	79.43	76.95	74.95	73.33
50	111.8	103.7	95.77	90.06	87.36	84.3	81.87	79.88	78.27
55	116	107.5	100.5	94.87	92.22	89.15	86.83	84.81	82.23
60	120.3	112.3	105	99.24	96.98	94.13	91.68	89.74	88.18
65	124.6	117	109.7	104.5	101.9	98.91	96.65	94.66	93.09
70	128.9	121.1	114.5	109.4	106.7	103.8	101.5	99.59	98.05
75	133.2	126.1	119.2	114.1	111.6	108.7	106.5	104.5	103

16.7 Package mechanical attributes

The XL710 is packaged in a 25 mm FCmBGA, illustration as shown.

Note: Make sure official drawings are used for your detailed design.

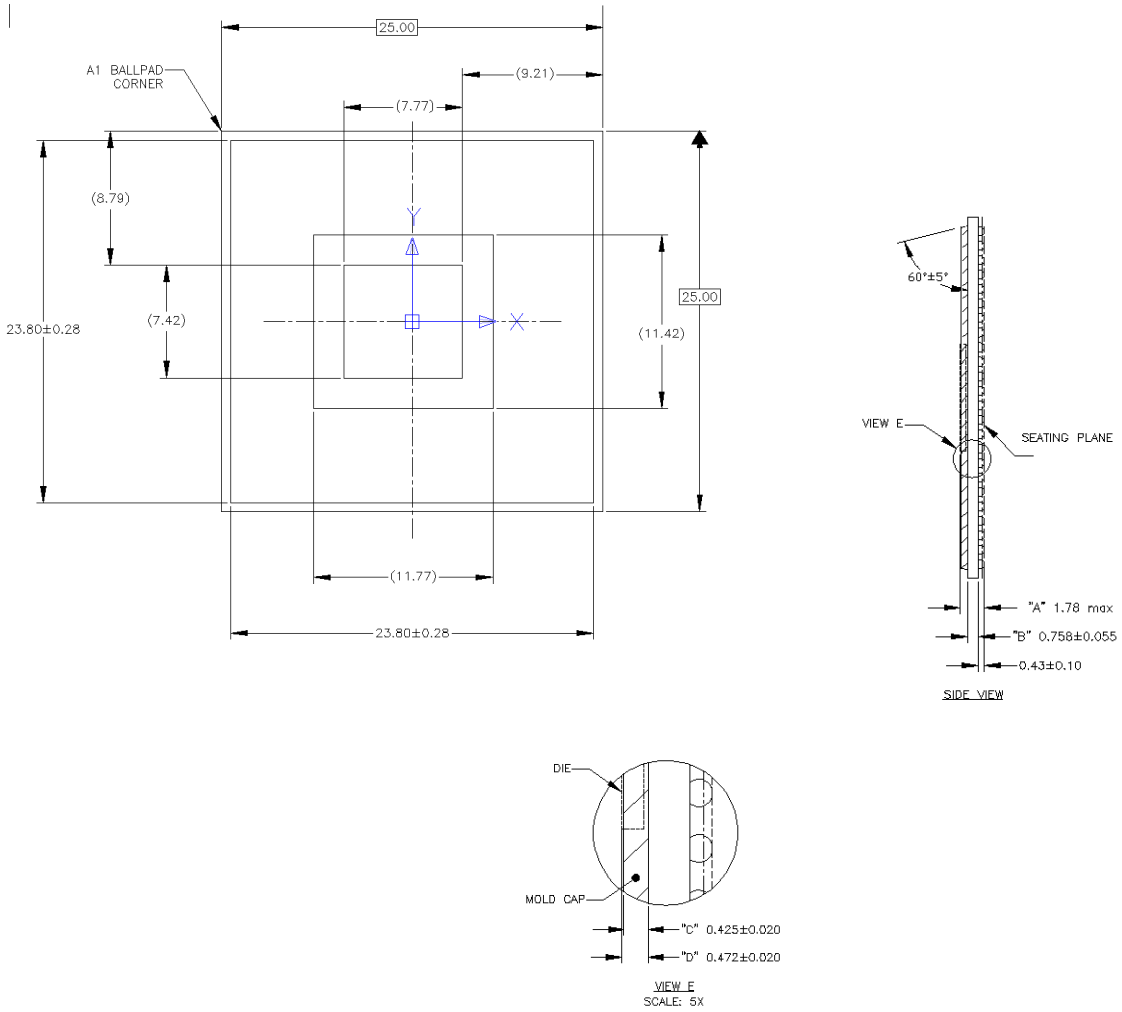


Figure 16-4. XL710 FCmBGA mechanical illustration

16.8 Thermal diode

The XL710 provides externally accessible thermal diodes for accurate die temperature measurements.

The thermal diode is accessible through the following external pins:

Signal Name	Pin Name	Ball #
RSVDY17_NC	THERM_DPS0	Y17
RSVDY18_NC	THERM_DPL0	Y18
RSVDJ7_NC	THERM_DPS1	J7
RSVDH7_NC	THERM_DPL1	H7

The recommended way for using the thermal diodes to connect an ADT7461 IC to pins RSVDY18_NC (THERM_DPL0) and RSVDH7_NC (THERM_DPL1). Pins RSVDY17_NC (THERM_DPS0) and (THERM_DPS1) should not be used for temperature measurements, and are for Intel use only. Anyone attempting to use ADT7461 should familiarize themselves with the ADT7461 Datasheet and Application Notes for configuration setup and layout consideration.

16.9 Measuring the thermal diode manually

Follow these steps to manually measure the thermal diode temperature:

1. Apply power to the XL710 and ensure all supply voltage rails are within regulation.
2. Apply 1 mA (maximum) current between the pins THERM_DPL and GND.
3. Measure the voltage between THERM_DPLx and GND. The voltage is inversely proportional to the temperature.
4. Use the following formula to calculate the temperature of the thermal diode:

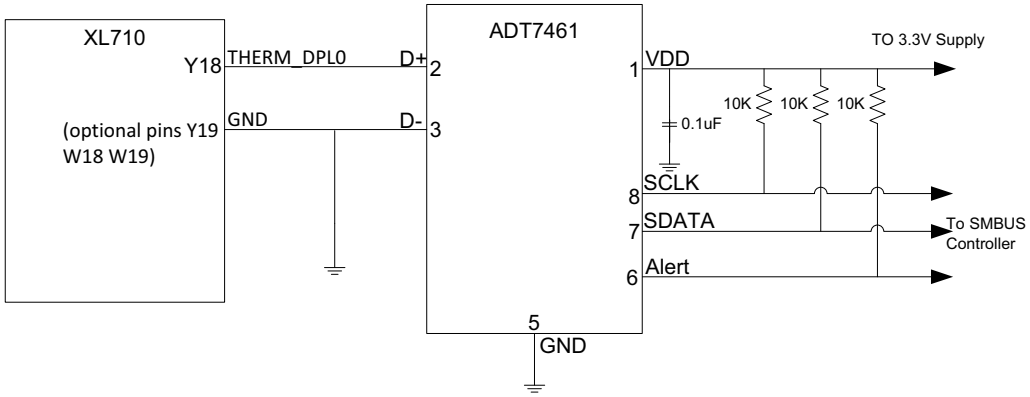
   $$\mathbf{T} = \mathbf{V}_{diode} + \mathbf{A}_{param} + \mathbf{B}_{param}$$

   When T is at [C] and V_{diode} is at [V], Parameters A_{param} + B_{param} are different for each Thermal Diode (TD0, TD1), and also depends on the current level driven to the TD during the measurement. The table below shows A_{param} + B_{param} values for 1 mA current.

Diode	A_{param}	B_{param}
TD0 (THERM_DPL0)	-558.33398	418.22971
TD1 (THERM_DPL1)	-578.43018	432.48579

16.10 Common application schematic

The common application schematic is the recommended way to implement thermal monitoring on an the XL710. For simplified setup, the SMBus connection can be shared with the XL710 (not shown) in a multi-drop configuration.



Note: Note that if RSVDH7_NC (THERM_DPL1) is used, optional ground pins of the XOL710 are H6, G6 and G7. Also note that to ensure good measurements, the ground of the XL710, ADT7461 and pin 3 (D-) of ADT7461 must be connected together to the same ground path.

16.11 ADT7461 SMBus access

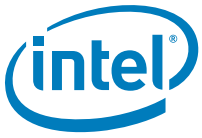
The ADT thermal monitor is accessible through SMBus transactions to the host machine. Temperature can accessed using the "External Temperature Value High Byte" (0x01) to read in the current value in °C. In some applications "External Temperature Value Low Byte" (0x10) is available and provides 2 more bits of resolution (down to 0.25 °C) for temperature readings. In most cases, it does not need to be implemented.

Note: When used in extreme environments, in its default configuration the ADT temperature reading is not valid below 0 °C.

ADT7461: ±1°C Temperature Monitor with Series Resistance Cancellation. Available at:

http://www.onsemi.com/pub/Collateral/ADT7461-D.PDF

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



17.0 Glossary and Acronyms

This section defines terms and acronyms commonly used throughout this specification. Another source for this type information is Section 1.4, "Overview of standards" which contains some of the acronyms defined in the industry standards (e.g. MII, MPA, Verbs, NC-SI, EEE, etc).

Table 17-1. Glossary and Acronyms

Auto-negotiation (AN) – An IEEE Ethernet method for exchanging PHY layer parameters such as speed and duplex mode between link partners.
Base Address Register (BAR) – Standard registers defined in PCI Config space that define how an I/O adapter responds to host memory-mapped I/O requests
Bit Error Rate (BER) – Number of bits received in error, divided by the total number of bits received.
Baseboard Management Controller (BMC) – 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.
Completion (Completed, Completes, etc) – When an RNIC has performed all functions specified for a given WQ operation, including Placement and Delivery, that WQ operation is said to be "completed". This can be determined by the Consumer through a Work Completion for Signaled Work Requests.
Completion Event Queue (CEQ) – The XL710 writes Completion events to this shared queue to make it easy for software to determine which CQ has new CQEs.
Completion Queue (CQ) – A sharable queue which can contain one or more Completion Queue Entries. A Completion Queue is used to create a single point of completion notification for multiple Work Queues. The Work Queues associated with a Completion Queue may be from different QPs and of differing queue types (SQs or RQs).
Completion Queue Entry (CQE) – Info that the XL710 writes onto a Completion Queue during the process of performing a Completion.
Consumer – A software process that communicates with the XL710. The Consumer typically consists of an application program or an OS adaptation layer which provides some OS-specific API.
Control / Status Register (CSR) – I/O adapter registers typically accessed by the host using memory-mapped I/O requests.
Cyclic Redundancy Check (CRC) – An error detecting code commonly used to protect network transmissions.
Direct Data Placement (DDP) – While processing a Work Request that requires sinking data to the Consumer, the XL710 will often perform Direct Data Placement (DDP). DDP is the XL710 process of autonomously moving data from the fabric directly into its final destination in memory. No intervention by OS/application software is required to perform DDP. FCoE Engine are DDP-capable.
Direct Write Exchange Offload (DWO) – DWO is a feature of the FCoE Engine. It is a hardware offload that eliminates host CPU involvement in the processing of FCP XFER_RDY Frames sent from an FCoE target to initiator.
Embedded Management Processor (EMP) – A the XL710 embedded processor that handles device initialization and also device configuration based on commands received from an Admin Queue, a management interface (e.g. NC-SI), or from a network port.
Error Correcting Code (ECC) – A code commonly used to protect network transmissions which allows both detection and recovery from errors.
Function-level Reset (FLR) – 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.

Table 17-1. Glossary and Acronyms (Continued)

Function Private Memory (FPM) – The XL710 uses host memory as backing store for a number of context objects such as queue state, FCoE DDP state, and iWARP objects. These objects are cached in the XL710 Host Memory Cache (HMC). Each XL710 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.
Interrupt Throttling (ITR) – A method of interrupt moderation that guarantees a minimum gap between two consecutive interrupts.
Keyboard, Video, Mouse (KVM) – Standard bundle of computer user I/O devices
LAN On Motherboard (LOM) – When the XL710 is integrated on a server motherboard, it qualifies as a LOM NIC.
Link Aggregation Control Protocol (LACP) – The IEEE protocol that enables the formation of Link Aggregation Groups (aggregation of one or more Ethernet links into a single logical link).
Link Layer Discovery Protocol (LLDP) – The IEEE protocol that enables a server to advertise it's identity, capabilities, and interconnections to other entities on an Ethernet fabric.
Lower Layer Protocol (LLP) – The protocol layer beneath the protocol layer currently being referenced. For example, for TCP the LLP is IP. Etc.
Management Data Input/Output Interface (MDIO) – A standard interface to connect a MAC to a PHY, used to access PHY registers.
Maximum Segment Size (MSS) – The largest amount of data that a network device can handle in a single, unfragmented piece.
Memory Window (MW) – 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.
Multi-Function Device (MFD) – A PCI Device with more than one PCI function
Multi-Function per Port (MFP) – 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)*.
Network Interface Controller (NIC) – An I/O adapter that enables a computer to communicate over a network.
Packet – A unit composed of headers, data and footers that are sent or received by a device. Also known as a frame.
Page List – A list of physical addresses describing a set of memory pages, which specifies the page size, list of physical addresses, and offset to the start of the memory region within the first page. The starting physical addresses of each page is aligned on power-of-two addresses and the size of the page is a power of two. Note that it is possible for the starting offset to be an offset into the first page and to be of a byte granularity and the entire list may have an arbitrary length.
Physical Buffer List (PBL) – In iWARP terminology, a Physical Buffer List can either be a Block List or a Page List. The XL710 does not support Block Lists, so in the context of the XL710, PBLs and Page Lists are the same thing.
Physical Function (PF) – A PCI Function that supports the PCI-SIG *Single Root I/O Virtualization and Sharing Specification*.
Quad – In the XL710 Datasheet, 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.
Receive Side Scaling (RSS) – 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 XL710 under OS control into groups of conversations. Each group of conversations is assigned its own receive queue and receiving processor.
Receive Queue (RQ) – 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.
RNIC – The generic term for a device that implements iWARP verbs functionality. The XL710 is an RNIC.

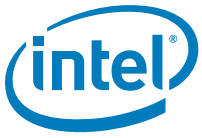


Table 17-1. Glossary and Acronyms (Continued)

Single-Function per Port (SFP) – 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)*. **Small Form-factor Pluggable** (SFP) – A *small form-factor pluggable* (SFP or SFP+) module is a compact, hot-pluggable transceiver that interfaces a network device like the XL710 to a fiber optic or copper networking cable.
Software Definable Pins (SDPs) – The XL710 device pins that have a high degree of software configurability and programmability. Also called "general purpose I/Os" (GPIOs).
Type, Length, Value (TLV) – 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 XL710 that uses TLV encoding is IEEE Link Layer Discovery Protocol (LLDP).
Transmit Segmentation Offload (TSO) – 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.
Upper Layer Protocol (ULP) – The protocol layer above the protocol layer currently being referenced. For example, for IP the ULP is TCP. Etc.
Virtual Ethernet Bridge (VEB) – This is an IEEE EVB term. A VEB is a VLAN Bridge internal to the XL710 that bridges the traffic of multiple VSIs over an internal virtual network.
Virtual Ethernet Port Aggregator (VEPA) – This is an IEEE EVB term. A VEPA multiplexes the traffic of one or more VSIs onto a single the XL710 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.
Virtual Function (VF) – 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.
Virtual Machine Devices queues (VMDq) – Virtual Machine Devices queues (VMDq) is a collection of NIC features designed to offload the hypervisor from performing classification and distribution of received packets to the Virtual Machines under its control. There are two versions of VMDq: VMDq1 and VMDq2. VMDq2 is a superset of VMDq1. Its major new features are: internal switching from a Transmit Queue to a Receive Queue, the ability to replicate received multicast and broadcast packets to multiple Receive Queues, the ability to sort received packets based on a combination VLAN tag and MAC address filter, and anti-spoofing transmit filters.
Virtual Machine Monitor (VMM) – A software component that allocates/exports/isolates server resources to the Virtual Machines (aka System Images). Other terms for VMM in this specification: Hypervisor, Virtualization Intermediary (VI).
Vital Product Data (VPD) – VPD is defined in the *PCI Local Bus Specification* (this is the conventional PCI spec, not PCIe). VPD is information that uniquely identifies hardware and software elements of a server. The VPD provides a server OS with information on various Field Replaceable Units such as part number, serial number, etc.
Virtual Station Interface (VSI) – This is an IEEE EVB term that defines the properties of a virtual machine's (or a physical machine's) connection to the network. Each downstream v-port on a the XL710 VEB or VEPA defines a VSI. A standards-based definition of VSI properties enables network management tools to perform virtual machine migration and associated network re-configuration in a vendor-neutral manner.
Wake on LAN (WoL) – Wake on LAN is an Ethernet technology that enables a computer to power on or "wake up" upon receipt of a network message, often called a "magic packet". Newer specifications in this area call WoL *Advanced Power Management wake up*.
Weighted Round-Robin (WRR) – A scheduling or arbitration algorithm which selects amongst requesters in a round-robin fashion, and in which each requester can be programmed to receive a different percentage share of resource bandwidth.

Table 17-1. Glossary and Acronyms (Continued)

Weighted Strict Priority (WSP) – A scheduling or arbitration algorithm which selects the requester with highest priority that has not exhausted its programmed percentage share of resource bandwidth.
Work Queue (WQ) – One of either a Send Queue or Receive Queue.
Work Queue Element (WQE) – A Work Request transformed by software into XL710-specific format, and enqueued on a Work Queue.
Work Request (WR) – A request to the XL710 by the Consumer to perform some operation. Gets transformed by software into a Work Queue Element.

Appendix A Transmit Scheduling

This section provides a high level description of internal scheduling structures, scheduling concepts and algorithms deployed by the XL710 scheduler. Standard software drivers do not have an ability to program scheduler directly, and they are limited to the programming interface exposed by admin queue, and described in Section 7.8.4.

This section enables a privileged device driver developers take advantage of the full scope of scheduler capabilities, and not be limited by standard configurations exposed by programming interface.

This section can be useful for advanced reader (firmware engineers) that are willing to have a better understanding of scheduler operation.

A.1 Hierarchical Scheduling Concept

A.1.1 Abstract Scheduling Tree

This section describes a concept of the tree scheduling, terminology and logical scheduling tree structure.

Diagram below shows an abstract tree structure. The structure of this tree and terminology is described below.

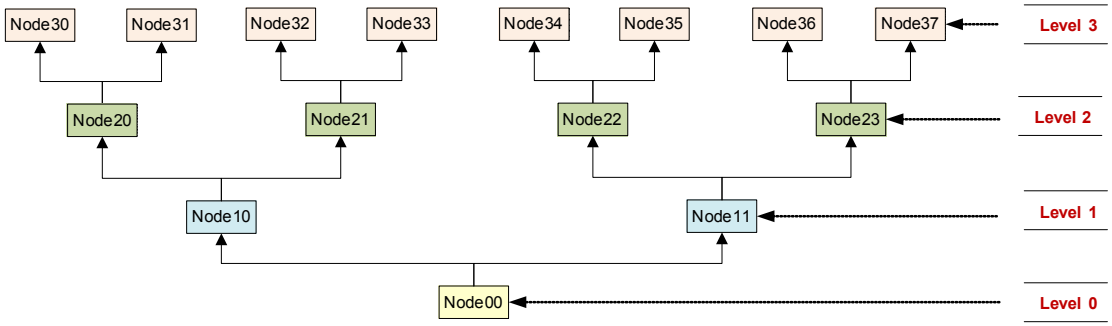


Figure A-1. Abstract Tree Structure

Tree components:

- **Node** - is a basic tree element. Each Tree Node may have at most one Parent Node, and one or more Child Nodes. Definition of tree configuration concludes establishment Parent-Child relation between all Tree Nodes.

1. **Root Node** - Tree Node that does not have a Parent Node. Root Node occupies the lowest level of the tree - Level 0. Node00 is a Root Node of the Tree in the example above. Scheduling tree traversals always starts from the Root Node.

- **Parent Node** - Tree Node that has one or more Child Nodes. Parent Node always located on the lower tree Level with respect to Child Nodes. Node10 is a Parent Node, with two Child Nodes Node20 and Node21. Parent Node leads to its Children Node at scheduling tree traversal.
- **Child Node** - Tree Node that has a Parent Node. Note that Child Node might be a Parent Node with respect with other Nodes. Node20 is a Child Node to Node10, and Parent Node with respect to Node30.
- **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, having Node10 as a Parent Node. Scheduling tree traversal involves selection of one of the Siblings in the tree to proceed traversal.
- **Leaf Node** - Tree Node that does not have any Child Node. Leaf Node is located on the highest level of the Tree. Node30 is a Leaf Node. Leaf Node is an end point of scheduling tree traversal, and refers to the actual work that needs to be done.
- **Tree Level** - Tree Nodes that are located on the same tree elevation. Nodes located on the same Level not necessarily have same Parent Node, since they might belong to different sub-trees. Nodes Node20 and Node22 are located on the same Tree Level - Level 2, but they belong to different Subtrees, since they have different Parent Nodes, Node10 and Node11 respectively.
- **Subtree** - Portion of the tree comprised of the Tree Node as a Root Node and its descendants. Subtree of Node10 includes Node20, Node21, Node30, Node31, Node32, Node33.
- **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** - Node is considered to be Blocked, if it cannot satisfy scheduling constrains applied, or all Child Nodes of that Node are Blocked. Leaf Node can also be Blocked because it does not have work available in any of transmit descriptor queues associated with it. Blocked Node cannot be included in scheduling tree traversal. This Blocked Node propagation characteristic guarantees that if Node is not Blocked, then there exists traversal from this Node to one of the Leaf Nodes, where all Nodes on that traversal are not Blocked.

A.1.2 Tree-based scheduling

Tree based scheduling is a process of finding a next Leaf Node to be scheduled. Effectively it involves finding a path in the tree leading from the Root Node to one of Non-Blocked Leaf Nodes, by traversing tree Nodes from Parent to Child. Found path cannot consist of any Blocked Nodes. Blocked Node cannot be included to the scheduling traversal. If Parent Node is not Blocked, then one of Child Nodes should not be blocked ever. This guaranties that if Node has been selected to scheduling traversal, there is a path from that Node leading to the Leaf Node and consisting of Not Blocked Nodes. Root Node serves as a base of the tree. Leaf Nodes are the only Nodes that are referring to the schedulable work items. The rest of the tree Nodes are used to enable flexible hierarchical arbitration and resource distribution.

Following a definition of the scheduling Tree traversal algorithm:

1. Select Root Node
2. Go to the next Tree level, i.e. Children Nodes
3. Select one of the Sibling Nodes.
4. Selection of the Sibling Node depends on arbitration scheme and constrains applied on each of the Siblings. See Appendix A.1.3 and Appendix A.1.5 for description of the arbitration schemes and bandwidth distribution mechanisms.
5. If selected Node is a Leaf Node, scheduling tree traversal is completed
6. Otherwise goto step 2 above.

Following an example of the scheduling traversal in the Tree shown in the diagram above.

- Select Root Node - Node00
- Climb to the next tree level (Level1), and select a Node among Siblings (Node10 and Node11), e.g. Node10
- Climb to the next tree level (Level 2), and select a Node among Siblings (Node20 and Node 21), e.g. Node21
- Climb to the next tree level (Level 3), and select a Node among Siblings (Node32 and Node33), e.g. Node33
- Node33 is a Leaf Node, tree traversal is complete

A.1.3 Bandwidth Allocation and Bandwidth Limit

The XL710 supports two 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. Both constrains are stated in the terms of bandwidth. Figure A-2 below graphically demonstrates both concepts, and their difference.

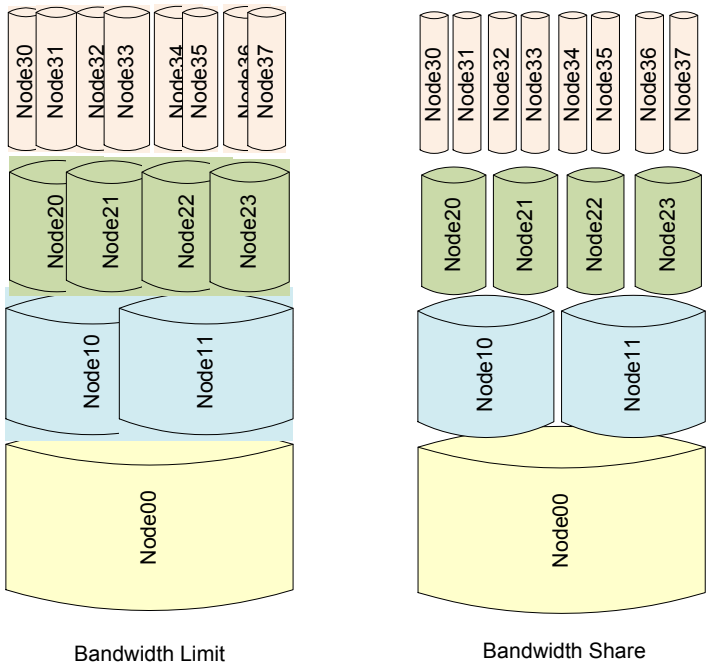


Figure A-2. Bandwidth Limit and Bandwidth Allocation

- **Relative Bandwidth Allocation** or bandwidth share - allows to specify a relative share of available bandwidth among Sibling Nodes. Sibling Nodes are sharing a 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. Once Node used its bandwidth share, it cannot be scheduled and considered to be Blocked, until all not Blocked Sibling Nodes had a chance to use their bandwidth share. Bandwidth share can be expressed in terms of weights or credits assigned to each tree Node.

-
- Sibling Nodes in the tree are sharing bandwidth of the Parent Node. For example, Node20 and Node21 are sharing bandwidth of Node10. If the share of the Node20 is 20% and share of Node21 is 80%, and if Parent Node10 allows 10Gb/s, then the bandwidth available for the Node20 is 2Gb/s, and bandwidth available for the Node21 is 8Gb/s. If Parent Node10 allows 5Gb/s, then the bandwidth available for the Node20 is 1Gb/s, and bandwidth share of the Node21 is 4Gb/s.
- Bandwidth share can be individually assigned to each Node in the tree.
- **Best Effort or Zero-Bandwidth Allocation** or bandwidth share. While relative bandwidth allocation described above guaranties relative part of the bandwidth to be always available for the Sibling Node, best effort or zero-bandwidth allocation does not guarantee any bandwidth allocation for the Node, but allows this Node to use a portion of the remaining bandwidth after all Sibling Nodes with relative bandwidth allocation had used their bandwidth share.
- For example, Node20 has assigned 50% of the Node10 bandwidth, and Node21 has a zero-bandwidth allocation. In this configuration, assuming that both Nodes have always data to transmit, Node20 will use its 50% of the Node10 bandwidth allocation first, and then both Nodes will share the remaining 50%. As a result Node20 will be using 75% and Node21 will be using 25% of Node10 bandwidth.
- **Bandwidth Limit** - allows to specify a maximum bandwidth that can be consumed by the tree Node. Child Node cannot consume more bandwidth than allowed by the bandwidth limit of the Parent Node. However sum of the bandwidth limits of the Children Nodes can exceed bandwidth limit of the Parent Node and potentially allow better utilization of the available bandwidth. Diagram above illustrates this concept, where Child Nodes represent an overlapping virtual pipes contained within Parent Node virtual pipe.
- Each tree Node may have an individually assigned bandwidth limit. Some Nodes in the tree may need to share Bandwidth Limit.
- **Shared Bandwidth Limit** - allows to specify a bandwidth limit for the group of Nodes that are not co-located in the tree, for example Child Nodes of the different Parent Nodes. Shared bandwidth limit should constrain a total bandwidth that can be used by those Nodes, similar to the bandwidth limit of the Parent Node restricting a total bandwidth available to all Child Nodes.
- For example Node20 and Node23 can be configured to have a Shared Bandwidth limit, and then the total amount of bandwidth available for these Node should not exceed a bandwidth available to the Node’s Parents Node10 and Node11 respectively, and sum of the bandwidth used by Node20 and Node23 should not exceed shared bandwidth limit. Node cannot have assigned both regular bandwidth limit and shared bandwidth limit.

A.1.4 Scheduling Rules

To guarantee correct scheduling following rules must apply:

- Any Tree Node can be selected for scheduling traversal or scheduled only if it has non-zero bandwidth share, and it has not reached its maximum bandwidth limit. If one of this conditions does not hold, Node is considered to be Blocked, and cannot be scheduled.
- Leaf Node can be scheduled only if an associated set of transmit queues, has a work available for transmission. Leaf Node is a tree Node and bandwidth limit and bandwidth share constrains can be applied to it as to any other tree Node as described above.

- Parent Node can be scheduled only if it has at least one Child Node that can be scheduled. Parent Node is a tree Node, and subject to the bandwidth limit and bandwidth shared constrains described in the first bullet.

This set of rules allows to guarantee that Node is added to the scheduling traversal, only if path exists from this Node to Not Blocked Leaf Node consisting Not Blocking Nodes among the Node descendants. This prevents false scheduling, and makes a depth of the scheduling tree traversal equal to the depth of the tree.

A.1.5 Arbitration Schemes

The XL710 supports several arbitration schemes that can be used to select a Node among Siblings during scheduling tree traversal. Each set of Sibling Nodes has an arbitration scheme assigned to it. Siblings are participating in arbitration only if they are Not Blocked.

Each Node may have two sets of credits associated with bandwidth allocation credits and bandwidth limit credits.

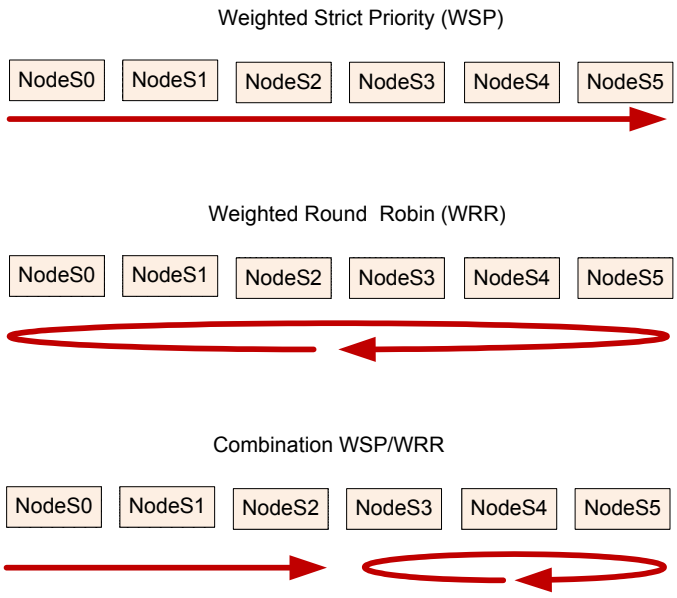


Figure A-3. Arbitration Options

- **Weighted Round Robin** (WRR) - This arbitration scheme equally prioritizes Nodes, and gives every Sibling Node chance to be selected in cyclic round robin fashion as long as it has credits and not Blocked.
- Arbiter iterates Sibling Nodes in cyclic round robin fashion. Every scheduling cycle arbiter moves to the next Sibling Node, even if credits of the current Node are not completely used. This allows even distribution of scheduling among Sibling Nodes as long as they have remaining scheduling credits. If current Node is Blocked, then arbiter moves to the next Sibling Node. If Arbiter used all Node

credits, it replenishes Node credits immediately, but does not use them until all not Blocked Nodes had their credits used and replenished.

- **Weighted Strict Priority** (WSP) - This arbitration scheme attempts to Schedule Nodes based on their priority as long as Node stays within its bandwidth allocation with respect to other Sibling Nodes, and within its bandwidth limit.
- Sibling Nodes are ordered accordingly to their priority. Arbiter always starts iteration from the first Sibling Node. If current Node is Blocked, Arbiter moves to the next Sibling Node. Arbiter effectively keeps scheduling Node with highest priority as long as this Node has credits and not Blocked.
- **Combination of Weighted Strict Priority and Weighted Round Robin** - This arbitration scheme combines WRR and WSP schemes described above. Sibling Nodes should be organized accordingly to the arbitration scheme: WSP Nodes first, followed by WRR Nodes. Each Node can be scheduled using one arbitration scheme only. WSP Nodes are ordered based on their priority. Each scheduling cycle arbiter should attempt to schedule a Node from WSP Nodes using WSP arbitration scheme described above. If all WSP Nodes are Blocked, arbiter should attempt to select a Node from WRR Nodes. For WSP Nodes arbiter should always start arbitration from the first Node, and for WRR Nodes, it should always start arbitration from the last scheduled Node.

A.1.6 Shared Bandwidth Limit

Bandwidth limit attribute described in Appendix A.1.3 can be configured for individual Switching Component, VSI, or TC/UP. Some usage models imply instantiation of the multiple tree Nodes for the same system resource, such as VEB or VSI. In order for the system resource be bandwidth limited, Transmit Scheduler must support shared bandwidth limit to multiple tree Nodes. See Appendix A.2.1.1 for example of such configuration.

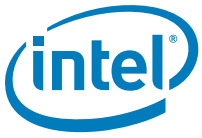
Not all scheduler configuration require use of the Shared Bandwidth Limit. See Appendix A.2.1.2 for example of bandwidth distribution scheme that does not use Shared Bandwidth Limit.

Instantiation of the shared bandwidth limiter, involves instantiation of the dedicated rate limiter for tree Node representing same system resource. If all tree Nodes use their dedicated credits, then no shared credits are accumulated. If some of the tree Nodes do not use their credits (e.g. due to lack of work available), shared bandwidth limit credits are accumulated, and other tree Nodes representing same system resource can take advantage of available shared credits. This scheme allows starvation free implementation of the Shared Bandwidth limits.

For example, if software instantiated shared rate limiter of 8Gb/s, and associated this rate limiter with VSI having 4 TCs configured, then each of VSI tree Nodes can be configured with 2Gb/s dedicated bandwidth limits. If all TCs of the VSI transmitting 2Gb/s worth of data, then no shared credits are accumulated, and VSI is limited to all its TCs to 2.5Gb/s. If one of the TCs currently does not have data available for transmission, other 3 TCs, can take advantage of available 2Gb/s bandwidth, and increase its bandwidth limit to 2.6Gb/s.

Instantiation of the Shared Bandwidth Limit and corresponding dedicated bandwidth limits is responsibility of firmware and transparent to software. Depending on configuration, firmware may translate software request to instantiate a bandwidth limit to the system resource to instantiation of several dedicated bandwidth limits and one shared.

Dedicated bandwidth limits must be configured in such way, that their total should add up to the Shared Bandwidth Limit. Effectively Shared Bandwidth Limit credits are distributed among dedicated bandwidth limits. Firmware can choose distribute shared bandwidth limit equally, or relatively to the bandwidth allocation.



The XL710 implementation of the shared bandwidth limits allows each tree Node use its dedicated credits even after periods of inactivity. Shared bandwidth limit has a control (max amount of accumulated credits) which allows to control burst generated by tree Nodes associated with the same Shared bandwidth limit.

A.2 The XL710 Tree-based Scheduler

The XL710 employs a centralized scheduling approach. All scheduling decisions are made by the main scheduler, and followed by arbiters throughout the transmit path. The XL710 scheduler schedules work for both stateless and stateful transmit queues.

The XL710 Stateless and Stateful transmit descriptor queues are organized in a Queue Sets, stateful and stateless Queue Sets respectively. The XL710 Scheduler each scheduling cycle selects one of Queue Sets. Index of selected Queue Set is passed to the transmit processing logic. Selection of the transmit queue from the Queue Set is outside of the scope of scheduler definition, and should be done by transmit processing logic.

Each Leaf Node in Scheduler tree is identified with a pair of Queue Sets. Selection of the Leaf Node is done using scheduling tree traversal, as described in Appendix A.1.2. Once Leaf Node selected, Scheduler selects one of two Queue Sets to be served at the current scheduling cycle. Arbitration between Queue Sets is round robin, one scheduling request in a time. If Queue Set does not have work available, second Queue Set is chosen instead. One of the Queue Sets must have work available to comply with scheduling rules described in Appendix A.1.4.

With each selected queue set Scheduler specifies a Quanta - number of bytes that can be transmitted per scheduling cycle. Quanta can span multiple transmit queues and multiple descriptors. It is up to the transmit processing logic to decide how many descriptors and queues use. Queues assigned to the same Queue Set have same priority and arbitrated round robin by processing logic. The XL710 support one Quanta that can be configured to values in range from 4KB-64KB. Software should be advised to keep Quanta value low, to keep burstiness of traffic generated by the XL710 low. This is particularly important for the bandwidth limited traffic. Once Queue Set has been selected, a Quanta worth of traffic for this Queue Set would be generated at wire-speed regardless of the Queue Set or scheduling path bandwidth limit. Increase of Quanta only recommended to resolve potential performance deficiencies cause by small Quanta value.

The XL710 hardware does not associate physical identity with a scheduling tree Node. Thus each Node can be configured to represent any physical resource

To allow proper support of PFC, each Node carries a bitmap of TCs that can be currently scheduled thru this Node (i.e. there is at least one Leaf Node among Node descendants that belongs to that TC, has transmit work available and not Blocked). The XL710 Scheduler uses a bitmask register allowing to mask off TCs that are blocked by PFC, and prevent them from scheduling.

Note, though Scheduler allows to top level of arbitration be UP, PFC is still performed on TC level, and therefore UPs mapped to the same TC will be flow controlled together.

Queue set can be scheduled only if one of the transmit queues belonging to the set has work available, i.e. descriptors are posted by software and ready for transmission.

A.2.1 Basic Scheduling Tree Configuration Options

The XL710 scheduler is based on the abstract scheduling tree concept described in the Appendix A.1.1. Structure and configuration of the scheduling tree depends on the system configuration and expected bandwidth distribution model. It is configured using scheduler configuration tables described in the Appendix A.3. Those tables define a tree topology, bandwidth allocation, arbitration scheme, bandwidth limit for each of the tree Nodes.

Abstract scheduling tree structure, described in the Appendix A.1.1, allows abstract representation of various complex chip configuration. Each tree Node can be associated with a physical resource, such as S-Comp, VEB/VEPA, VSI, TC and UP etc. Tree structure is very convenient to represent hierarchical relationship and dependency of various chip resources, and allows scheduling process naturally follow the chip resource hierarchy. Use of abstract tree structure, when tree Nodes can represent different chip resources, allows great flexibility, and ability to support wide variety of system and chip configurations without creating dedicated solution for each one of them.

The XL710 support variety of the Scheduling Tree configurations. Two main configurations are enabled by the general purpose Programming Interface, and those are primarily discussed here. Alternative configurations can be enabled using Privileged Programming Interface.

Primary two Scheduler configurations modes are: ETS-Based and VNet-Based configurations. Configuraiton option selection is done on per Physical Port base. It is programmed by firmware using TBD register and kept in NVRAM.

Both basic configuration schemes are following topology of internal switching fabric, and allow software to configure bandwidth allocation and bandwidth limits to various switching components and VSIs. The main difference is a bandwidth configuration options available in each of schemes. Following sections point out difference and configuration options provided by each scheme.

A.2.1.1 ETS-Based Scheduler Configuration

Figure A-4 below shows a logical diagram depicting a concept of ETS-Based bandwidth distribution scheme.

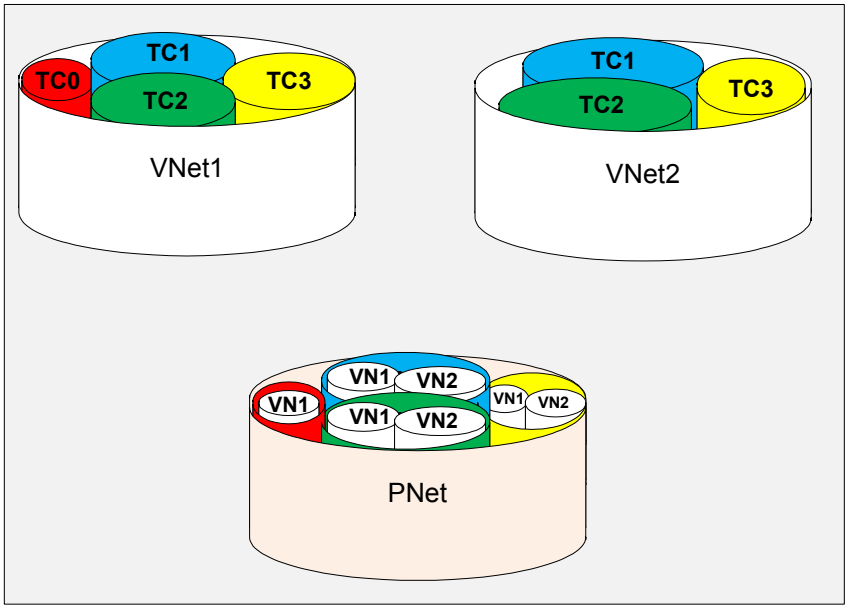


Figure A-4. ETS-Based Bandwidth Distribution

This figure shows two Virtual Networks VNet1 and VNet2. Each virtual network carries traffic of different types by Traffic Class (TC). Both Virtual Networks are merging into Physical Network. Physical Network is configured accordingly to ETS specification. Bandwidth of physical network is divided between Traffic Classes. Each Traffic Classes carries traffic belonging to respective traffic types of both Virtual Networks. Bandwidth distribution between Virtual Networks within same Traffic Class is invisible outside of the chip.

Diagram of the Physical Network shows a concept of the bandwidth distribution in ETS-based configuration. Bandwidth is first distributed between types of traffic (or Traffic Classes), and then it is distributed between virtual networks within each Traffic Class. Therefore if one of the Virtual Networks does not have traffic of the certain type currently available, the remaining bandwidth would be consumed by the same traffic type from other Virtual Networks. E.g. if VNet2.TC3 does not have any work available, the remaining TC3 bandwidth will be consumed by VNet1.TC3, as long as it has enough work available, and does not exceed any bandwidth limit.

This scheme allows bandwidth allocation per traffic type for each switching component and VSI, and does not support a bandwidth allocation per Virtual Network.

This is a default bandwidth allocation scheme for the XL710 Scheduler.

Figure A-5 shows an example of the system configuration, and bandwidth allocation using ETS-Based scheduler configuration scheme.

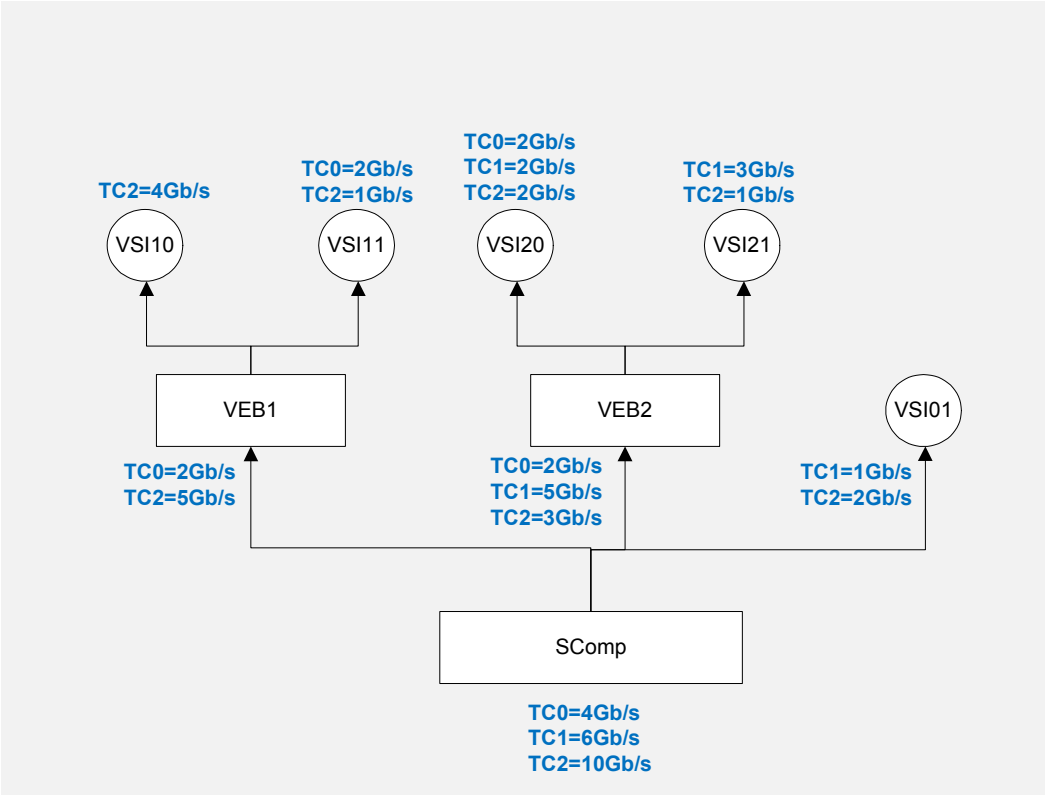


Figure A-5. Example of ETS-Based System Configuration

Each one of the switching components (SComp, VEB1, VEB2) and VSIs can be configured with ETS (bandwidth allocation per traffic type or TC) and bandwidth limit. Bandwidth allocation must be consistent, i.e. if TC has 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=4Gb/s, then VEB1.TC0 + VEB2.TC0 + VSI01.TC0 = 4Gb/s, and VSI10.TC0 + VSI11.TC0 = VEB1.TC0.

Figure A-6 below shows an ETS-based Scheduling Tree configuration corresponding to the system configuration shown above.

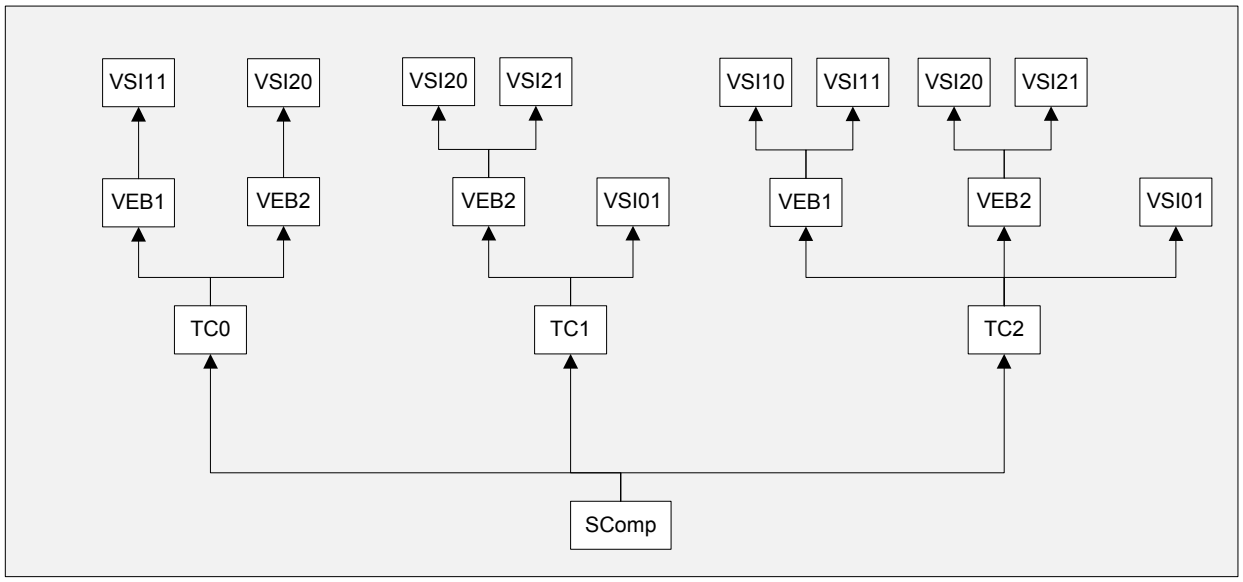


Figure A-6. ETS-Based Scheduling Tree Configuration Example

Structure of this scheduling tree resembles topology of the system configuration shown on Figure A-5 . This tree has four tree levels (one not shown):

- TC Level - this is the lowest tree level. It is configured with Physical Port or SComp egress port ETS bandwidth allocation (e.g. Nodes TC0, TC1 and TC2 on the Figure above).
- UP Level (not shown on the diagram) is the next tree level. It also follows SComp egress port ETS configuration. If more than single UP is associated with TC, this level defines a bandwidth allocation among UPs associated with same TC.
- Switching Component Level - is a next scheduling tree level. This level consists of Switching Components (VEB/PA) and VSIs directly attached to SComp. Each tree Node corresponds to the TC/UP configured for the Switching Component/VSI. Single Switching Component and VSI may appear in the tree multiple times, once for each configured TC/UP. (e.g. Nodes VEB1_TC0, VEB1_TC2, VEB2_TC0, VEB2_TC1, VEB2_TC2, VSI01_TC1 and VSI01_TC2 on the Figure above).
- VSI Level - last tree scheduling level. This level consists of the VSIs of Switching Components. Each tree Node corresponds to the TC/UP configured for the VSI. Single VSI may appear in the tree multiple times, once for each configured TC/UP. (e.g. 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. Software sees a logical bandwidth configuration diagram shown on Figure A-5 , and controls exposed to software relate to the switching elements and VSIs shown on Figure A-5 . Firmware creates and manages 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. Firmware allocates a multiple scheduling Nodes for VEB (VEB1_TC0 and VEB1_TC2), and configures those Nodes with credits provided by software.

Scheduling tree can be extended with more scheduling levels. Maximum depth is the tree supported by hardware is 7.

Software can enable bandwidth limit to any switching component and VSI. If Switching Component or VSI has more than single Traffic Class enabled, it appears multiple times in the Transmit Scheduling Tree. To configure bandwidth limit to such Switching Component or VSI, firmware must use a Shared Bandwidth Limits described in Appendix A.1.6. Use of Shared or basic bandwidth limits is hidden from software, and decision whether to use basic or shared bandwidth limit is done by firmware depending on the selected bandwidth allocation mode, and system configuration.

A.2.1.2 VNet-Based Scheduler Configuration

Figure A-7 below shows a logical diagram depicting a concept of ETS-Based bandwidth distribution scheme.

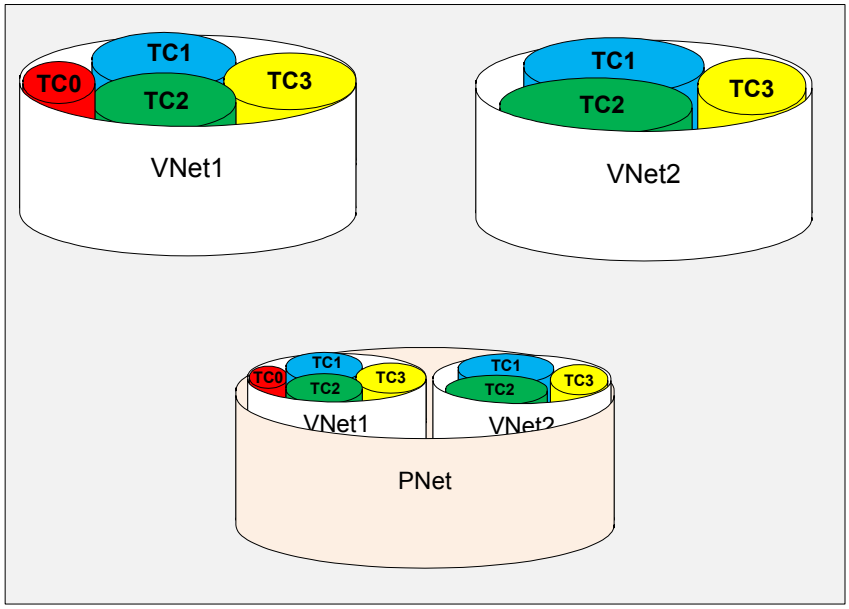


Figure A-7. VNet Based Bandwidth Distribution

This figure shows two Virtual Networks VNet1 and VNet2. Each virtual network carries traffic of different types by Traffic Class (TC). Both Virtual Networks are merging into Physical Network. Physical Network is configured to distribute bandwidth between Virtual Networks first, and then distribute bandwidth within each Virtual Network between traffic types configured for that Virtual Network.

This bandwidth distribution scheme seems to be logical for the cloud-based environments with multiple tenants willing share physical resources, providing isolation of bandwidth allocation throughout the fabric. Current standards do not allow take advantage of this bandwidth distribution scheme.

This scheme allows to distribute bandwidth between Virtual Networks, and provide a global SLA guaranties, and not limit those to the particular traffic type, as ETS-based scheme described in Appendix A.2.1.1. Within each Virtual Network bandwidth can be then distributed between types of traffic. E.g. if VNet2.TC3 does not have any work available, the remaining bandwidth will be consumed

by other traffic types configured for VNet2 (VNet2.TC1 and VNet2.TC2) as long as they have enough work available, and do not exceed any bandwidth limit, rather than having VNet1.TC3 consuming it like in ETS-based scheme.

Figure A-8 shows an example of the system configuration, and bandwidth allocation using VNet-Based scheduler configuration scheme.

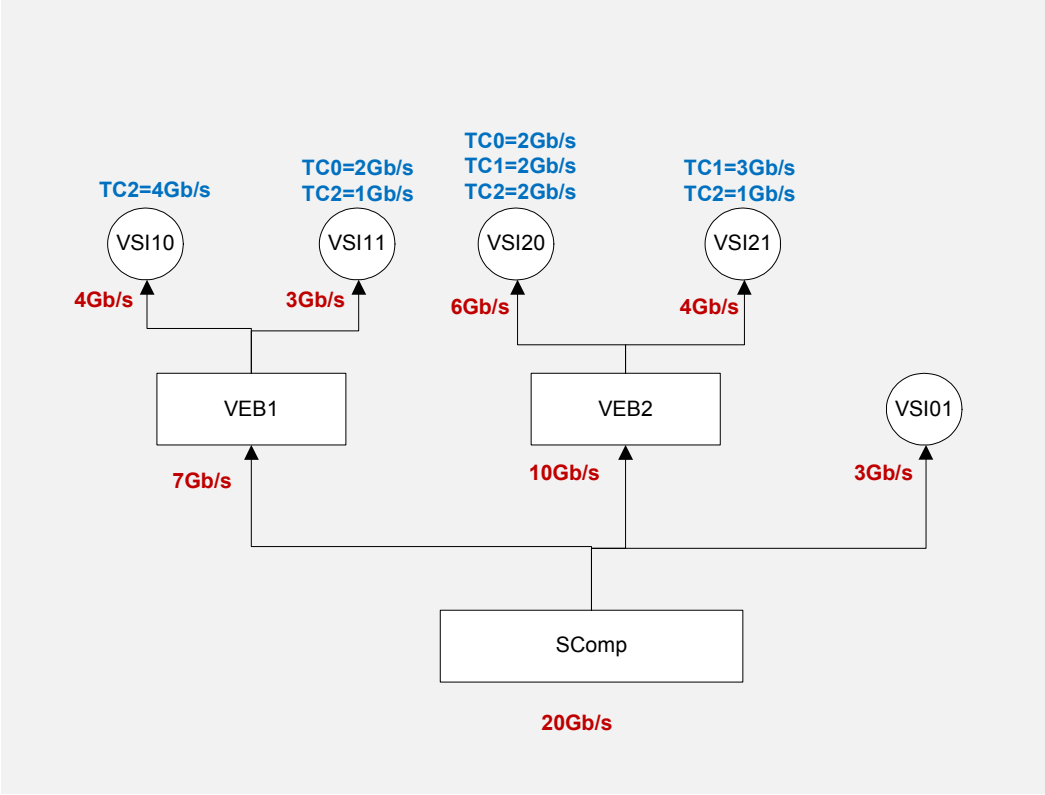


Figure A-8. Example of VNet-based System Configuration

Each of the Switching Components and VSIs has an allocated bandwidth, and may have a bandwidth limit associated with. VSIs are also configured with ETS, or per traffic type bandwidth allocation. Bandwidth allocation must be consistent. Amount of bandwidth allocated to the switching component, should be equal to the sum of bandwidth allocated to all VSIs, and amount of bandwidth allocated for VSI must be equal to the sum of bandwidth allocated to all TCs. For example, bandwidth allocated to VSI10 (4Gb/s) and VSI11 (3Gb/s) should be equal to the bandwidth allocated to VEB1 (7Gb/s).

Figure A-9 below shows an ETS-based Scheduling Tree configuration corresponding to the system configuration shown above.

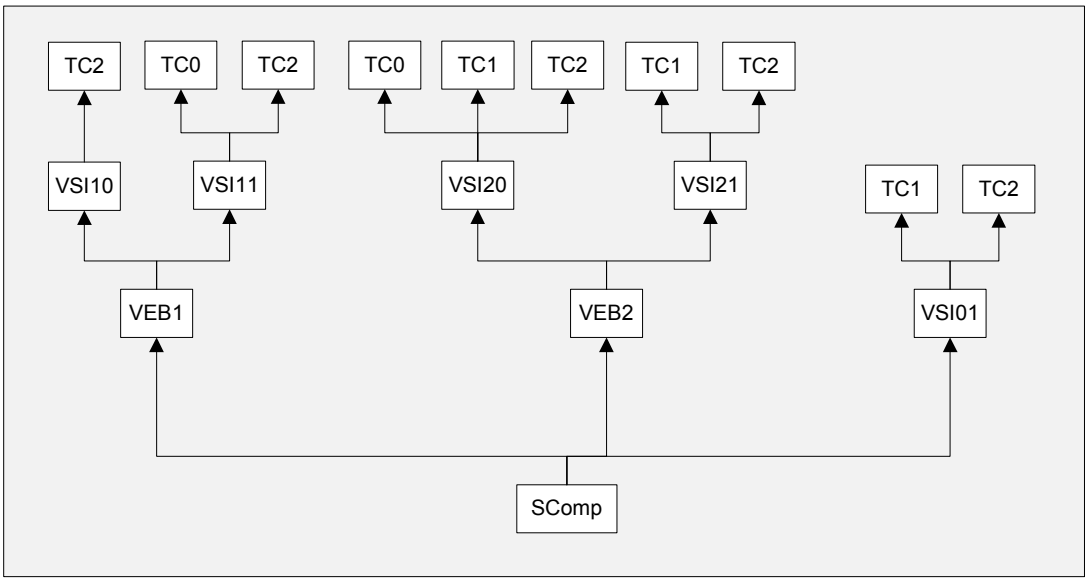


Figure A-9. VNet-Based Scheduling Tree Configuration Example

Structure of this tree exactly follows a topology of internal switching fabrics shown on Figure A-8 . This tree has four levels (one not shown).

- Switching Components and VSI tree level - this is the lowest tree level, programmed with bandwidth allocation for switching components, and VSIs directly attached to the SComp (e.g. Nodes VEB1, VEB2 and VSI01). Each VSI and VEB has a single Node associated with.
- VSI tree level - this is the next tree scheduling level, programmed with bandwidth allocation for VSIs attached to the switching components (e.g. Nodes VSI10, VSI11, VSI20, and VSI21). Each VSI has a single tree Node associated with.
- TC tree level - this is a first tree level that represents VSI ETS. This tree level exists for each VSI that has ETS enabled. This level is configured to distribute bandwidth between Traffic Classes of the same VSI.
- UP tree level - this is the second ETS level. It is not shown on the diagram above. It is used to configure bandwidth allocation between UPs belonging to the same VSI TC.

Similar to ETS-based configuration, software is not aware of internal Scheduler configuration tree structure. Software is using logical representation, shown on Figure A-8 to program scheduler configuration tables. Firmware, translates software AdminQ commands, and performs actual configuration of the scheduler configuration tables.

If software requests to instantiate a bandwidth limit to the Switching Component or VSI, firmware should use a basic bandwidth limits, since each switching component and VSI is represented by single tree Node.

A.2.2 Alternative Configurations

This section gives examples of several alternative configurations that are slight modification of the default configuration described in Appendix A.2.2.

A.2.2.1 VEB Directly Attached to Physical Port

One of possible chip configurations, is omitting S-Comp switching component, and connecting a single VEB or VEPA switching component directly to the physical port.

This configuration simply eliminates one of the tree scheduling levels. Following diagram shows possible tree configurations in ETS-Based and VNet-based schemes,

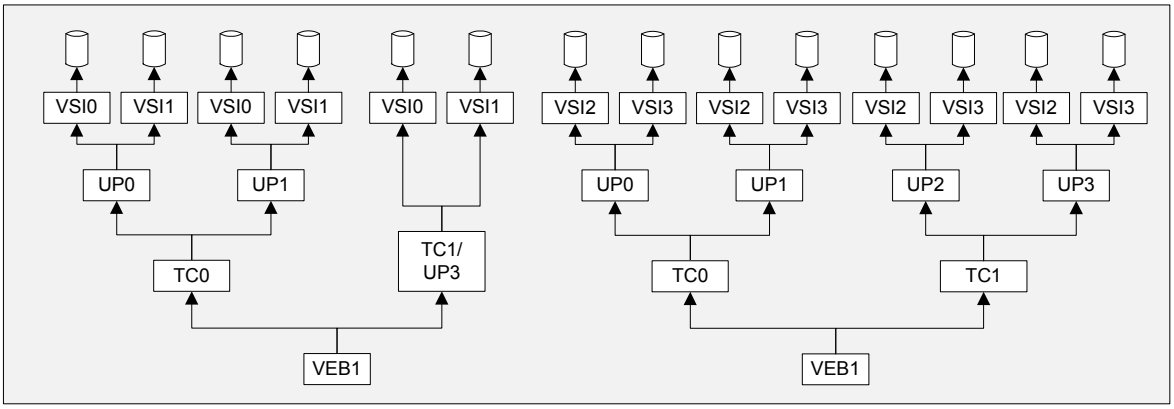


Figure A-10. VEB Direct Attached Scheduler Tree Configuration (ETS-Based)

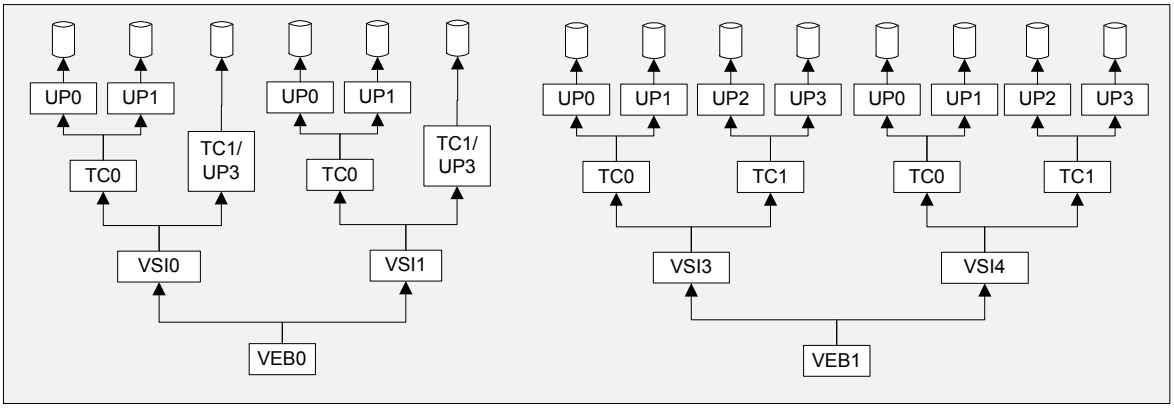


Figure A-11. VEB Direct Attach Scheduler Tree Configuration (VNet-Based)

Diagrams above show an example of the direct VEB attached configuration for two physical ports. Each port has a one VEB attached to it, VEB0 and VEB1 respectively. Each VEB is configured with two VSIs. All VSIs have ETS enabled. Some of the TC Nodes are collapsed since they have only one UP assigned to them (e.g. TC1 on the left side suibtree). Upper diagram shows an ETS-Based tree configuration, and Lower diagram shows a VNet-Based tree configuration. In both configurations a VEB Node takes a place of the root o the tree. The rest of the Nodes are distributed based on the respective bandwidth distribution scheme.

A.2.2.2 Tree Level Extension to Reduce Number of Siblings

Scheduling performance depends on multiple factors, such as depth of the scheduling tree, number of Sibling Nodes, number of active bandwidth limiters.

Scheduling tree might be configured with a large number of Siblings on different tree levels. Here a number of Siblings estimate per tree level:

- S-Comp tree level - 4 Siblings, as a number of S-Comps, or physical ports
- VEB/VEPA/S-Channel tree level - 8 VEB/VEPA Nodes + maximum number of VSIs supported per physical port, or since VEB/VEPA intermediate switches will consume at least one VSI, the number is the maximum number of VSIs.
- VSI tree level - maximum number of VSIs
- TC and UP levels - 8 Siblings per VSI (Upto 8 Leaf Nodes on two levels together)

Two of five tree levels have a potential to have a large number of Siblings Nodes. Scheduler performance depends on the depth of the tree and number of Siblings. Firmware might be required to trade-off a depth of the tree and reduction in number of Siblings Nodes to reduce an impact of the tree structure on the Scheduler performance. This can be done by introducing an intermediate level of “dummy” tree Nodes that do not carry any functionality beyond splitting a Nodes with large number of Children to the two level subtrees.

Diagram below shows an example of such tree modification.

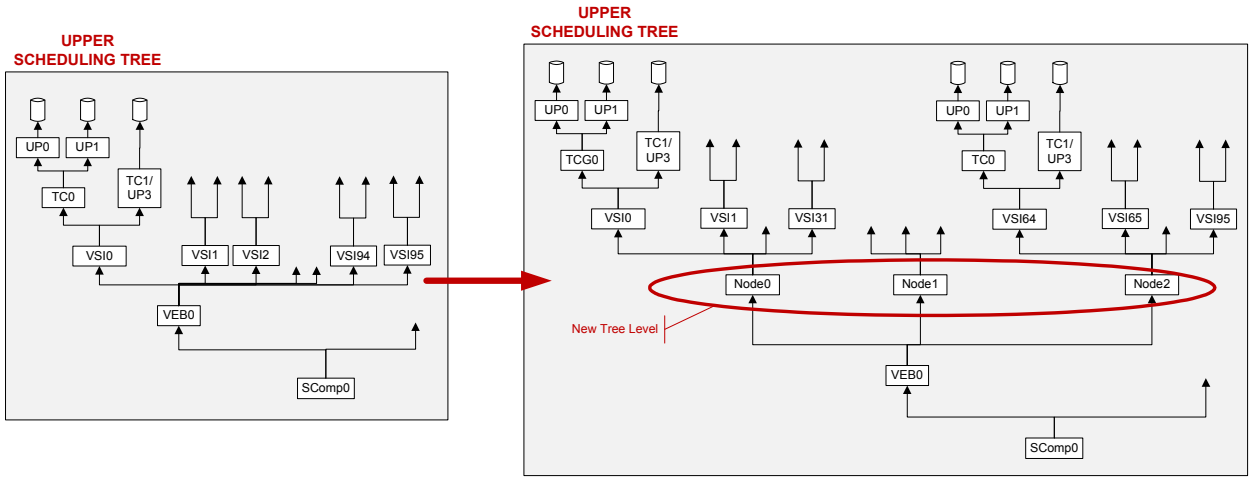


Figure A-12. Sibling Nodes Reduction Example

This diagram shows an example of tree with excessive number of Siblings on the VSI tree level (96 VSIs instantiated for the VEB0) on the left side, and the new tree with reduced number of Siblings on the right side. The new tree has one tree level added - Node0, Node1 and Node2. Those Nodes are made Children of the VEB0 Node, and each of the new Nodes carry 32 Children - VSI Nodes. Bandwidth allocation for those Nodes should be based on the bandwidth allocation for the Child VSI Nodes. New Nodes should not have bandwidth limits assigned to, and arbitration scheme used on this tree level should be Weighted Round Robin.

New tree structure might cause some deviations to the relative bandwidth allocation among VSIs. For example, assuming that all 96 VSIs on the left side tree has identical bandwidth allocation, then if the only VSIs that have work available are VSI0, VSI32 and VSI64-VSI71 (total of 10 VSIs),

- Left side tree configuration - each VSI will get equal bandwidth allocation of 1/10 of bandwidth available for VEB0.
- However in the right hand side tree configuration, bandwidth distribution would be different. Since all VSIs had configured in the original tree with equal bandwidth allocation, in the new tree, Node0, Node1 and Node2 will have even allocation as well. And given same work available assumption, VSI0 and VSI32 will get 1/3 of the VEB bandwidth, and VSI64-VSI71 will get 1/24 of VEB0 bandwidth allocation each.

In spite of deviation in relative bandwidth distribution shown in example above, it is not clear whether in steady state this deviation would be important enough to prevent firmware from rebalancing Scheduling Tree to improve overall scheduler performance.

A.2.2.3 Multiple Queue Sets Directly Attached to the Physical Port

This section describes a non-standard chip configuration that might be important for the future customers. In this configuration the XL710 does not have any internal switching components instantiated. Instead, software would like to allocate large number of stateless Queues, assign those Queues to the different physical ports, and specify relative bandwidth allocation and bandwidth limits to those Queues.

Following diagram shows a possible Scheduling tree configuration:

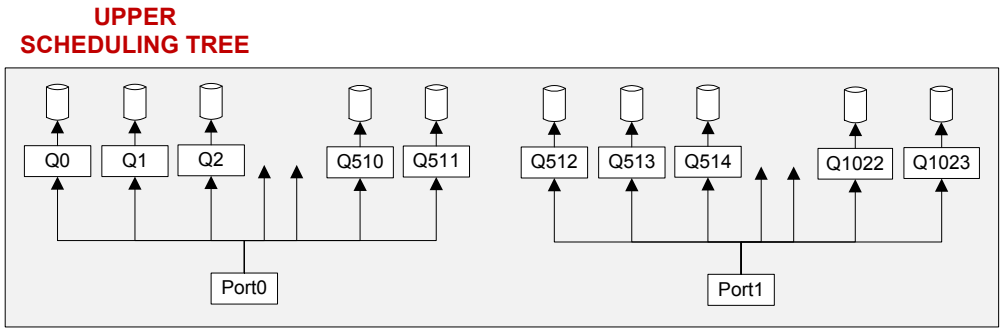


Figure A-13. Multi-Queue Flat Tree Configuration

The XL710 scheduler can support upto 1024 independently schedulable Queues. Each Queue will be associated with a dedicated Queue Set. Scheduler tree structure would be quite flat - two levels. First level is a port arbitration level, and second level is arbitration between Queues belonging to the same physical port. The diagram above shows an example of the dual physical port configuration with 512 Queues assigned to each port. In this configuration Queue tree level has large number of Siblings. To reduce number of Siblings per tree level, a techniques described in the Appendix A.2.2.2 can be used.

Following diagram shows a tree with reduced number of Siblings per tree level.

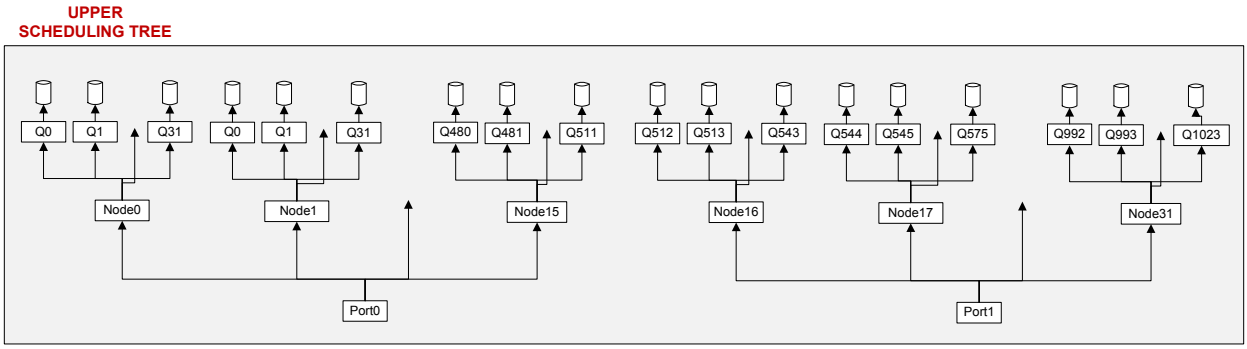
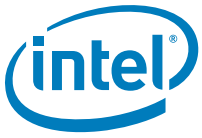


Figure A-14. Siblings Reduction



This diagram shows an alternative Scheduling tree configuration, which has one more tree level added to reduce the number of Sibling Nodes. This new tree level consists of 32 Nodes - Node0-Node31 respectively. Each Node has 32 Children Queue Nodes, which all together adds up to total of 1024 Queues. All new Nodes are configured based on the bandwidth distribution assigned to the their Child Queue Nodes. Reduction of number of Siblings allows better control over the worst case Scheduler performance, and trades it for the additional tree level, and possible deviations in bandwidth distribution as described in Appendix A.2.2.2.

A.3 Scheduler Configuration

This section describes how to configure Scheduling configuration tables, and guides firmware through implementation of the AdminQ commands defined in Chapter 7.8.4.

A.3.1 Overview of Scheduler Configuration Tables

Scheduler is configured using a set of Scheduler configuration tables. Scheduler configuration tables are shared by all PCI functions, and managed by firmware. Scheduler configuration tables are defined in the "Transmit Scheduler" MAS document. In this section we provide a high level description of each table that needs to be configured. Scheduler table configuration is performed by firmware, and software can use AdminQ commands described in Chapter 7.8.4 to alter default scheduler tables configuration.

Each Node in the scheduling tree has an entry in each one of the tables described below, and each Node has the same index for addressing all the scheduler data structures.

A.3.1.1 Node Table

Node Table is a main Scheduler configuration table. This table is used to define a scheduling tree structure. Each entry corresponds to the individual Node in the Scheduling tree. There is technically no root node in the tree, there is a set of nodes that define the base of each tree. These base nodes need to be consecutively located as the first Nodes in the table (Node 0 through Node N).

A Parent Node holds an index of the first Child Node and last Child Node. All Children Nodes of the same Parent (Sibling) should be organized by upto 8 Nodes (aligned to the Node Index). Each Nodes Octet is located continuously in the memory. Node Octets of the Siblings are chained in linked list using Node Octet table.

The node table is organized in groups of 8 entries (0-7, 8-15, etc.) to optimize the sibling arbitration cycles. There can be more than 8 siblings which would require multiple reads of linked Nodes Octets (e.g. 12 siblings will require two reads from the Nodes table, and one read from the Node Octet table, 19 siblings will require three reads from the Nodes table, and two reads from the Nodes Octet table, etc.).

In addition to tree configuration, each Node table entry carries a work available and status information required for the scheduling, such as per Traffic Class work available information, which is based on Children status information for Parent Nodes, or on availability of work in Queue Sets in case of Leaf Node. Indication whether Node can be scheduled based on bandwidth limit and bandwidth allocation credits. Node table entry also used to keep bandwidth allocation credits for both relative and best effort bandwidth allocation schemes.

Number of entries in the Node table is calculated based on the maximum number of Leaf Nodes supported. For the 1024 Leaf Nodes, assuming default Scheduling Tree configuration Nodes table should carry at most:

- 1024 Nodes for UPs or TCs with single UP
- 512 Nodes for TCs that have more than single UP associated with
- 384 Nodes for VSIs
- 16 Nodes for VEB/VEPA switching components
- 4 Nodes for S-Comps
- Total of 1024+512+384+16+4 = 1940 tree Nodes

Note: Maximum number of nodes in tree configuration, assuming collapsing of Nodes with single Child is limited by the binary tree. Number of Nodes in the binary tree that has 1024 Leaf Nodes is: 1024+512+256+128+64+32+16+8+4+2+1 = 2047, therefore 2048 Nodes should be sufficient to implement any tree with 1024 Leaf nodes.

A.3.1.2 Branch Table

This table has same number of entries as a Node table. Each entry in this table carries a list of Nodes that are leading from the Root Node of the tree to the Node. This table is a service table, and it allows traversal of the Scheduling tree in the reverse direction - from the Child Node to the Parent Node. Keeping Branch Nodes in a separate table, rather than using a back-pointer from the Child to Parent allows pipelined backward tree traversal.

Branch table entry must be allocated for every tree Node.

A.3.1.3 Bandwidth Limit Table

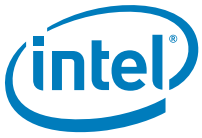
Bandwidth limit table carries a bandwidth limit configuration for each Node. If bandwidth limit is enabled for the Node, the corresponding entry in Bandwidth Limit Table should be initialized with the credit update and the maximum number of bandwidth limit credits that can be accumulated for the Node. The maximum number of credits is specified a number of Quanta worth of credits that can be accumulated. Credits are specified in units of 8B. Once bandwidth limit table entry is configured, corresponding entry in the Node table should be set to indicate that bandwidth limit is enabled for the Node.

If bandwidth limit is not enabled for the Node, then corresponding Bandwidth Limit table entry should be initialized with zeros.

A.3.1.4 Shared Bandwidth Limit table

Shared bandwidth limit table carries accumulated shared bandwidth limit credits. Number of entries in this table is limited to 512. Each entry can be associated with multiple Bandwidth Limit table entries. If bandwidth limit table entry is associated with the shared bandwidth limit, it carries an index of the corresponding entry in the Shared Bandwidth Limit table.

Shared bandwidth limit table carries number of accumulated credits, and the maximum number of credits that can be accumulated by that shared bandwidth limit.



A.3.1.5 Best Effort Table

This table is used to allow best effort scheduling for the Siblings Nodes. Nodes Table entry of the Parent Node carries an indication od the bandwidth allocation scheme used for the Children Nodes. Two options are available: relative bandwidth allocation and best effort or zero-bandwidth guarantee allocation.

- In relative bandwidth allocation, each Node gets a non-zero bandwidth allocated to it, and the remaining bandwidth is shared across Siblings with work available proportionally to the original bandwidth allocation.
- In best-effort or zero-bandwidth allocation scheme, Node might have a zero bandwidth allocated to it. Those Nodes won't be able to use any bandwidth until all Nodes used their non-zero bandwidth allocation, and then remaining bandwidth is equally shared among all Nodes.

If relative bandwidth allocation is configured for the Sibling Nodes, then entry in the zero-bandwidth guarantee table corresponding to the Parent Node should be zeroed.

If best-effort bandwidth allocation is configured for the Sibling Nodes, then Parent Node should be configured with the Total Number of bandwidth allocation credits available for all Children nodes.

A.3.1.6 Ready List Mapping Table

This table specifies mapping of each Queue Set to the corresponding Leaf Node index. This mapping is required to allow unbalanced Scheduling tree configuration and satisfy requirement of continuous location of the Sibling Node Octets in the Scheduling tree.

In addition to the index of the corresponding Leaf Node in the tree, this mapping table carries a TC of that Queue Set, and identification of the PCI Function that owns the Queue Set.

Ready List Mapping table also carries an index of the Branch table entry to be used to traverse predecessor Nodes of the Leaf Node associated with the Ready List Mapping table entry. Firmware can use this field to set an index of the First Child entry in the Branch Table to all Siblings to reduce amount of Branch table updates required for tree restructure.

Note: The XL710 restricts each Queue Set be owned by one PCI Function and carry traffic of single TC.

Note: Ready List Mapping Table is the only table that is not ordered by the Node indexes in the tree.

A.3.1.7 Nodes Octet Linked List Table

To simplify management of the Nodes table, all Nodes table entries are allocated in 8ths. All Nodes located in the same Nodes Octet are continuously located in the memory.

If Parent Node has multiple Child Nodes, these Nodes should be located continuously within an allocated Nodes Octets. If number of Child Node exceed a space available in one Nodes Octet, multiple Nodes Octets can be allocated. First Nodes Octet is continuously occupied by Siblings starting from the First Child Node. First Child Node is recommended to be a first Node in Nodes Octet, but not required. All subsequent Siblings must be continuously located in linked Nodes Octets.

Child Nodes belonging to different Parent Node are allowed to share Nodes Octet, but they must remain continuos with respect to other Child Nodes belonging to the same Parent Node.

Nodes Octet table has 256 entries of 8b each. Each entry is an index of the next Nodes Octet in the chain. Nodes Octet table entry only valid if a corresponding Nodes Octet is included into the Nodes Octet chain or linked list, and is not a last Nodes Octet in the list. Nodes Octet index is calculated by Node Index >> 3. Hardware uses a First and a Last Node Index to identify a first Nodes Octet and a last Nodes Octet in the linked list.

A.3.2 Scheduler Configuration Restrictions

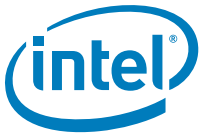
This section describes set of restrictions on configuration of the XL710 Scheduler tables.

- Node indexing in Scheduler Configuration tables
- Majority of the configuration tables keep same Node Index to allow easy access to information related to the same tree Node
- Location of the base tree level Nodes.
- Since tree Root Node does not carry any practical functionality, the zero level Nodes must be continuously located in the beginning of the Scheduler Configuration tables, and occupy entries 0-(N-1), where N is a number of the Nodes on the base tree level.=. If tree Root Node is used to bandwidth limit entire tree, this Node still must be located at the beginning of the Scheduler Configuration table at entry 0.
- Special meaning for the Node0. Node0 cannot be a Leaf Node and cannot be a valid Child Node. Most of the time Node0 would be occupied by the Node corresponding to the Port0. Therefore an index 0 can be used as an invalid Node Index when referring to the Leaf or Child Nodes.
- Location of Sibling Nodes
- Scheduler configuration tables (all except for Branch and Ready List mapping) are organized and managed in Node Octets. Siblings MUST be continuously located within Nodes Octet. If not all Sibling Nodes can fit in the same Octet, multiple Octets can be used. If Siblings consume multiple Node Octets, they must be continuously located within Nodes Octets, and Nodes Octets chained using Node Octet table. First Child Node is not required to be a first Node within Nodes Octet.
-

A.3.3 Scheduler Resources Reduction

There are several steps that allow reduction in the chip resources, in particular memory used by the Scheduler.

- Reduction in number of Stateful Queue Sets.
- POR defines number of Queue Set pairs to match the number of Leaf Nodes. The XL710 supports asymmetric stateful and stateless resource allocation. E.g. number of VFs supporting iWARP is limited to 32, while the total number of VFs is 128. Given this asymmetry, we can change a scheme of the Queue Set resource allocation and reduce number of Queue Sets available for the stateful Queues to 256. This change should not require modification of the Scheduler configuration structures, since it is already uses cross reference to associated Leaf Nodes and Queue Sets.
- Reduction in number of Bandwidth Limited Nodes.
- POR defines that each tree Node might have a bandwidth limit enabled. This requires bandwidth limit table supporting max number of tree Nodes. Alternative approach would be to limit the number of tree Nodes that can be bandwidth limited to 256, and add cross reference between Nodes table and bandwidth limit. This will allow to reduce the size of the Bandwidth Limit table, and relax performance requirements of the bandwidth limit update logic.



- Reduction in number of Best Effort Sibling sets.
- POR defines that each Node might be a Parent Node with Children Nodes scheduled using best-effort bandwidth allocation scheme. Alternative approach would be to reduce number of such Node to 256, and add a cross reference to the Node table, referring to the corresponding entry in the Best Effort table. This will allow to reduce the size of the Best Effort table.
- Elimination of the Branch Table.
- Branch table allows fast retrieval of the tree branch Nodes, and it is used to speed up updates that are done independently from the scheduling flow. Such as work available, suspend, bandwidth limit, credits return, etc. Instead of using Branch table, Node table entry can carry a back reference to the parent Node. This will allow backward branch traversal, but will serialize update operation and potentially cause performance degradation for the scheduler.

A.3.4 Firmware Scheduler Interface

Firmware uses a register-based interface to access and program 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 pci 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 upto software and firmware to coordinate their access to the scheduler programming interface.

Scheduler provides two types of accesses to its configuration structures:

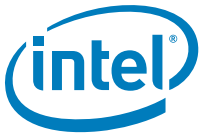
- Immediate Access - a basic operation allowing to read and write one entry of the scheduler configuration tables. Usually firmware will perform one table access in a time using this interface.
- Batched Access - a more advanced operation mode, allowing to firmware to post multiple commands and have them executed as a single atomic operation. Scope of commands that can be used for batch operation is more limited than immediate commands, and primarily includes operations of copy of already existing entry to the new location, or update a field in the table entry.

Firmware is allowed concurrently use both interfaces. Scheduler does not guarantee ordering between Batched and Immediate commands, and executes them in round-robin sequence along with other scheduling operations.

Table below lists all scheduler interface commands supported via Immediate or Batched interface. Majority of commands can be posted via either one of interface. Any invalid command leads to the critical error and suspends a respective interface. Detected errors are reported in TSCDIFSTATUS register. Interface can be enabled back by setting a ICMDCLRERR or BCMDCLRERR bit in the TSCDIFCTRL register.

Table A-1 Scheduler Interface Commands

Command Name	Command Attributes (fields in registers)		Description
	IFICMDL IFBCMDL	**IFICMDH IFBCMDH**	
WriteField	TBLTYPE, TBLENTRYIDX	FLDOFFS, FLDSIZE, VALUE	Change a value of the specified table entry. Firmware provides a table name, index of the entry in the table, field offset within the entry in bits, size of the field and value to be written to the field. This command can be used both as an immediate or batch command.
WriteEntry	TBLTYPE, TBLENTRYIDX	TBLTYPE, TBLENTRYIDX	Write to the specified table entry with the data from the TSCDIFDATA register. Scheduler interface provides only one data register shared for read and write operation as well as for immediate and batch commands. Firmware provides name of the table, and index of the entry in the table. This command is mostly used for immediate operations, unless firmware chooses to update multiple table entries with the same data.
ReadEntry	TBLTYPE, TBLENTRYIDX	TBLTYPE, TBLENTRYIDX	Read from the specified table entry to the TSCDIFDATA register. Scheduler interface provides only one data register TSCDIFDATA shared for read and write operation. Scheduler interface provides only one data register shared for read and write operation as well as for immediate and batch commands. Firmware provides name of the table, and an index of the field in the table. This command is usually used for immediate operation only.
CopyEntries	TBLTYPE, TBLENTRYIDX	NUMENTS, ENTRYIDX	Copy specified number of entries from source table entry index in the table, to the target entry index in the table. The maximum number of entries that can be copied with single operation is 8. Copy operation should not cross Node Octets both on source and target. If firmware attempts to copy more than 8 entries, or copies entries crossing Node Octet boundary, it should use multiple Copy Entries commands to accomplish that. Note that copying multiple entries in one command is allowed only in Node Table and BW limit Table. For all other table types, NUMENTS must be set to “1”. Source and destination are not allowed to overlap. Firmware provides a table name, indexes of source and target tables entries, and number of entries to copy. This command is mostly used for Batched operation, but can be used for Immediate operation as well.
CopyField	TBLTYPE, TBLENTRYIDX	FLDOFFS, FLDSZ, ENTRYIDX	Copy a specified field from the source table entry to the target table entry. Firmware provides a table name, indexes of the source and the target table entries, field offset within an entry in bits, and size of the field. This command is mostly used for Batched operation, but can be used for Immediate operation as well.



Command Name	Command Attributes (fields in registers)		Description
	IFICMDL IFBCMDL	**IFICMDH IFBCMDH**	
CopyAndShift	TBLTYPE, TBLENTRYIDX	ENTRYIDX	This command applies to Branch Table only. It allows to copy a source entry to the target entry, and shift a content of the target entry by one field. I.e. predecessor0 in source entry becomes predecessor1 in the target entry, etc. This operation is useful when creating a child Branch Table entry from the Parent Branch table entry. Firmware provides indexes of the source and target Branch table entries. This command is mostly used for Batched operation, but can be used for Immediate operation as well.
Control	CTRLTYPE, TBLTYPE, TBLENTRYIDX	ENTRYIDX	Controls Batch interface. Currently the only supported Control commands are a BatchDone,Node Suspend and Node Resume commands. BatchDone command completes Batch, and allows scheduler to switch to perform other scheduling flows. This command is used for Batched operation only. Node Suspend/Resume commands allows firmware to suspend/resume specified Node or Ready List. Transmit scheduler will be setting/clearing a Suspend bit in the Nodes table or in Ready List Mapping table entry, depending on the table type specified in the request, and will be updating a Work Available status of the branch Nodes respectively. If Suspended/ Resumed Node is the only Node with Work Available for the particular Traffic Class. Parent Node will have its Work Available status updated, and change propagated to other predecessor Nodes. Node Suspend and Resume commands can be used both as a Batched or Immediate Commands,.

Submission of Immediate and Batched commands is performed using TSCDIFICMDH/TSCDIFBCMDH and TSCDIFICMDL/TSCDIFBCMDL registers. Only few commands are using a TSCDIFICMDH/ TSCDIFBCMDH registers (WriteField, CopyField and CopyEntries). If command does not use TSCDIFICMDH/TSCDIFBCMDH register, it can be posted just by writing to TSCDIFICMDL/TSCDIFBCMDL register. Otherwise, firmware write to TSCDIFICMDH/TSCDIFBCMDH register first, and follow it with a write to TSCDIFICMDL/TSCDIFBCMDL register. Write to low register indicates to hardware that command is ready and can be processed.

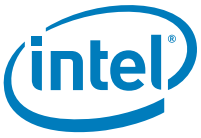
Both Immediate and Batched interface support limited number of outstanding commands (Immediate - single command, batch interface upto 63 commands). Firmware must use TSCDIFSTATUS register to validate that interface has a free space for the new command. Overflow of immediate or batched interfaces is critical error, reported in TSCDIFSTATUS register and leads to suspension of the interface.

A.3.4.1 Immediate Command Interface

Immediate Command interface is provided by TSCDIFICMD, TSCDIFDATA and TSCDIFSTATUS registers. This interface is mostly used for WriteEntry and ReadEntry commands.

Firmware flow will usually include:

- Read TSCDIFSTATUS register to make sure that no Immediate Command is pending.
- This operation can be skipped if previously posted immediate operation has been completed.
- Write data to the TSCDIFDATA register (for WriteEntry operations). Data should be written to the lower words of the TSCDIFDATA register. Order of writing data is not important.



- Write command to TSCDIFICMDH/L registers. Only few commands are using TSCDIFICMDH register, if posting different command, firmware can skip writing to TSCDIFICMDH register, and fill TSCDIFICMDL register only, content of the TSCDIFICMDH should be ignored then.
- Read data from the TSCDIFDATA register for ReadEntry operations.

Though Immediate Interface is primarily designed for ReadEntry and WriteEntry operations, it can be used by firmware to post other commands, except for Control commands.

A.3.4.2 Batched Command Interface

Batched Command interface allows firmware to post multiple commands that need to be executed in atomic fashion. This interface allows firmware to perform majority of Scheduler Configuration tables without suspending normal scheduler operation flow, except for execution of series of Batched commands, which will consume few scheduling cycles at most.

Batched Command interface allows firmware to post upto 63 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, and Scheduler can switch to perform other scheduling flows. Commands posted to Batched interface are not executed immediately, they are queued to the 63-deep Batch FIFO. Firmware should “ring DB” by writing to TSCDIFCTRL register to request execution of the posted sequence of batched commands.

Here an example of posting Batched command sequence:

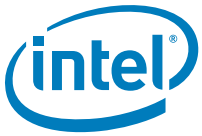
- Read TSCDIFSTATUS register to make sure that batched interface has enough space available.
- Write one or more commands to TSCDIFICMDH/L registers. TSCDIFICMDH register should be used for WriteField, CopyField and CopyEntry commands only.
- Complete sequence with writing BatchDone command to TSCDIFICMDL register
- Write to TSCDIFCTRL 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 BatchDone command. Each time firmware posts BatchDone command to the Batched interface, it must ring DB. Hardware increment an internal counter with each DB ring, and will decrement with each processed BatchDone command. If counter remains positive, hardware will process next batch after it completes a round of performing other scheduling flows.

If number of commands in sequence exceeds the size of the Batch FIFO, firmware can still post that sequence. It will need to ring a DB after posting first 63 commands, without posting BatchDone, keep posting more Batched Commands monitoring number of available batched commands in TSCDIFSTATUS register, and after posting all commands conclude sequence with posting BatchDone. In this case, Scheduler will process all commands in sequence atomically, but due to disparity in scheduler and firmware performance characteristics, such update can impact scheduler performance, since it won’t be processing any other scheduling flows till completion of Batched sequence.

Firmware can use Batched command interface to post sequences of commands that not necessarily require atomicy.



A.3.5 Standard Scheduler Configuration using AdminQueue Commands

This section describes a logical updates of the Scheduler configuration tables required to support standard AdmiQ commands defined in Chapter 7.8.4.

A.3.5.1 Performance Requirements

Performance of scheduler configuration update has two aspects:

- How quickly individual update can be done, or number of updates per second
- How much scheduler configuration update impacts Scheduler performance, or delays scheduling operations.

The first parameter of updates per second greatly depends on the firmware performance, and time it takes to complete entire operation.

The second parameter depends on efficiency of scheduler programming interface. Having batched interface allowing firmware to build a sequence of operations that must be performed atomically with respect to other Scheduling flows, and have those batches executed by the Scheduler in-between scheduling flows greatly reduces impact of the scheduling configuration updates on the overall scheduler performance, and makes this impact independent of the firmware processing speed, as long as batched interface allows sufficiently long atomic sequence updates.

Based on the proposed Scheduler update interface, and validated assumption that it supports sufficiently long atomic sequences, the impact of the scheduling configuration change for the standard scheduler configuration on the overall scheduler performance should be negligible.

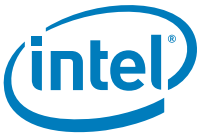
Initial firmware evaluation based on the standard scheduler configuration changes, and Scheduler interface, leads to believe that firmware should be able to sustain a scheduler configuration update rate of 200 updates per second. If for some reason, system will require an update rate beyond firmware capabilities, PF software issuing AdminQ commands will be responsible to pace an update rate to match rate supported by firmware.

A.3.5.2 Resource Allocation Strategy

This section focuses on configuration of Scheduler table depending on the selected Scheduler configuration scheme (ETS-Based or VNet-Based). Majority of resource allocation recommendations are the same for both configuration schemes. When recommendations are different, it is explicitly indicated in the description.

Scheduler configuration tables allow dynamic and flexible resource allocation but they are not sized to provide maximum amount of resources for each switching component. This precludes firmware from using a worst-case resource preallocation strategy. This together with a restrictions on the scheduler tables configuration described in Appendix A.3.2 requires firmware to adapt dynamic resource allocation scheme, described in this section, that might lead to potential subtree or entire tree reconfiguration to condense sparse allocated resources.

Most of the scheduler configuration tables are organized based on the tree Node Index, except for the Ready List Mapping Table, which is organized by the Queue Set index and Shared Bandwidth Limit table. For the allocation of the Ready List Mapping table see Appendix A.3.5.3.1. For the allocation of the Shared bandwidth limit table entry see Appendix A.3.5.4.3.



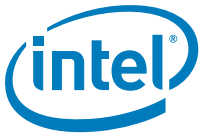
In addition firmware may need to keep some extra tracking structures, such as a free list of Node Octets ordered by the number of free available entries in Octet.

Here set of guidance that firmware should follow when allocating entries in scheduler configuration tables.

- Four lowest entries of the scheduling tree should be used for the switching components directly connected to the chip physical ports, and should allow arbitration between ports. In single port configuration, only entry 0 of the Node Configuration table should be configured, and other three entries should remain invalid. Firmware should preallocate entire Nodes Octet (0).
- VNet-Based Configuration: The XL710 supports upto 16 VEB/VEPA switching components, with maximum of 8 per physical port. Firmware should reserve 4 Nodes Octets - one per port for switching components, or VSIs connected to S-Channels. In ETS-Based configuration, VEB/PA Nodes are distributed across multiple TCs and UPs, and therefore may consume much more tree Nodes.
- VNet-Based Configuration: Due to small number of switching components supported by the XL710 (total of 20), firmware should not mix VSIs belonging to different switching components in the same Nodes Octet.
- VNet-Based Configuration: Due relatively large number of TCs and UPs (Scheduler supports an average of two TCs/UPs per VSI), firmware won't be able to do such simplification for TC Nodes, and will have to support TCs belonging to difference VSIs sharing same Nodes Octet.
- ETS-Based Configuration: TCs, and UPs are the lowest tree level. Total number of combined TC/UP Nodes is 8 per port. Firmware can pre-allocate those Nodes (or even 16 Nodes per port). POR enables single level ETS - TC level only. This reduce number of Nodes to be preallocated per Physical Port to 8.
- When allocating a new Node, firmware should reuse Nodes from already allocated Octets belonging to the Parent Node, if any available. If not new Nodes Octet should be allocated.
- To reduce amount updates of Branch table, firmware should keep upto date only Branch table entries of the first Sibling, and configure both Ready List Mapping table and Rate Limit table refer to the branch entry of the first sibling for all sibling Nodes. This should reduce overall Branch table update cost when adding a tree level, or moving Nodes within Scheduler configuration tables.
- When deallocating Nodes, firmware should not deallocate Branch table entry for the first Sibling of VSI or VEB Nodes Octet.
- Firmware should take advantage of extra Nodes available (25% of available Nodes), Cleanup process can be postponed and performed in the background.
- Batched command interface, described in Appendix A.3.4, should be used to allow atomic updates of the Scheduler configuration tables without suspending scheduler operation.
- Though batched interface allows atomic sequences longer that 63 commands, firmware should avoid using those due to hardware performance penalties. Overhead of use of long Batched sequences should always be compared with overhead of suspending Scheduler operation to perform scheduler configuration tables update.
- Guidance described above do not necessarily apply to non-standard scheduler configuration supported by hardware. Hardware implementation should not assume that firmware will always follow those guidance.

A.3.5.3 Queue Set Management

Firmware is responsible for the Queue Set allocation and management. This section describes Queue Set related firmware flows.



A.3.5.3.1 Queue Set Allocation

Firmware can either use a Ready List Mapping Table or maintain its own private bitmap of allocated Queue Sets. Since a pair of Queue Sets is assigned to the same Leaf Node, both Queue Sets are allocated at the same time, and it is sufficient to use a single bit or single entry in Ready List Mapping Table to represent both.

Though tree Node configuration restricts allocation of the Sibling Nodes to consecutive entries in the Node table, this restriction does not apply to the Queue Set allocation due to indirect referencing.

If firmware chosen to use Ready List Mapping Table, it should use a Node Index 0 - to indicate an invalid Leaf Node, and use it as an indication of the vacant Queue Set. Once Queue Set is allocated, the Leaf Node Index should be updated with the index of the tree Leaf Node.

When allocating Ready List Mapping Table entry, firmware should initialize all table entry fields. See MAS for the table entry content and field description.

A.3.5.3.2 Queue Set Identification

Queue Set is identified by Queue Set handle.

Queue Set handle is provided upon completion of Create VSI, Configure VSI Bandwidth Allocation per Traffic Type, Configure VSI Bandwidth Limit per Traffic Type, or Query VSI Bandwidth Configuration per Traffic Type. To retrieve Queue Set Handle firmware will need to reach a Leaf Node referring to the Ready List Mapping Table. In case of ETS-Based configuration this can be done by walking tree Nodes associated with VSI.

Queue Set handle is 16b value that can carry encrypted index of the Queue Set in the Ready List mapping table.

A.3.5.3.3 Queue Set Deallocation

Queue Set deallocation operation can be caused by AdminQ command, directly or indirectly, or by FLR/VFLR.

Usually Queue Set deallocation is a result of changing configuration of the corresponding VSI including disabling VSI.

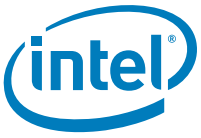
Prior to initiating configuration changes leading to Queue Set deallocation, such as TC allocation to VSIs, or VSI deallocation, software must validate that all QPs associated with Queue Set are destroyed, and cannot lead to work available update targeting this Queue Set. Firmware and hardware are not responsible to truck resource allocation, and trust Physical Function software.

In case of VFLR or FLR hardware will schedule a special token to cleanup the pipeline. Once hardware pipeline cleanup is done, firmware can start its own cleanup of the scheduling table entries owned by resetted VF/PF.

Hardware guaranties that no work will be scheduled to VF/PF under reset.

Deallocation of Queue Set is included into scheduler resources cleanup. Deallocated Queue Set is marked with Leaf Node of 0, and can be reused for other VF/PF.

A.3.5.3.4 Low Latency Queue Set



VNet-Based configuration scheme requires special handling of the low latency traffic. Scheduling tree is structured to distribute bandwidth between switching components and VSIs, and performs TC arbitration only post VSI level. Such structure may cause significant delays to the low latency traffic, since it might take a round via entire tree before VSI is getting scheduled.

To overcome this configuration shortcoming, and provide a priority scheduling for the low latency traffic within bandwidth allocated to the corresponding VSI, the XL710 supports a low latency Queue Sets. Software can choose particular Queue Set to be a low latency Queue Set. This Queue Set should be associated with the low latency (usually strict priority) TC.

Hardware uses a special priority arbitration scheme for such Queue Sets.

- With a Low Latency Queue Set allocation hardware requests Scheduler to preallocate bandwidth allocation credits using standard scheduling tree configuration.
- When work becomes available on the Low Latency Queue Set, and Queue Set has preallocated credits, its getting scheduled bypassing standard scheduling mechanism, and uses preallocated credits
- When amount of preallocated credits falls below configurable threshold, hardware requests allocation of additional credits
- Allocation is credits is done using normal Scheduling flow
- If Low Latency Queue Set runs out of preallocated credits, then it is not scheduled using priority arbitration scheme. This forces Low Latency Queue Set stay within configured bandwidth allocation limits.

Firmware when allocating a Low Latency Queue Set, should set a Low Latency bit in Ready List Mapping table, and post command to the scheduler requesting to preallocate credits for that Queue Set.

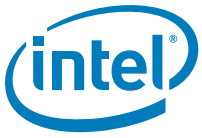
A.3.5.4 Resource Tracking

Software will use VSI SEID and Switching Component SEID to refer to VSIs and Switching Components in AdminQ commands. Respective Switching Configuration table entries will carry an indexes of the corresponding Scheduler Tree Nodes. Firmware is responsible to keep track and locate Tree Nodes associated with UPs and TCS, and in case of ETS configuration. per-TC VSI and Switching Component Nodes.

Following sections discuss resource tracking in each configuration.

A.3.5.4.1 ETS-Based Configuration

- UP and TC Nodes.
- Firmware maintains a lookup table of 16 entries per Physical Port. Entries 0-7 allocated for TCs (TC0-TC7 respectively, entries 8-15 allocated for UPs (UP0-UP7) respectively. Each entry in the table carries an Tree Node Index corresponding to the TC/UP.
- Note, since POR has been reduced to enable single level ETS, firmware is required to track only 8 Nodes per Port, or statically allocate those.
- VEB and VSI Nodes
- Firmware will maintain a lookup table (VSI/VEB/VEPA Lookup table or VVVLT) of 2K entries. Each entry will occupy 16 bits. Total of 4KB of memory. Each entry will carry an index of the next table entry in the chain (lower 11 bits) and a TC particular Node is associated with (high 3 bits). Each VEB and VSI switching entity might have upto 8 tree Nodes allocated for - on Node per TC. A “Next” field of the lookup table entry is used to create a linked list of tree Nodes allocated for the same VSI or VEB/PA. Index of the lookup table entry equals to the Index of the tree Node in the Nodes table.



Switching Configuration Table entry referred by SEID will keep an index of the first lookup table entry corresponding to one of the tree Nodes allocated for respective VSI or VEB/PA. Lookup table will allow firmware to identify all tree Nodes associated with particular switching components without depending on the ETS-Based tree configuration.

A.3.5.4.2 VNet-Based Configuration

- VEB/PA Nodes
- Each VEB/PA has only one Node instantiated in this configuration. Index of this Node is kept in the Switch Configuration table.
- VSI Nodes
- Each VSI has only one Node instantiated in this configuration. Index of this Node is kept in the Switch Configuration table.
- TC/UP Nodes
- TC/UP Nodes are allocated per VSI. Firmware does not keep a record of all TC/UP Nodes allocated per VSI. Whenever firmware needs to modify ETS configuration of VSI, it reads VSI subtree, consisting of TC and UP Nodes allocated for VSI. Firmware does keep a record of UP/TC each tree node is associated with (4b x 2048 = 1KB). This information is used to match a TC/UP indexes with the Tree Node indexes associated with those TCs/UPs. Order of the TC/UP Nodes in the tree is not necessarily matching UP/TC indexes.

A.3.5.4.3 Shared Bandwidth Limit Table

The XL710 supports limited number of Shared Bandwidth Limit accounts - 512. Shared bandwidth limit table entries are allocated on demand, and their indexing is not related to the indexing of the Nodes table. Bandwidth Limit table entry carries an index of the Shared bandwidth limit table entry associated with particular Node, if any. Firmware can keep a bitmap of 512 bits to keep a map of vacant entries in Shared Bandwidth Limit table.

A.3.5.5 Access Validation

Firmware should restrict use of AdminQ command to PF software only.

Firmware also should validate that AdminQ command that accesses/programs resources owned by particular PCI function is issued by the PCI Function that owns the resource, or on behalf of this PCI function.

Firmware should performance validation per switching component, or per virtual port. Firmware should use information kept in the corresponding entry of the Switching Structure Representation table. Switching Structure Representation entry is identified by SEID provided within AdminQ command.

A.3.5.6 VNet-Based Configuration Scheme

A.3.5.6.1 VSI Allocation

Allocation and initial configuration of the Scheduler configuration tables is done as a part of allocation of the new internal switch VSI. Scheduler does not have a dedicated AdminQ command to allocate a VSI. Allocation and update of the scheduling tables is performed as a part of the Internal Switch

configuration command described in Chapter 7.4.9.4.2. New VSI is configured with default number of UPs.

VSI allocation does not require atomic update of the scheduler configuration table and can be done concurrently with running scheduler. Even when Nodes are connected to the Scheduler tree, they have No-Work-Available, and therefore cannot be scheduled. Software can associated QPs with newly allocated Queue Sets only after create VSI call returns.

Perform following scheduling table updates:

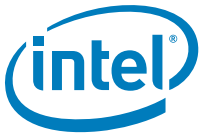
- Allocate a VSI Node in the Node Scheduler table.
  - Use available Nodes in Octet's allocated for the Parent (assuming that order is not important for VSIs, and it is not unless strict priority arbitration will be enabled for VSIs).
  - Do not use Nodes Octet associated with other switching components.
  - If cannot allocate Node, allocate a new Nodes Octet, and chain it to Octet(s) allocated for the Parent Switching component
  - If VSI is a first child of the Switching Component, update VSI Branch Table entry using Parent's Branch table entry and CopyAndShift operation. Note that if switching component is not a first child, then Branch table of the first child among switching component siblings should be used.
- Configure a VSI Node entry
  - Single bandwidth allocation credit, bandwidth limit and best effort should be disabled by default, No Work Available
- Zero out corresponding entries in the Bandwidth Limit and Best Effort tables, and initialize corresponding entry in Branch table to list all Nodes leading from the VSI Node to the tree Root Node. Firmware can use a preconfigured template in respective tables, and CopyEntry operation.
- Allocate number of UP Nodes, as specified AdminQ command.
  - UP Nodes owned by different VSIs can share same Octet.
  - Update Branch table entry of the first child UP Node, using a Branch Table entry of the first child VSI Mode, and CopyAndShift operation.
- Configure each of UP Nodes
  - Single bandwidth allocation credit, bandwidth limit and best effort should be disabled by default, No Work Available
- For each allocated UP Node zero out corresponding entries of the Bandwidth Limit and Best Effort tables, Use preconfigured template, and CopyEntry operation.
- Update VSI Node Table entry to refer to first and last UP Nodes.
- If only one UP was configured fro the VSI, firmware can share same tree Node for VSI and UP.
- Allocate a Queue Set for each UP as described in Appendix A.3.5.3.1.
- For each new allocated Ready List Mapping Table entry, configure it with corresponding TC, and make it refer to the UP Node as a Leaf Node.
- Store VSI Node Index in the switching element entry referred by SEID of the allocated VSI.
- Record Queue Set indexes and return them to software as QS_Handles in AdminQ completion.

A.3.5.6.2 Configure VSI Bandwidth Limit

Software can enable bandwidth limit for the instantiated VSI. See Chapter 7.8.4.5 for the AdminQ command definition.

Following steps should be performed to configure bandwidth limit for the VSI:

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1.
- Retrieve VSI Node Index



- Configure Bandwidth Limit table entry referred by the Node Index using WriteEntry operation.
- Enable bandwidth limit in the respective Node table entry, using WriteField command.

A.3.5.6.3 Configure VSI Bandwidth Allocation

Software can change a default bandwidth allocation for the VSI using AdminQ command described in section Chapter 7.8.4.7.

Bandwidth allocation is relative. Therefore modification of the bandwidth allocation for the particular VSI should be done with respect to all other VSIs of the same switching component. Software that owns switching component, is required to maintain a map of bandwidth allocations, and be responsible for the validity of configuration.

Following steps should be performed to configure bandwidth allocation for the specified VSI.

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1.
- Retrieve VSI Node Index
- Update credits increment field of the Nodes table with specified number of credits, using WriteField command.

A.3.5.6.4 Configure VSI UPs

Software can change number and UPs configured for the VSI. See Chapter 7.8.4.8 for AdminQ command definition.

If software has ETS enabled for the VSI, it must disable it prior to changing UP configuration of the VSI. Firmware should fail UP configuration change otherwise. This requirement allows to simplify firmware flows, and management of the Scheduler configuration tables. Assumption is that change in UP configuration would be rare event, and most of the time would be performed prior ETS is enabled for VSI at first place.

Following steps should be performed to change UP configuration of the VSI:

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1.
- Retrieve VSI Node Index
- Identify Children Node of the VSI Node. Those Nodes should be a UP Nodes.
- Allocate new tree Nodes required for new UPs
  - Firmware can choose reuse already allocated Nodes, though from resource management stand point it seems to be easier to allocate new Nodes and deallocate old Nodes.
  - Allocation and update of new Nodes does not require atomicy, and can be performed using WriteEntry operation.
  - UPs are allowed to share Octet with UPs belonging to different VSIs
- Order all UP Nodes in the tree based on their UP Indexing.
- Program all new allocated Nodes with single bandwidth allocation credit, no bandwidth limit, no best effort and no work available.
- For each new allocated Node zero-out corresponding entries of the Bandwidth Limit and Best Effort tables.
  - For the first UP Node, update a Branch table entry. Firmware can either CopyAndShift entry of the first VSI sibling, or can CopyEntry of the previous first UP.
- Update VSI Node Table entry to refer to first and last UP Nodes.

- For each UP Node allocate an entry in the Ready List Mapping Table, or reuse previously allocated for that UP entry, and update the Leaf Node field.
- Firmware MUST preserve allocation of the Queue Sets to the UP Nodes for the UPs that were kept from the previous configuration.
- Record Queue Set indexes and return them to software as QS_Handles in AdminQ completion.

A.3.5.6.5 Configure VSI ETS/SLA

Software can enable/disable and modify ETS/SLA configuration of the specified VSI. See Chapter 7.8.4.9 for AdminQ command definition.

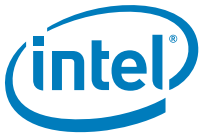
Software MUST UPs allocated for VSI in this command. If software wants to change UP configuration of the VSI it should use Configure VSI UP command, described in Appendix A.3.5.6.4.

To enable and modify ETS/SLA configuration of VSI software MUST specify complete configuration. Software is not allowed to specify incremental modifications.

Changing ETS configuration of VSI may result in changing tree structure of the active scheduling tree, which will require firmware to use atomic sequences of the updates to reduce impact on Scheduler performance.

Following steps should be performed to enable/modify ETS/SLA configuration of the VSI.

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1.
- Retrieve Node Index
- Calculate number of tree levels required for ETS configuration and number of Nodes per level.
  - If ETS configuration has more than one TC enabled, and at least one TC has more than one UP mapped to it, then specified ETS configuration will require two scheduling tree levels.
  - If TC has only one UP associated with, then TC should be collapsed, and one tree Node should be used to represent both TC and UP.
  - Number of the Nodes on the first tree level equals to the number of TCs enabled for ETS.
  - Number of tree Nodes on the second tree level equals to the number UPs mapped to each TC, excluding the case when only one UP is mapped to TC.
- Calculate total number of TC Nodes that need to be added to the tree, (all UP Node are already allocated), and fail operation if Scheduler configuration table does not have enough free Nodes available.
- Note that Ready List Mapping table should have all entries allocated and associated with the corresponding UP Node.
- Allocated UP Nodes are Children of VSI Node.
- Identify UP - to Queue Set mapping of the existing configuration. Mapping of UP to the Queue Set MUST remain unchanged. If UP Nodes is migrated to accommodate scheduling table structure restriction, respective Ready List Mapping table entry should be updated to preserve the mapping.
- Allocate new tree Nodes required for TCs with more than one UP mapped. TC Nodes belonging to different VSI should share Node Octets, and they can share Octets with UP Nodes as well. Nodes used for previous ETS configuration can be either freed or reused depending on the firmware implementation choice.
- For each new allocated TC Node zero-out corresponding entries of the Bandwidth Limit and Best Effort tables. Use pre-configured template Node and CopyEntry operation.
- Update a branch table entry for the first TC Node. Use branch entry of the first VSI child Node, and CopyAndShift operation.
- Update branch table entries of all first UP Nodes using corresponding first TC branch table entry, and CopyAndShift operation.



- . Update VSI Node table entry to refer to the fist and last TC Node allocated for it.
- If Total TC Credits in AdminQ command is not zero, allocate entry in Best Effort table corresponding to the VSI Node, and update this entry with the Total TC credits. This indicates that provided configuration is SLA, and bandwidth allocation between TCs is best effort.
- If any TC was marked as a strict priority, order allocated TC Nodes within the Nodes table to have strict priority TCs coming first, ordered accordingly to their priority (TC7-TC0), followed by weighted round robin (WRR) Nodes, and update WSP to WRR switch point field of the VSI Node table entry with the index of the first WRR Node. If all Nodes are SWP, then set switching point to 0. If all Nodes are WRR, set switching point to the index of the first Child Node.
- Allocated UP Nodes are identified using First and Last Child indexes fields of VSI Node. UP Indexes of allocated UP Nodes are identified using lookup table maintained by firmware.
- If any of TC Nodes is a Leaf Node (case when TC has only one UP mapped to it), update a corresponding entry in the Ready List Mapping Table (based on UP) and make the Leaf Node field refer to the new Node. Release previously allocated UP Node.
- For all TC Nodes that are not Leaf Nodes update references to the first and last UP Nodes.
- Update bandwidth allocation credits in the Nodes table for all TC Nodes, including collapsed TC/UP Nodes.
- If TC has a non-zero Total UP Credits in AdminQ command, allocate entry in Best Effort table corresponding to this TC Node, and update this entry with the Total Number of UP Credits. This indicates that provided configuration is SLA, and bandwidth allocation between UPs within that TC is best effort.
- If any UP was marked as a strict priority, order allocated UP Nodes within the Nodes table to have strict priority UPs coming first, ordered accordingly to their priority (UP7-UP0), followed by weighted round robin (WRR) UP Nodes, and update WSP to WRR switch point field of the TC Node table entry with the index of the first WRR Node. If all Nodes are SWP, then set switching point to 0. If all Nodes are WRR, set switching point to the index of the first Child Node
- Update bandwidth allocation credits in the Nodes table for all UP Nodes.

A.3.5.6.6 Configure VSI UP/TC Bandwidth Limit

Software can configure bandwidth limit to each component of ETS/SLA configuration: i.e. TCs and UPs participating in ETS/SLA configuration. See Chapter 7.8.4.7 for AdminQ command definition.

Software MUST maintain a map of current ETS configuration, and bandwidth limits can be configured only for enabled ETS/SLA configuration components. Attempt to configure bandwidth limit for disabled ETS/SLA component, should fail.

Software MUST use TC and UP handles while referring to the TCs and UPs.

Software cannot provide an incremental updates to the ETS bandwidth limit configuration. Bandwidth limit of 0 is used to disable bandwidth limit.

Following steps should be performed to configure ETS/SLA bandwidth limit for the VSI.

- UP and TC Nodes are contained within VSI Node subtree. Firmware identifies those Nodes by reading corresponding Nodes Table entries.
  - — First Children Nodes of VSI using First and Last Node Indexes to identify TC Nodes,
  - — Then for each identified TC Node, firmware reads its Children Nodes, if any available to identify UP Nodes.
  - — Node type and UP/TC Index is identified using a lookup table maintained by firmware.
- Software should provide bandwidth limits only for the previously enabled UPs and TCs.

- Update respective entries of the Bandwidth Limit table using WriteEntry operation. Bandwidth Limit table can be updated with immediate command, Nodes table should be updated as atomic operation.
- Enable bandwidth limit in the respective Node table entry, using WriteField operation.

A.3.5.6.7 Query VSI Bandwidth Configuration

Software can query for the current bandwidth configuration of the VSI. See Chapter 7.8.4.15 for the AdminQ command definition.

Since this operation does not change scheduler table structure it can be done with immediate ReadEntry commands without impacting scheduler performance.

Following steps should be performed to retrieve current bandwidth configuration of the specified VSI.

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1.
- Retrieve VSI Node Index
- Walk Node subtree, identify UP Nodes (Leaf Nodes), and provide a bitmap of valid UPs. UP Index for each identified UP Node can be retrieved from the lookup table maintained by firmware..
- Retrieve bandwidth allocation credits from the Nodes table entry
- If Nodes Table entry indicates that bandwidth limit is enabled for the Node, retrieve bandwidth limit increment and max credits from the corresponding Bandwidth Limit table.

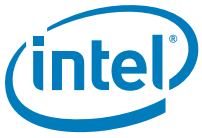
A.3.5.6.8 Query VSI ETS/SLA Configuration

Software can query for the current ETS/SLA configuration of the specified VSI. See Chapter 7.8.4.16 for AdminQ command definition.

Since this operation does not change scheduler table structure it can be done with immediate ReadEntry commands without impacting scheduler performance.

Following steps should be performed to retrieve current ETS/SLA configuration of the specified VSI.

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1.
- Retrieve VSI Node Index
- Read VSI Node from Nodes table, using ReadEntry operation.
- Check if best effort enabled, and retrieve TC Total credits value from the corresponding Best Effort table entry.
- Use First Child Node Index and Last Child Node Index to identify TC Nodes.
- Use WSP to WRR switch point to identify TC Nodes that are configured to strict priority
- Read TC Nodes and retrieve TC bandwidth allocation for each Node
- For each TC Node check whether bandwidth limit is enabled, and retrieve bandwidth limit and bandwidth limit max values from the Bandwidth Limit table
- For each TC that is not Leaf Node
  - Check if best effort enabled for TC, and retrieve UP Total credits from the Best Effort table
  - Use First Child Node Index and a Last Child Node Index to identify UP Nodes associated with TC
  - Use reference to Ready List Mapping Table and a TC value in the respective entry to retrieve TC value.
  - Use WSP to WRR switch point to identify UP Nodes that are configured to strict priority
  - Read UP Nodes and retrieve UP bandwidth allocation for each Node



— For each UP Node check whether bandwidth limit is enabled, and retrieve bandwidth limit and bandwidth limit max values from the Bandwidth Limit table

A.3.5.6.9 Switching Component Allocation

Allocation and initial configuration of the Scheduler configuration tables is done upon allocation of the new internal switching component. Scheduler does not have a dedicated AdminQ command for that, and update of the scheduling tables is performed as a part of the Internal Switch configuration command described in Chapter 7.4.9.4.2.

Switching component can be added instead of already allocated VSI. In that case, VSI Node, including scheduling subtree built on top of it, should be migrated and become a first VSI of the new instantiated switching component.Since this operation involves moving already configured scheduling tree components, it should be performed using series of atomic operations. In that case new switching component should take entry in the table originally occupied by VSI Node, and VSI Node should be moved to the new location. In addition to moving VSI Node, all descendant Nodes that are first siblings, including VSI node itself must have branch table entries updated.

Perform following scheduling table updates:

- Allocate a switching component Node in the Node Scheduler table. Due to small number of supported switching components, they should be allocated from the reserved Node Octets.
- Configure a Node entry with a single bandwidth credit, no bandwidth limit, to best effort and no work available.
- Zero out corresponding entries in the Bandwidth Limit and Best Effort tables, using preconfigured template Node and CopyEntry operations.
- If new added switching component is a first sibling, then initialize corresponding entry in Branch table.
- Store switching component Node Index in the switching element entry referred by SEID of the allocated switching component.

A.3.5.6.10 Configure Switching Component Bandwidth Limit

Software can enable bandwidth limit for the instantiated switching component. See Chapter 7.8.4.9 for the AdminQ command definition.

Following steps should be performed to configure bandwidth limit for the switching component:

- Use SEID to locate switching component in the Switching Structure Representation table, Chapter 7.4.9.1.
- Retrieve Node Index
- Configure Bandwidth Limit table entries referred by the Node Index, using WriteEntry commands. Bandwidth Limit table can be updated with immediate command, but Nodes table should be updated as atomic operation.
- Enable bandwidth limit in the respective Node table entry, using WriteField command

A.3.5.6.11 Configure Switching Component Bandwidth Allocation

Software can change a default bandwidth allocation for the switching component using this AdminQ command. See section Chapter 7.8.4.9 for AdminQ command definition.

Bandwidth allocation is relative. Therefore modification of the bandwidth allocation for the particular switching component should be done with respect to all other virtual ports or switching components sharing same uplink switching component (e.g. VEBs and S-Channels connected to the same S-Comp). Software that owns switching component, is required to maintain a map of bandwidth allocation, and responsible for the validity of configuration.

Following steps should be performed to configure bandwidth allocation for the specified switching component.

- Use SEID to locate switching component in the Switching Structure Representation table, Chapter 7.4.9.1.
- Retrieve Node Index
- Update credits increment field of the Nodes table with specified number of credits using WriteField command

A.3.5.6.12 Configure Switching Component ETS

This command cannot be used in VPN-Based configuration scheme.

A.3.5.6.13 Configure Switching Component UP/TC Bandwidth Limit

This command cannot be used in VPN-Based configuration scheme.

A.3.5.6.14 Query Switching Component Bandwidth Configuration

Software can query for the current bandwidth configuration of the switching component. See Chapter 7.8.4.17 for the AdminQ command definition.

Since this operation does not change scheduler table structure it can be done with immediate ReadEntry commands without impacting scheduler performance.

Following steps should be performed to retrieve current bandwidth configuration of the specified switching component.

- Use SEID to locate switching component in the Switching Structure Representation table, Chapter 7.4.9.1.
- Retrieve Node Index
- Retrieve bandwidth allocation credits from the Nodes table entry using ReadEntry operation.
- If Nodes Table entry indicates that bandwidth limit is enabled for the Node, retrieve bandwidth limit increment and max credits from the corresponding Bandwidth Limit table, using ReadEntry operation.

A.3.5.6.15 Query Switching Component ETS Configuration

This command cannot be used in VPN-Based configuration scheme.

A.3.5.7 ETS-Based Configuration Scheme

A.3.5.7.1 VSI Allocation

VSI Allocation in ETS-based environment is more complex process. It involves allocation of upto 8 VSI Nodes (one per TC), and adding each Node to the respective Parent VEB Node per TC. All allocated VSI Nodes would be instantiation of the same VSI per TC.

AdminQ provides firmware with an SEID of the uplink switching component, and a bitmap of TCs enabled for VSI. TCs enabled for VSI must be a subset of TCs enabled for Switching Component. In case of SComp, enabled TCs are defined by Physical Port ETS configuration. In case of VEB/PA, TCs are enabled either upon VEB/PA creation, or configured using Configure Switching Component Bandwidth Allocation or Bandwidth Limit per Traffic Type AQ command.

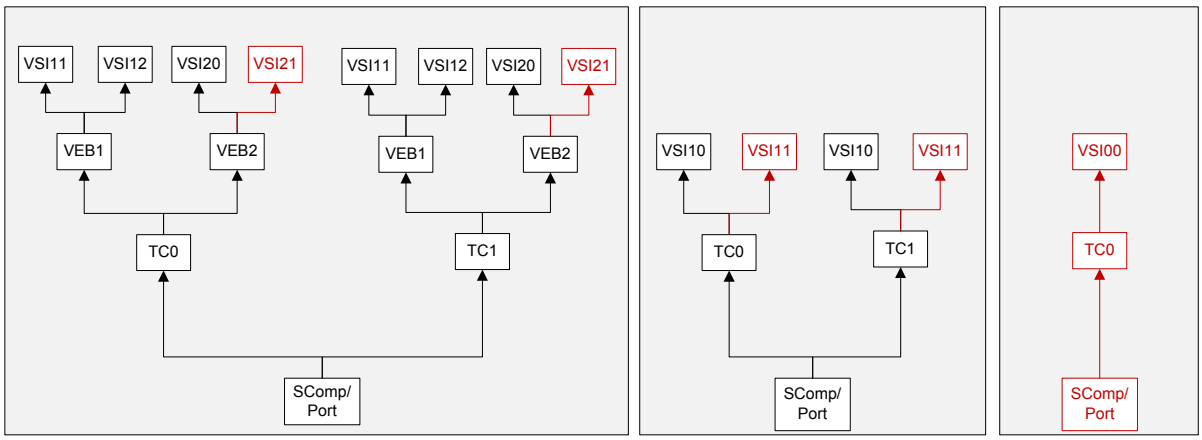


Figure A-15. VSI Allocation Cases

Figure A-15 above shows all three VSI Allocation cases. New Nodes and connections are shown in red. Left side diagram shows adding VSI to VEB/PA. Middle diagram shows adding VSI to the SComp. Right side shows a default VSI allocation for Physical Port.

Sequence of scheduler table updates is similar to those described for the VNet-Based configuration scheme, and omitted here.

In ETS-Based configuration, VSI Nodes are usually tree Leaf Nodes, and therefore their allocation results in allocation of the Queue Sets.

Perform following scheduling table updates:

- Allocate VSI Nodes in the Node Scheduler table.
  - — One Node for each enabled TC
  - — Use SEID of the parent switching component to identify a VSI/VEB/PA lookup table index kept in Switch Configuration table
  - — Identify a Parent Tree Node per TC for each VSI node, using VSI/VEB/PA lookup table. All tree Nodes allocated for the parent Switching Component are chained in this table, and each entry

carries an TC associated with. If parent switching component is SComp, VSI should use TC Nodes as a Parent Nodes (middle case on the Figure).
  - Update VSI/VEB/PA lookup table entries corresponding to VSI Nodes allocated per TC, creating a linked list.
  - Record an VSI/VEB/PA lookup table index of the first VSI Node in the chain in Switching Configuration table
- Update Parent VEB Nodes or TC Nodes in case when VSI is connected directly to SComp or Physical Port.
- Configure all allocated per TC VSI Node entries
  - Single bandwidth allocation credit, bandwidth limit and best effort should be disabled by default, No Work Available
- Zero out corresponding entries in the Bandwidth Limit and Best Effort tables, and initialize corresponding entry in Branch table to list all Nodes leading from the VSI Node to the tree Root Node. Firmware can use a preconfigured template in respective tables, and CopyEntry operation.
- Allocate a Queue Set for each VSI per TC as described in Appendix A.3.5.3.1.
- For each new allocated Ready List Mapping Table entry, configure it with corresponding TC, and make it refer to the respective VSI Node as a Leaf Node.
- Record Queue Set indexes and return them to software as QS_Handles in AdminQ completion.

A.3.5.7.2 Default VSI Allocation

Default VSI is allocated for each Physical Port in SFP configuration or for each Physical Function enable for the Physical Port in MFP Mode. Firmware should follow a VSI initialization sequence described in Appendix A.3.5.6.1 assuming TC0 enabled.

Default VSI is configured with default bandwidth allocation of single credit, with no bandwidth limit enabled.

Software will need to use Configure VSI bandwidth Allocation per Traffic Type command to enable other TCs for the default VSI.

A.3.5.7.3 Changing DCB Configuration

If DCB is not enabled for the Physical Port, all VSIs and VEB/PAs are allocated with TC0 enabled. DCB configuration can be enabled or changed at any point. If DCB is enabled or modified after one or more switching components and VSI were allocated to the Physical Port, those components remain configured with prior DCB configuration (including No-DCB), and firmware updates a tree by adding or removing TC Nodes accordingly to the new DCB configuraiton. It is software responsibility to configure enabled TCs and per TC bandwidth configuration of all VSIs and VEB/PAs.

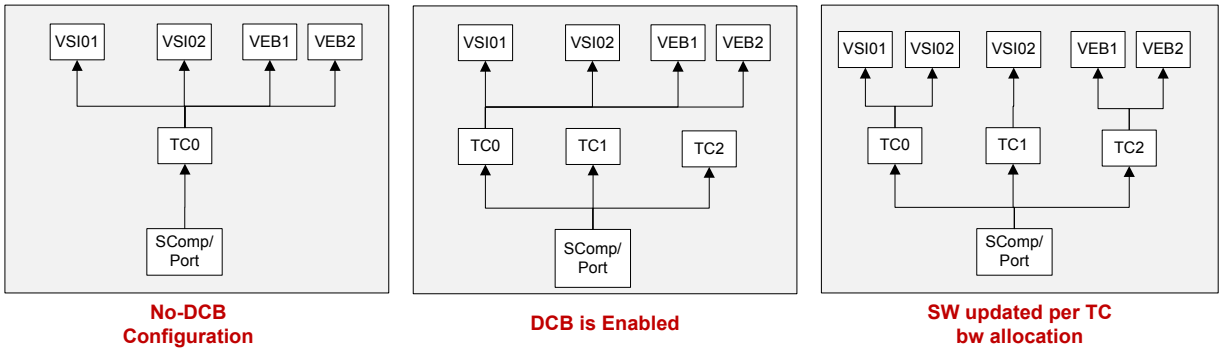


Figure A-16. Enabling DCB

Figure A-16 shows an example of tree transition as a part of DCB enable flow. Left side shows a configuration with multiple VSIs and VEBs allocated for the SComp in No-DCB configuration. Middle diagram shows new TC Nodes added by firmware as a result of DCB enabled fro the Physical Port. Right side shows a tree configuration after software modified TCs enabled and per TC bandwidth allocation for the allocated VSIs and VEB/PAs.

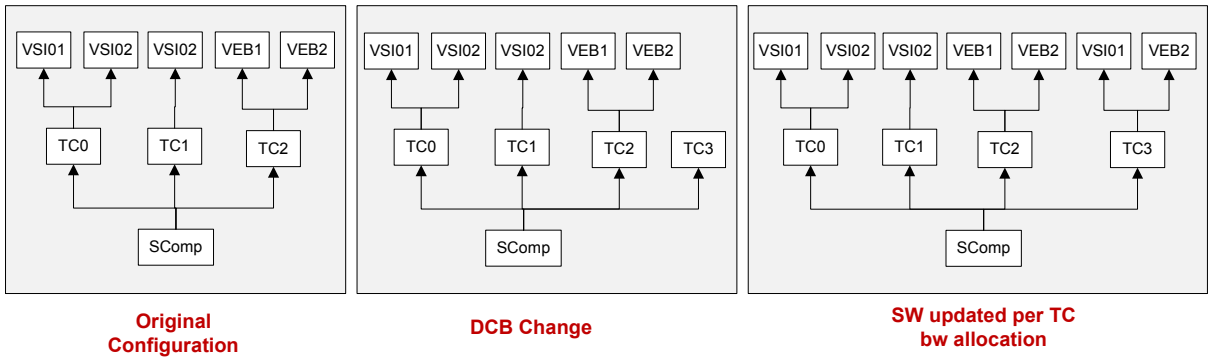


Figure A-17. Increasing number of TCs

Figure A-17 shows an example of tree transition as a part of DCB configuration change - number of TCs increased. Left side shows an original configuration with various VSIs and VEBs configured with different TCs. Middle figure shows a DCB configuration change by adding TC3. Right side shows a new tree configuration after software reconfigured VSIs, VEB/PAs with new set of TCs.

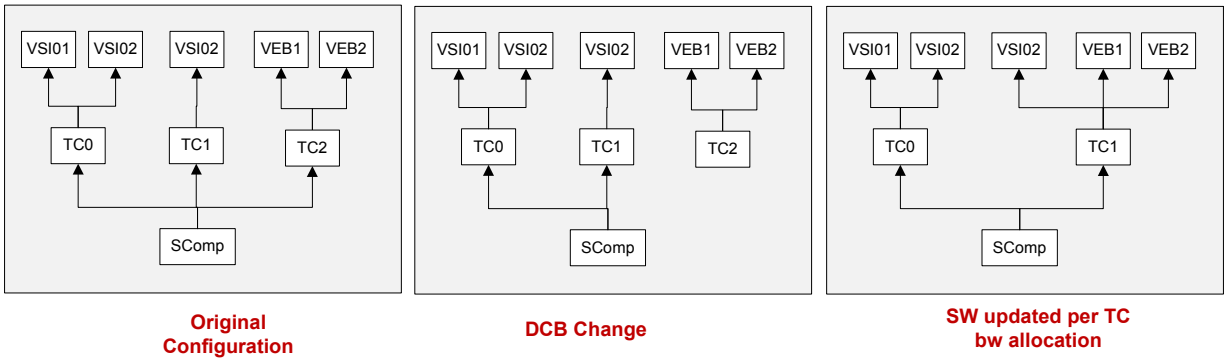


Figure A-18. Decreasing number of TCs

Figure A-18 shows an example of tree transition as a part of DCB configuration change - number of TCs decreased. Left side shows an original configuration with various VSIs and VEBs configured with different TCs. Middle figure shows a DCB configuration change by removing TC2. Note that in this tree, transmit scheduler can keep scheduling work on unaffected TCs. No work would be scheduled for any Queue Set associated with TC2. When removing TC2 from the scheduling tree, firmware should request hardware to "Suspend" that TC2 Node. DCB configuration change might not come synchronised with software managing bandwidth and TC for VEBs and VSIs. Firmware should allow updates of the subtree of TC disconnected due to DCB configuration change. Once software is notified about DCB change by LLDP firmware agent, software is responsible to adjust TC allocation for VSIs and VEBs to match ETS configuration of the physical port. Right side shows a new tree configuration after software reconfigured VSIs, VEB/PAs with a new set of TCs.

A.3.5.7.4 Configure VSI Bandwidth Limit

Single VSI system entity can be represented by multiple Tree Nodes, depending on the number of TCs enabled for VSI. If VSI has only one TC enabled, then the process of enabling Bandwidth Limit is similar to one described in Appendix A.3.5.6.2.

If VSI has multiple TCs enabled, then multiple VSI Tree Nodes will be instantiated. In that case, firmware will have to instantiate a Shared Rate Limit, and configure both Private and Shared Bandwidth Limit tables.

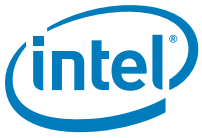
Total bandwidth limit configured for VSI should be distributed between Private Bandwidth Limit accounts (table entries).

Firmware can use one of two options in distributing credits:

— Even distribution among all Nodes
— Uneven distribution, relative to the bandwidth allocation credits.

Following steps should be performed to configure bandwidth limit for the VSI in case of multiple TCs enabled for VSI:

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1. This entry refers to the first entry in VSI/VEB/PA lookup table.
- Walk the linked list of VSI/VEB/PA lookup table entries, and identify all tree Nodes allocate for VSI.
- Allocate and configure Shared Bandwidth Limit table entry using WriteEntry operation



- Calculate private credits for each VSI Node using one of the algorithms described above
- Configure Bandwidth Limit table entry referred by the Node Index using WriteEntry operation.
- Enable bandwidth limit in the respective Node table entry, using WriteField command.

A.3.5.7.5 Configure VSI Bandwidth Allocation per Traffic Type

Software can configure VSI bandwidth allocation within each type of traffic. See Chapter 7.8.4.8 for AdminQ command definition.

In ETS-Based configuration each VSI does not have its own per-say independent ETS configuration. VSI ETS configuration is derived from the physical port ETS configuration, by hierarchically distributing per-TC/UP bandwidth between VEBs and their VSIs.

Change in VSI per traffic type bandwidth allocation does not impact ETS configuration of physical port. It only impacts VSI relative bandwidth allocation among other VSIs belonging to the same VEB within particular TC. Relative bandwidth calculation and credits allocation is responsibility of software.

If VSI had a bandwidth limit enabled prior to changing number of TCs enabled for VSI, firmware is responsible to reprogram Private and Shared Bandwidth limit table entries corresponding to the per TC VSI Nodes. See details below

- Firmware will need to read a bandwidth limit table entries for all TCs that were previously enabled for VSI
- Calculate total amount of Credits increments that were allocated to all Nodes together
- Recalculate a new credits increment given the number of TCs currently enabled for VSI (e.g. new credit increment = total number of credits / number of TCs)
- Update bandwidth limit table entries for each TC enabled for VSI with a new credits increment
- No need to reprogram Shared Bandwidth Limit table.

Software can use this AQ command both to change TCs enabled for VSI and modify VSI bandwidth allocation within each of enabled TCs.

Change in the bandwidth distribution does not impact tree structure. Change in TCs enabled for VSI does lead to tree structure change. TCs enabled for VSI must be enabled for the parent switching component. In case of SComp or Physical Port it is a Physical Port ETS configuration. Change in TCs enabled for VSI leads to the change in the Queue Sets enabled (one Queue Set per TC enabled for VSI). Queue Sets that are not affected by the change (belonging to the TCs that were previously enabled for VSI) should remain functional during transition. But the bandwidth distribution can be distorted during transition period.

Software must provide a full configuration of TCs enabled for VSI. This command can be used both to increase, decrease or change TC allocation for the VSI, and result in addition, removal or moving tree Nodes within the scheduling tree.

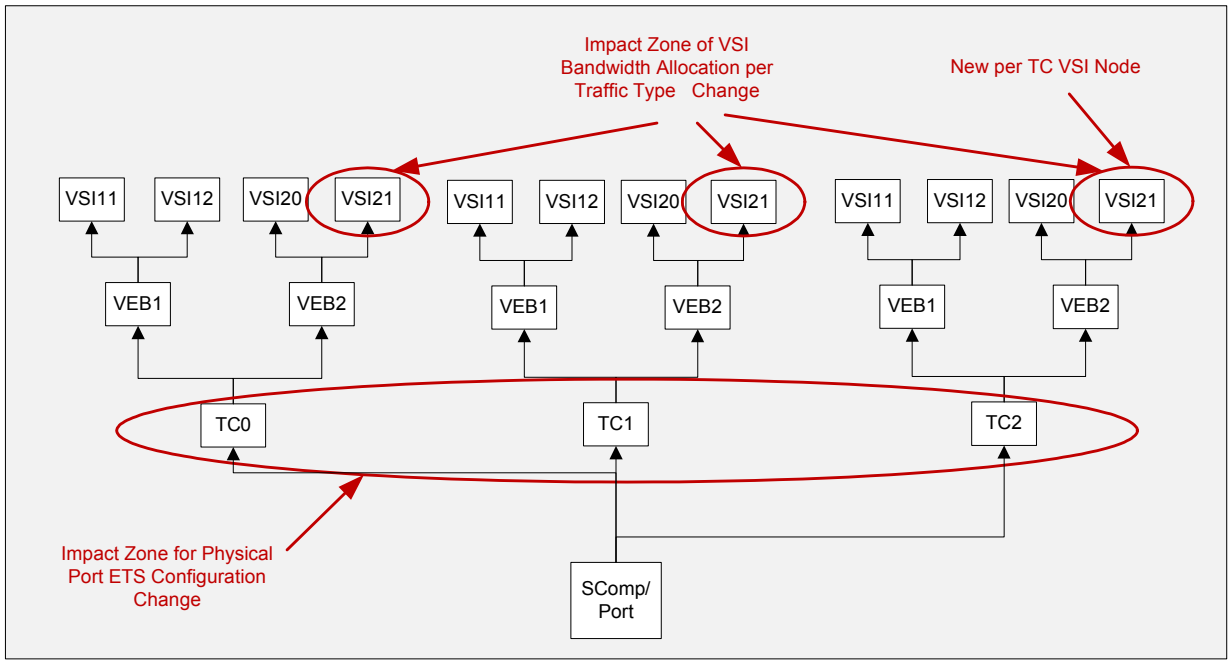
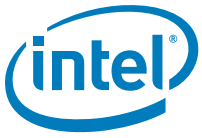


Figure A-19. Example of Per Traffic Type Configuration Change

Following steps should be performed to change TCs enabled for VSI.

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1. This entry refers to the entry in VSI/VEB/PA lookup table.
- Walk the linked list of VSI/VEB/PA lookup table entries, and identify all tree Nodes allocate for VSI.
- Use SEID of the uplink switching component to identify an index of the VSI/VEB/PA of the first tree Node. This step should be done if the uplink switching component is VEB. Otherwise firmware should use a TC nodes for each TC enabled for VSI.
- Walk the linked list of VSI/VEB/PA lookup table entries and identify all tree Nodes allocated for the parent switching component.
- Allocate a new tree Nodes required for new VSI Nodes per TC
  - Firmware can choose reuse already allocated Nodes, though from resource management stand point it seems to be easier to allocate new Nodes and deallocate old Nodes.
  - Allocation and update of new Nodes does not require atomicy, and can be performed using WriteEntry operation.
- Remove old tree Nodes that used to be configured for VSI and not removed from the list of TCs enabled
  - Remove of previously configured Nodes requires atomic update, since those Nodes are currently been configured as a part of the scheduler tree.
- Each per TC VSI Node is added as a Child to the respective VEB Node along with other VSI Nodes allocated for the VEB. Or as a Child Node of the TC Node, when VSI is directly connected to SComp or Physical Port.
- Program all new allocated Nodes with single bandwidth allocation credit, no bandwidth limit, no best effort and no work available.



- For each new allocated Node zero-out corresponding entries of the Bandwidth Limit and Best Effort tables.
  - If VSI Node is a First Sibling, update a Branch table entry. Firmware can either CopyAndShift entry of the first VSI sibling, or can CopyEntry of the previous first TC.
- Update respective Parent VEB Nodes to refer to first and last VSI Nodes.
- For each VSI Node allocate an entry in the Ready List Mapping Table, or reuse previously allocated for that VSI entry, and update the Leaf Node field.
- Firmware MUST preserve allocation of the Queue Sets to the VSI Nodes for the TCs that were kept from the previous configuration.
- Record Queue Set indexes and return them to software as QS_Handles in AdminQ completion.

Following steps are performed to change bandwidth allocation within TCs enabled for VSI.

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1. This entry refers to the entry in VSI/VEB/PA lookup table.
- Walk the linked list of VSI/VEB/PA lookup table entries, and identify all tree Nodes allocate for VSI.
- Update bandwidth allocation credits in the Nodes table for all VSI UP Nodes.

A.3.5.7.6 Configure VSI Bandwidth Limit per Traffic Type

Software can configure bandwidth limit to individual VSI Node allocated for the particular traffic type. See Chapter 7.8.4.7 for AdminQ command definition.

Software cannot provide an incremental updates to the TC/UP bandwidth limit configuration. Bandwidth limit of 0 is used to disable bandwidth limit.

In ETS-Based configuration, enabling bandwidth limit for VSI on particular TC effectively means installing a Private Bandwidth Limit on the corresponding VSI TC Node.

This AQ command can be used to change TCs enabled for VSI, similar to Configure Bandwidth Allocation per Traffic Type command described in Section A.3.5.7.5. For the description of adding or removing VSI tree Nodes due to change in TC mapping, see Section A.3.5.7.5.

Software is not allowed to configure both a Bandwidth Limit for entire VSI using AQ command described in Section A.3.5.7.4 and configure Bandwidth Limit per Traffic Type. If VSI already has a bandwidth limit configure, attempt to configure bandwidth limit per Traffic Type should fail.Following steps should be performed to configure TC bandwidth limit for the VSI.

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1. This entry refers to the entry in VSI/VEB/PA lookup table.
- Walk the linked list of VSI/VEB/PA lookup table entries, and identify all tree Nodes allocate for VSI.
- Update respective entries of the Bandwidth Limit table using WriteEntry operation. Bandwidth Limit table can be updated with immediate command, but Nodes table should be updated as atomic operation.
- Enable bandwidth limit in the respective Node table entry, using WriteField operation.

A.3.5.7.7 Query VSI Bandwidth Configuration

Software can query for the current bandwidth configuration of the VSI. See Chapter 7.8.4.15 for the AdminQ command definition.

Since this operation does not change scheduler table structure it can be done with immediate ReadEntry commands without impacting scheduler performance.

Following steps should be performed to retrieve current bandwidth configuration of the specified VSI.

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1.
- Use VSI/VEB/PA lookup table to identify all tree Nodes allocated for VSI and a TC for each NodeIf VSI Nodes have a Shared Rate Limiting account allocate,
  - Read Private Rate Limit accounts for each VSI Node in the chain
  - Sum credits allocated for each Node to calculate a VSI bandwidth limit

A.3.5.7.8 Query VSI Bandwidth Configuration per Traffic Type

Software can query for the current VSI Bandwidth configuration per Traffic Type. See Chapter 7.8.4.16 for AdminQ command definition.

Since this operation does not change scheduler table structure it can be done with immediate ReadEntry commands without impacting scheduler performance.

Following steps should be performed to retrieve VSI bandwidth configuration per traffic type.

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1.
- Use VSI/VEB/PA lookup table to identify all tree Nodes allocated for VSI and a TC for each Node Read VSI Node from Nodes table, using ReadEntry operation.
- Update bandwidth allocation information for each Node
- If VSI Nodes have a bandwidth limit enabled, but do not have a same shared rate limit account associated with all Nodes, retrieve bandwidth limit information.

A.3.5.7.9 VSI Deallocation

VSI deallocation can be requested using Switch Configuration AQ Command (Delete Element). VSI can be deallocated only if all Queue Sets associated with VSI (for all TCs enabled for VSI) are empty and do not have Transmit Queues associated with.

Following steps should be performed to deallocate VSI

- Use SEID to locate VSI in the Switching Structure Representation table, Chapter 7.4.9.1.
- Use VSI/VEB/PA lookup table to identify all tree Nodes allocated for that VSI
- Read VSI Nodes from Nodes table, using ReadEntry operation.
- Validate that each VSI Node is a Leaf Node. For ETS-Based scheme this is the only configuration supported.
- Validate that associated Queue Sets are empty and do not have any Transmit Queues associated with.
- Deallocate VSI Nodes
- No need to update Work Available status of the Parent Node.
  - Since Queue Set associated with VSI should not have any Queue associated with, VSI should not have a work available, and therefore cannot impact on Work Available status of the Parent Node.
  - Even if VSI Node was the only Node with Work Available, not updating Work Available status for Parent Node could lead to one false scheduling in worst case.
- Deallocate Queue Sets associated with deallocated VSI Nodes

A.3.5.8 Switching Component Configuration

A.3.5.8.1 Switching Component Allocation

Allocation and initial configuration of the Scheduler configuration tables is done upon allocation of the new internal switching component. Scheduler does not have a dedicated AdminQ command for that, and update of the scheduling tables is performed as a part of the Internal Switch configuration command described in Chapter 7.4.9.4.2.

Adding SComp Switching Component does not change scheduler tree configuration.

VEB/PA switching component usually added as an uplink switching component to the VSI already allocated for the Physical Port or SComp, except for the VEB/PA used for internally switched traffic only. To cover all possible configuration changes firmware should support both.

If VEB/PA added as an uplink port for existing VSI, a change should be done using series of atomic operations. In that case, VSI becomes a first VSI allocated for the VEB, and VEB takes place in the tree previously belonged to VSI. All TCs that were enabled for VSI must be enabled for VEB as well, otherwise VEB allocation should fail. If additional VSIs need to be added to VEB, this is done as a separate Add VSI operation.

TCs enabled for VEB must be a subset of TCs enabled for the Physical Port as a part of Physical Port ETS configuration. VEB might have more TCs enabled than a default VSI configured for the port.

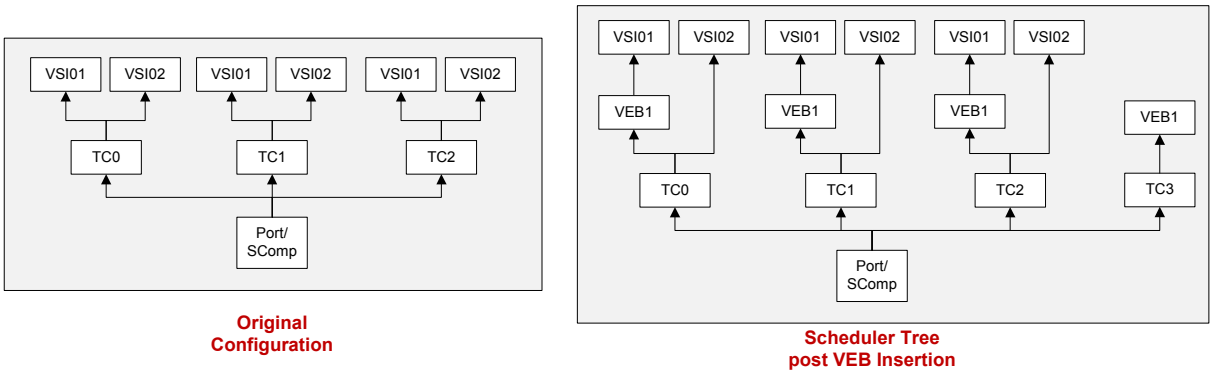


Figure A-20. Example of VEB Insertion

Figure A-20 shows an example of VEB insertion as an uplink of already allocated VSI (VSI01). VEB cannot be inserted as an uplink to multiple VSIs, only to one, since multiple VSIs allocated to SComp will have attached to different S-Channels, and all VSIs attached to VEB share S-Channel of VEB. In this example VEB is added as an uplink to VSI01, and VSI02 remains directly attached to the SComp. VEB can be configured with different number of traffic classes that are enabled for VSI. In this example VEB has TC0-TC3 enabled, while VSI has only TC0-TC2 enabled. VEB cannot add TCs that were not enabled for the Physical Port, so TC3 Tree Node should already have been configured as a part of Physical port ETS Configuration.

Adding VEB Node does not result in allocation of new Queue Sets.

Allocation of VEB/PA switching component involves allocation of multiple tree Nodes - one Node per TC. Firmware maintains a VSI/VEB/PA lookup table that allows to track all tree Nodes allocated to the particular switching component or VSI.

Allocation of Tree Nodes for the Switching Components in ETS-Based scheme is quite similar to allocation of VSI Nodes. See Chapter A.3.5.7.1.

Perform following scheduling table updates:

- Allocate VEB/PA Nodes in the Node Scheduler table.
  - One Node for each enabled TC
  - Use UP/TC lookup table or preallocated TC Nodes to locate parent TC Nodes
  - Update VSI/VEB/PA lookup table by chaining tree Nodes allocated for VEB
  - Record an index of the first VSI/VEB/PA lookup table entry in the Switching Configuration table allocated for the Switching Component
- Update Parent TC Nodes
- Configure all allocated per TC VEB/PA Node entries
  - Single bandwidth allocation credit, bandwidth limit and best effort should be disabled by default, No Work Available
- Zero out corresponding entries in the Bandwidth Limit and Best Effort tables, and initialize corresponding entry in Branch table to list all Nodes leading from the VSI Node to the tree Root Node. Firmware can use a preconfigured template in respective tables, and CopyEntry operation.
- If VEB inserted as an uplink port to VSI, update corresponding per TC VEB Nodes to refer to per TC VSI Nodes as a first child nodes
  - Update Branch table for all per TC VSI Nodes.
  - Per TC VEB Nodes should inherit bandwidth configuration of respective per TC VSI Nodes. This included bandwidth allocations, and bandwidth limits.\
- If VEB has more TCs enabled than a default VSI, firmware should allocate VEB Nodes with respective TCs Nodes a Parent, with no Children Nodes, and have a default bandwidth allocation configured for those Nodes.

A.3.5.8.2 Configure Switching Component Bandwidth Limit

Software can enable bandwidth limit for the instantiated switching component. See Chapter 7.8.4.9 for the AdminQ command definition.

Flow of configuring bandwidth limit for entire switching component is very similar to the flow of configuring bandwidth limit for VSI described in Section A.3.5.7.4.

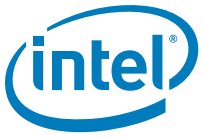
Depending on the number of TCs enabled for Switching Component, this may involve a need to allocate and configure Shared Bandwidth Limit table entry.

For more details on allocation and configuration flow see Section A.3.5.7.4.

A.3.5.8.3 Configure Physical Port ETS

Software can configure ETS of the chip physical port.

Due to small number of TC/UP Nodes in ETS-based configuration, firmware can reserve a set of dedicate tree Nodes. Firmware also maintains a lookup table that allows easily locate UP and TC tree Nodes based on their UP and TC Indexes.



ETS configuration of the Physical Port can be changed after VSI and VEB/PA tree Nodes were allocated for the Physical Port. See Appendix A.3.5.7.3 for more details.

Following steps are required to configure ETS for the Physical Port

- Allocate or use reserved Nodes for TCs enabled by ETS configuration
- For each new allocated Node zero-out corresponding entries of the Bandwidth Limit and Best Effort tables. Use pre-configured template Node and CopyEntry operation.
- Update a branch table entry for the first TC Node.
- If any TC was marked as a strict priority, order allocated TC Nodes within the Nodes table to have strict priority TCs coming first, ordered accordingly to their priority (TC7-TC0), followed by weighted round robin (WRR) Nodes, and update WSP to WRR switch point field of the VSI Node table entry with the index of the first WRR Node. If all Nodes are SWP, then set switching point to 0. If all Nodes are WRR, set switching point to the index of the first Child Node.
- Update bandwidth allocation credits in the Nodes table for all TC Nodes
- If new configuration disabled TC, corresponding TC Node must be suspended using “Suspend” firmware command.
  - Firmware is not responsible to initiate cleanup of the subtree allocated for such TC Node
  - Software must use Configure VSI/VEB/PA Bandwidth Allocation/Limit per Traffic Type commands to change TC allocation for the affected VSI/VEB/PAs and this will drive firmware to cleanup disconnected subtree
  - While suspended TC is disconnected from the scheduling tree, a subtree Nodes are still can be updated due to Work Available/No Work Available requests.
  - If DCB configuration is changed again, a TC Node and its subtree can be added back to the scheduling tree using “Resume” operation.
  - Suspended TC Node is released after software released all VSI/VEB/PA Nodes corresponding to that TC

A.3.5.8.4 Configure Switching Component Bandwidth Allocation per Traffic Type

Software can configure Switching component bandwidth allocation per traffic type. See Chapter 7.8.4.14 for AdminQ command definition.

This command can be applied to VEB/PA switching components.

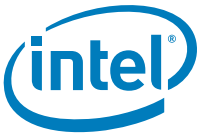
Each switching component may have multiple tree Nodes allocated, one per TC enabled for the switching component.

This command can be used to change TCs enabled for VEB. In ETS-Based configuration, VEB will have a separate tree Node per TC. TCs enabled for VEB must be a subset of TCs enabled for the corresponding Physical Port. If VEB has VSI allocated, TCs enabled for VEB must be a superset of TCs enabled for VSIs.

Flow of scheduler tables update for the VEB/PA bandwidth allocation per Traffic Type is very similar to one described in Section A.3.5.7.5 for VSI. Unlike VSIs, VEB is not associated with a Queue Sets, and therefore change in TC configuration of VEB does not result in change of Queue Sets allocated.

If software requests to disable TC previously enabled for VEB, it must make sure that all VEB VSIs have this TC disabled, prior to making VEB configuration change.

See Section A.3.5.7.5 for flow description.



A.3.5.8.5 Configure Switching Component Bandwidth Limit per Traffic Type

Software can configure bandwidth limit to individual Switching Component Node allocated for the particular traffic type. See Chapter 7.8.4.13 for AdminQ command definition.

Software cannot provide an incremental updates to the TC bandwidth limit configuration. Bandwidth limit of 0 is used to disable bandwidth limit.If switching component is connected directly to the Physical Port (SComp), then bandwidth limit would need to be configured for the particular TC tree Node.

Software can use this AQ Command to change TC enabled for VEB/PA. TCs enabled for VEB must be a subset of TCs enabled for the corresponding Physical Port. If VEB has VSI allocated, TCs enabled for VEB must be a superset of TCs enabled for VSIs.

Flow of scheduler tables update for the VEB/PA bandwidth limit per Traffic Type is very similar to one described in Section A.3.5.7.6 for VSI. Unlike VSIs, VEB is not associated with a Queue Sets, and therefore change in TC configuration of VEB does not result in change of Queue Sets allocated.

If software requests to disable TC previously enabled for VEB, it must make sure that all VEB VSIs have this TC disabled, prior to making VEB configuration change.

See Section A.3.5.7.6 for flow description.

A.3.5.8.6 Query Switching Component Bandwidth Configuration

Software can query for the current bandwidth configuration of the switching component. See Chapter 7.8.4.17 for the AdminQ command definition.

Since this operation does not change scheduler table structure it can be done with immediate ReadEntry commands without impacting scheduler performance.

Flow of retrieving VEB/PA configuration is similar to the flow described in Section A.3.5.7.7 for VSI.

A.3.5.8.7 Query Physical Port ETS Configuration

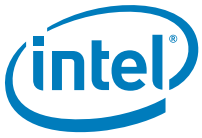
Software can query for the current ETS configuration of the specified switching component. See Chapter 7.8.4.18 for AdminQ command definition.

Since this operation does not change scheduler table structure it can be done with immediate ReadEntry commands without impacting scheduler performance.

This command only valid for the switching component connected directly to the physical port. Otherwise request should be rejected, and error returned.

Following steps should be performed to gather PhysicalPort ETS configuration

- Firmware should use TC/UP lookup table to gather information about TCs
- for each TC Node:
  - Read TC Node entry in the Nodes table
  - Use the XL710 registers to recover UP-to-TC mapping
  - Gather per TC bandwidth allocation credits
  - If TC Node had a bandwidth limit enabled, read corresponding entry of Bandwidth Limit table
- For each UP Node
- Provide gather data in completion of AdminQ command



A.3.5.8.8 Query Switching Component Bandwidth Configuration per Traffic Type

Software can query for the Switching Component bandwidth configuration per traffic type. See Chapter 7.8.4.19 for AdminQ command definition.

Since this operation does not change scheduler table structure it can be done with immediate ReadEntry commands without impacting scheduler performance.

Flow of retrieving VEB/PA configuration is similar to the flow described in Section A.3.5.7.8 for VSI.

Since VEB is not associated with Queue Sets, this step in the flow described in Section A.3.5.7.8 should be omitted.

A.3.6 Scheduler FLR/VFLR Support

Upon FLR or VFLR event, hardware will scan Ready List Mapping Table, identify Ready Lists owned by PCI Function, suspend all RLM Entries by setting a Suspend bit and schedule a special pipe-cleaner token. This token will travel thru LAN pipelines. Completion of hardware pipeline cleanup is reported in TBD register. Firmware will use this register as an indication that it can perform scheduler cleanup for the VF/PF. Token is scheduled one for each pipeline, regardless number of Ready Lists affected..

Firmware cannot assume that hardware actually ran No-Work-Available for each Suspended Ready Lists, and therefore not allowed to clear suspend bit in RLM table entry without reassigning it to the different PCI function. During FLR/VFLR flow hardware will only set a Suspend bit in ready List, and will run No-Work-Available flow only if suspended Ready List is being scheduled, as a part of the mis-scheduling processing.

Hardware guaranties that upon completion of the hardware cleanup no new work be generated for the FLR'ed or VFLR'ed functions..

Firmware is responsible for the cleanup and deallocation of the Scheduler resources upon completion of hardware pipe cleanup flow.

A.3.6.1 Standard Scheduler Configuration

In standard configuration Scheduler follow configuration of the internal switching fabric. It assumes following rules of the switching components allocation to PCI Functions.

- SComp is either owned by Physical Function or by Firmware in case it is shared by multiple Physical Functions (MFP Model). Single Physical Function may own at most one SComp.
- VEB/VEPA is owned by single Physical Function. Single Physical Function may own one or more VEBs.
- VSI is owned by Physical or Virtual Function. Single Physical or Virtual Function may own one or more VSIs.

Following steps should be performed by firmware to properly modify Scheduler resource allocation for PF FLR event

- Scan Switching Structure Representation table, Chapter 7.4.9.1
- For each switching component owned by PF
  - — Retrieve Node Index

  - — All Queue Sets owned by PF and respective VFs should be suspended now, and not processed by the Scheduler
  - — All VSIs and switching components owned by VF/PF should not have work-available, and therefore should be skipped by Scheduler during scheduling process
- Walk suspended subtrees, cleanup Nodes, and release Ready List Mapping Table entries
  - — Clean does not require atomicy, since it is performed on the Nodes that are not processed by Scheduler.

Following steps should be performed by firmware to properly modify Scheduler resource allocation for VF FLR event

- Scan Switching Structure Representation table, Chapter 7.4.9.1
- For each virtual port owned by VF
  - — Retrieve Node Index
  - — All Queue Sets owned by VF should be suspended now, and not processed by the Scheduler
  - — All VSIs owned by VF should not have work-available and therefore should be skipped by Scheduling during scheduling process.
- Walk suspended subtrees, cleanup Nodes, and release Ready List Mapping Table entries
  - — Clean does not require atomicy, since it is performed on the Nodes that are not processed by Scheduler.