Intel® Ethernet Controller X550

Datasheet

Ethernet Products Group (EPG)

PRODUCT FEATURES

General

- Serial Flash interface
  - Firmware authentication on update
- Configurable LED operation for software or customizing OEM LED displays
- Device disable capability
- Package sizes
  - 25 mm x 25 mm (X550-BT2)
  - 17 mm x 17 mm (X550-AT2)

Networking

- 10 GbE/1 GbE/100 Mb/s copper PHYs integrated on-chip
- Support for jumbo frames of up to 15.5 KB
- Flow control support: send/receive pause frames and receive FIFO thresholds
- Statistics for management and RMON
- 802.1q VLAN support
- TCP segmentation offload: up to 256 KB
- IPv6 support for IP/TCP and IP/UDP receive checksum offload
- Fragmented UDP checksum offload for packet reassembly
- Message Signaled Interrupts (MSI)
- Message Signaled Interrupts (MSI-X)
- Interrupt throttling control to limit maximum interrupt rate and improve CPU usage
- Flow Director (16 x 8 and 32 x 4)
- 128 transmit queues
- Receive packet split header
- Receive header replication
- Dynamic interrupt moderation
- TCP timer interrupts
- Relaxed ordering
- Support for 64 virtual machines per port (64 VMs x 2 queues)
- Support for Data Center Bridging (DCB);(802.1Qaz, 802.1Qbb, 802.1p)

Host Interface

- PCIe 3.0 Base Specification
- Bus width — x1, x4, x8
- 64-bit address support for systems using more than 4 GB of physical memory

MAC Functions

- Descriptor ring management hardware for transmit and receive
- ACPI register set and power down functionality supporting D0 and D3 states
- A mechanism for delaying/reducing transmit interrupts
- Software-controlled global reset bit (resets everything except the configuration registers)
- Four Software-Definable Pins (SDP) per port
- Wake up
- IPv6 wake-up filters
- Configurable flexible filter (through NVM)
- LAN function disable capability
- Programmable memory transmit buffers (160 KB/port)
- Default configuration by NVM for all LEDs for pre-driver functionality

Manageability

- SR-IOV support
- Eight VLAN L2 filters
- 16 Flex L3 port filters
- Four Flexible TCO filters
- Four L3 address filters (IPv4)
- Advanced pass through-compatible management packet transmit/receive support
- SMBus interface to an external Manageability Controller (MC)
- NC-SI interface to an external MC
- Four L3 address filters (IPv6)
- Four L2 address filters

Revision 2.6
January 2021
333369-008

Revision History

Revision	Date	Notes
2.6	January 20, 2021	Updates include the following: • Updated title page to include "Firmware authentication on update" under General Product Features. • Updated Section 1.3.3, "Serial Flash Interface" with discussion on signed firmware authentication.
2.5	August 6, 2020	Updates include the following: • Updated table in Section 2.2.5, "NC-SI". • Updated table in Section 2.2.8, "RSVD and No-Connect Pins". • Updated Table 2-1, "External Pull-Up Resistors". • Updated Section 3.7.3.2, "Auto-Negotiation and Link Setup". • Updated Table 4-1, "Notes for Power-Up Timing Diagram". • Updated table containing list of support flash devices in Section 12.8.1, "Flash".
2.4	April 9, 2020	Updates include the following: • Updated title of Section 6.2.2.79, "Reserved (0x004E)". • Updated Section 6.2.17.6, "NC-SI Configuration 2 (0x0005)" — Added Bit 14, *NC-SI Package ID from SDP Workaround*. • Updated Figure 11-12, "MCTP Bus Transition State Machine". • Updated Table 12-2, "X550-AT Power Consumption" — Revised device total power numbers. • Added Section 12.8, "Devices Supported" and Section 12.8.1, "Flash".
2.3	November 19, 2018	Updates include the following: • Added changes related to NVM Recovery Mode: — Section 8.2.2.20.2, "Firmware Semaphore Register - FWSM (0x00010148)" — Added Bit 5, *OPERATION_MODE*. — Added Section 15.3, "NVM Recovery Mode". • Updated Section 6, "Non-Volatile Memory Map". • Updated Table 11-43, "Get MCTP Version Support Returned Value". • Updated table in Section 12.3.10.1, "Aux Power on Peak Current Consumption". • Updated Table 14-2, "The X550 Absolute Thermal Maximum Rating (°C)"
2.2	July 21, 2017	Updates include the following: • Added Section 2.2.8.1, "Pin Differences in the X550-AT Single Port Device". • Section 11.7.6.1.3 — Added reference to list of support message types. • Section 11.7.6.1.3.1 — Modified verbiage in "Value" column for Bytes 3:5 in Table 11-44. • Section 12.3.9 — Added new table for X550-AT power consumption. • Section 12.3.10.1 — Updated values in associate table.
2.1	May 10, 2016	Updates include the following: • Removed EEC.FLUPD bit. No longer used for triggering Shadow RAM dump. • Removed FLUPDATE register (0x00015F54). • Table 3-26 — Updated description for SDP1. • Section 9.2.3.6.7, "Link Capabilities Register (0xAC; RO)" — Changed default value for ASPM support (bits 11:10) to 10b. • Section 11.8.3.1, "Driver Info Host Command" — Updated Table 11-49. • Table 12-3 and Table 12-4 — Changed Device Total Power units from mW to W. • Table 12-20 — Updated thermal diode typical ESR value to 2.77 Ω. • Table 15-2 — Updated ID Code values. • Other miscellaneous updates.

Revision	Date	Notes
2.0	January 8, 2016	Updates include the following: • Updated PHY Registers section. • Changed Max temperature in NVM mode to 102 (Tjunction max changed 107). • Added NBASE-T information. • Removed 10BASE-T information. • Removed x2 lane width. • Updated power numbers. • Updated heat sink and other thermal information.
1.9[1]	October 27, 2015	Initial release (Intel public)

1. There were no previous versions of this document released.

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

Contents

Chapter 1 Introduction

1.1 Scope

This document describes the external architecture (including device operation, pin descriptions, register definitions, etc.) for the X550, a dual port 10GBASE-T Network Interface Controller.

This document is intended as a reference for logical design group, architecture validation, firmware development, software device driver developers, board designers, test engineers, or anyone else who might need specific technical or programming information about the X550.

1.2 Product Overview

The X550 is a derivative of the X540. Many features of its predecessor remain intact; however, some have been removed or modified as well as new features introduced.

The X550 includes two integrated 10GBASE-T copper Physical Layer Transceivers (PHYs). A standard MDIO interface, accessible to software via MAC control registers, is used to configure and monitor each PHY operation.

1.2.1 System Configurations

The X550 is targeted for system configurations such as rack mounted, pedestal servers or workstations, where it can be implemented used as an add-on NIC or LAN on Motherboard (LOM), or purchased from Intel as a standard PCIe* adapter card.

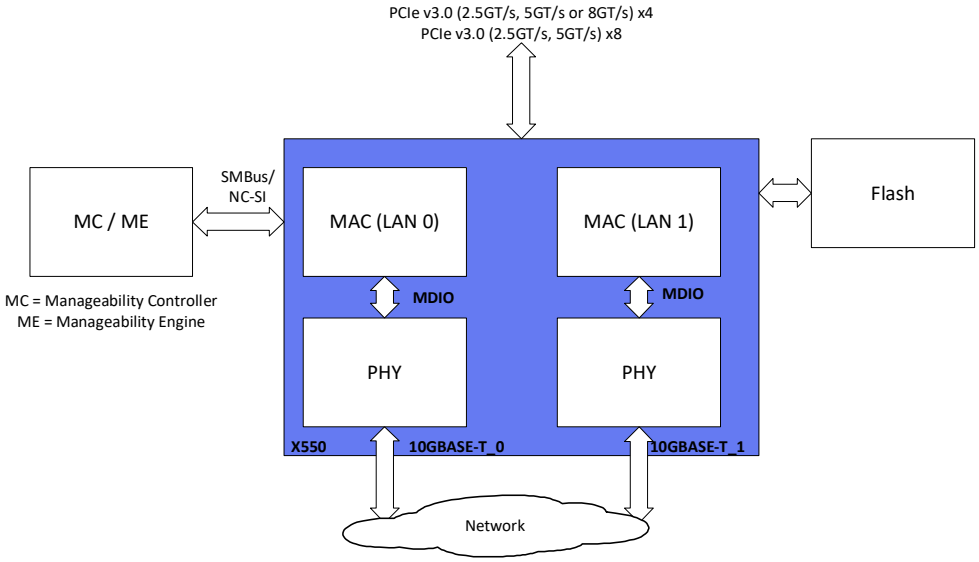


Figure 1-1. Typical Rack/Pedestal System Configuration

1.3 External Interfaces

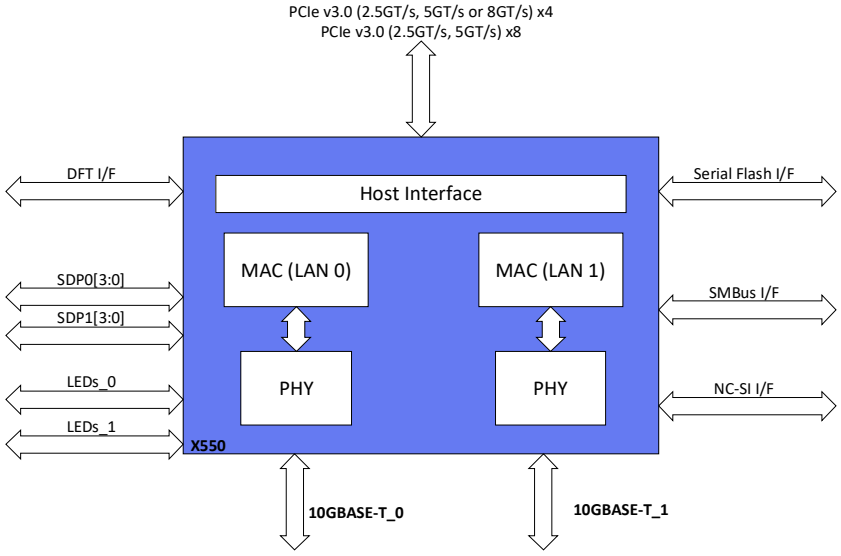


Figure 1-2. X550 External Interfaces Diagram

1.3.1 PCIe Interface

The X550 supports PCIe v3.0 (2.5 GT/s, 5 GT/s or 8 GT/s). See Section 2.2.1 for full pin description and Section 12.4.5 for interface timing characteristics.

1.3.2 Network Interfaces

Two independent 10GBASE-T interfaces are used to connect the two X550 ports to external devices. Each 10GBASE-T interface can operate at any of the following speeds:

- 10 Gb/s, 10GBASE-T mode
- 5 Gb/s, NBASE-T mode
- 2.5 Gb/s, NBASE-T mode
- 1 Gb/s, 1000BASE-T mode
- 100 Mb/s, 100BASE-TX mode
- Refer to Section 2.2.2 for full-pin descriptions. For the timing characteristics of those interfaces, refer to the relevant external specifications listed in Section 12.4.6.

1.3.3 Serial Flash Interface

The X550 uses an external SPI serial interface to a Flash device, also referred to as Non-Volatile Memory (NVM). The X550 supports serial Flash devices with up to 4 MB of memory.

The X550 implements a signed firmware authenticated capability to verify the firmware and critical device settings with built-in detection of corruption. This is done at the time of firmware updates. See Section 3.4.1.1, "NVM Protection" for details.

1.3.4 SMBus Interface

SMBus is an optional interface for pass-through and/or configuration traffic between an external Manageability Controller (MC) and the X550.

The X550's SMBus interface supports a standard SMBus, up to a frequency of 1 MHz. Refer to Section 2.2.4 for full-pin descriptions and Section 12.4.4.3 for timing characteristics of this interface.

1.3.5 NC-SI Interface

NC-SI is an optional interface for pass-through traffic to and from an MC. The X550 meets the NC-SI version 1.0.0 specification.

Refer to Section 2.2.5 for the pin descriptions, and Section 11.6 for NC-SI programming.

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

The X550 has four SDP pins per port that can be used for miscellaneous hardware or software-controllable purposes. These pins can each be individually configured to act as either input or output pins. Via the SDP pins, the X550 can support IEEE1588 auxiliary device connections and other functionality. For more details on the SDPs see Section 3.5 and the ESDP register (Section 8.2.2.1.4).

1.3.7 LED Interface

The X550 implements four output drivers intended for driving external LED circuits per port. Each of the four LED outputs can be individually configured to select the particular event, state, or activity, which is indicated on that output. In addition, each LED can be individually configured for output polarity as well as for blinking versus non-blinking (steady-state) indications.

The configuration for LED outputs is specified via the LEDCTL register (see Section 8.2.2.1.10). In addition, the hardware-default configuration for all LED outputs can be specified via an NVM field (see Section 6.2.5.5 and Section 6.2.5.6), thereby supporting LED displays configured to a particular OEM preference. For more details on the LEDs see Section 3.6.

1.4 Feature Summary

Table 1-1 to Table 1-7 list the X550's features in comparison to previous dual-port 10 GbE Ethernet controllers.

Table 1-1. Network Features

Feature	82599	X540	X550
Compliant with the 10 GbE and 1 GbE Ethernet/802.3ap (KX/KX4) specification	Y	N	N
Compliant with the 10 GbE 802.3ap (KR) specification	Y	N	N
Compliant with XFI/SFI interface	Y	N	N
Compliant with the 1000BASE-BX specification	Y	N	N
Full-duplex operation at all supported speeds	Y	Y	Y
Half-duplex at 100 Mb/s operation	N	N	N
10 GbE/1 GbE/100 Mb/s copper PHYs integrated on-chip	N	Y	Y
802.3az Energy Efficient Ethernet (EEE) support	N	N	Y
Support jumbo frames of up to 15.5 KB	Y[1]	Y[1]	Y[1]
MDIO interface Clause 45	Y	Y (internally)	Y (internally)
Flow Control support: Send/receive pause frames and receive Fifo thresholds	Y	Y	Y
Statistics for Management and RMON	Y	Y	Y
802.1q VLAN support	Y	Y	Y
SerDes interface for external PHY connection or system interconnect	Y	N	N
SGMII interface	Y (100 Mb/s and 1 GbE only)	N	N
Double VLAN	Y	Y	Y

1. All the products support full-size 15.5 KB jumbo packets while in a basic mode of operation. When DCB mode is enabled, or security engines enabled, or virtualization is enabled, or OS2BMC is enabled, then only 9.5 KB jumbo packets are supported. Packets to/from the MC longer than 2 KB are filtered out.

Table 1-2. Host Interface Features

Feature	82599	X540	X550
PCIe* version (Speed)	PCIe v2.0 (5/2.5 GT/s)	PCIe v2.1 (5/2.5 GT/s)	PCIe v3.0 (8/5/2.5 GT/s)
Number of lanes	x1, x2, x4, x8	x1, x2, x4, x8	x1, x4 x8 (For X550-BT2, x8 available in Gen 1/2 only)
64-bit address support for systems using more than 4 GB of physical memory	Y	Y	Y
Outstanding requests for Tx data buffers	16	16	16
Outstanding requests for Tx descriptors	8	8	8
Outstanding requests for Rx descriptors	8	8	8
Credits for P-H/P-D/NP-H/NP-D (shared for the two ports)	16/16/4/4	16/16/4/4	16/16/4/4
Max Payload Size supported	512 Bytes	512 Bytes	512 Bytes
Max Request Size supported	2 KB	2 KB	2 KB
Link layer retry buffer size (shared for the two ports)	3.4 KB	3.4 KB	3.4 KB
Vital Product Data (VPD)	Y	Y	Y
End to End CRC (ECRC)	Y	Y	Y
TLP Processing Hints (TPH)	N	N	Y
Latency Tolerance Reporting (LTR)	N	N	Y
ID-Based Ordering (IDO)	N	N	Y
Access Control Services (ACS)	N	Y	Y
ASPM optional compliance capability	N	Y	Y
PCIe functions off via pins, while LAN ports are on	N	Y	Y

Table 1-3. Miscellaneous Features

Feature	82599	X540	X550
Serial Flash Interface (SFI)	Y	Y	Y
4-wire SPI EEPROM interface	Y	N	N
Configurable LED operation for software or OEM customization of LED displays	Y	Y	Y
Protected NVM Space for Private Configuration	Y	Y	Y
Device disable capability	Y	Y	Y
Package size	25 mm x 25 mm	25 mm x 25 mm	25 mm x 25 mm 17 mm x 17 mm
Embedded thermal sensor	N	Y	Y
Embedded thermal diode	N	Y	Y
Watchdog timer	Y	Y	Y
Time Sync (IEEE 1588)	Y	Y[1]	Y
Time Stamp in packet	N	N	Y

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

Table 1-4. LAN Functions Features

Feature	82599	X540	X550
Programmable host memory receive buffers	Y	Y	Y
Descriptor ring management hardware for transmit and receive	Y	Y	Y
ACPI register set and power down functionality supporting D0 and D3 states	Y	Y	Y
Integrated IPsec security engines: AES-GCM 128-bit; AH or ESP encapsulation; IPv4 and IPv6 (no option or extension headers)	1024 SA / port	1024 SA / port	1024 SA / port
Software-controlled global reset bit (resets everything except the PCIe configuration registers)	Y	Y	Y
Software-Definable Pins (SDP) (per port)	8	4	4
Four SDP Pins can be configured as general purpose interrupts	Y	Y	Y
Wake on LAN (WoL)	Y	Y	Y
IPv6 Wake-up Filters	Y	Y	Y
Configurable (through NVM) Wake-up Flexible Filters	Y	Y	Y
Default configuration by NVM for all LEDs for pre-driver functionality	Y	Y	Y
LAN Function Disable capability	Y	Y	Y
Programmable memory transmit buffers	160 KB / port	160 KB / port	160 KB / port
Programmable memory receive buffers	512 KB / port	384 KB / port	384 KB / port

Table 1-5. LAN Performance Features

Feature	82599	X540	X550
TCP/UDP segmentation offload	256 KB in all modes	256 KB in all modes	256 KB in all modes
TSO interleaving for reduced latency	Y	Y	Y
TCP Receive Side Coalescing (RSC)	32 flows / port	32 flows / port	32 flows / port
Data Center Bridging (DCB), IEEE Compliance to: • Enhanced Transmission Selection (ETS) - 802.1Qaz • Priority-based Flow Control (PFC) - 802.1Qbb	 Y (up to 8) Y (up to 8)	 Y (up to 8) Y (up to 8)	 Y (up to 8) Y (up to 8)
Rate limit VM Tx traffic per TC (i.e. per TxQ)	Y	Y	Y
IPv6 support for IP/TCP and IP/UDP receive checksum offload	Y	Y	Y
Fragmented UDP checksum offload for packet reassembly	Y	Y	Y
FCoE Tx / Rx CRC offload	Y	Y	Y
FCoE transmit segmentation	256 KB	256 KB	256 KB
FCoE coalescing and direct data placement	512 outstanding Read — Write requests / port 256 buffers per request	512 outstanding Read — Write requests / port 256 buffers per request	2048 outstanding Read — Write requests / port 1024 buffers per request
Message Signaled Interrupts (MSI)	Y	Y	Y
Message Signaled Interrupts (MSI-X)	Y	Y	Y
Interrupt Throttling Control to limit maximum interrupt rate and improve CPU use	Y	Y	Y
Rx packet split header	Y	Y	Y

Table 1-5. LAN Performance Features [continued]

Feature	82599	X540	X550
Multiple Rx queues (RSS)	Y (multiple modes)	Y (multiple modes)	Y (multiple modes)
Flow Director Filters: up to 32 KB flows by hash filters or up to 8 KB perfect match filters	Y	Y	Y
Number of Rx queues (per port)	128	128	128
Number of Tx queues (per port)	128	128	128
Low Latency Interrupts (LLI)	Y	Y	N
DCA support	Y	Y	N
TCP timer interrupts	Y	Y	Y
No snoop	Y	Y	N
Relax ordering	Y	Y	Y
DMA coalescing	N	N	Y

Table 1-6. Virtualization Features

Feature	82599	X540	X550
Support for Virtual Machine Device Queues (VMDq1 and Next Generation VMDq)	64	64	64
L2 Ethernet MAC Address filters (unicast and multicast)	128	128	128
L2 VLAN filters	64	64	64
PCI-SIG SR IOV	Y	Y	Y
RSS table per VF	N	N	Y
Traffic shaping	Y	Y	Y
Anti-spoof	MAC, VLAN	MAC, VLAN	MAC, VLAN, EtherType
Malicious driver protection	N	N	Y
Forwarding modes	MAC, VLAN	MAC, VLAN	MAC, VLAN, E-tag
VEB support • Multicast and broadcast packet replication • Packet mirroring • Packet loopback	 Y Y Y	 Y Y Y	 Y Y Y
VEPA support (on top of VEB support) • Source pruning	 N	 N	 Y
E-tag filtering support	N	N	Y

Table 1-7. Manageability Features

Feature	82599	X540	X550
Advanced pass-through-compatible management packet transmit/ receive support	Y	Y	Y
SMBus interface to an external MC	Y	Y	Y
New Management Protocol Standards Support (NC-SI) interface to an external MC	Y	Y	Y
L2 address filters	4	4	4
VLAN L2 filters	8	8	8
Flex L3 port filters	16	16	16
Flexible TCO filters	4	4	1
L3 address filters (IPv4)	4	4	4
L3 address filters (IPv6)	4	4	4
Host-based Application-to-BMC Network Communication patch (OS2BMC)	N	Y	Y
Flexible MAC Address	N	Y	Y
MC inventory of LOM device information	N	Y	Y
iSCSI boot configuration parameters via MC	N	Y	Y
MC monitoring	N	Y	Y
NC-SI to iMC	N	Y	Y
NC-SI arbitration	N	Y	Y
MCTP over SMBus (pass-through and control)	N	Y (control only)	Y
MCTP over PCIe (pass-through and control)	N	N	Y
NC-SI package ID via SDP pins	N	Y	Y
NC-SI Flow control	N	N	Y

1.5 Overview: New Capabilities Beyond the X540

1.5.1 NBASE-T Support

Support for 2.5GBASE-T and 5GBASE-T is added to the X550.

1.5.2 Filtering Capabilities

1.5.2.1 Flow Director Improvements

Two new modes, based on cloud tenant ID or on MAC, VLAN are added to the X550 to allow support of the features described below. Also supported is a better definition of the packet which are candidate to the flow director filtering and the ability to drop candidate packets that do not match any filter.

1.5.2.2 802.1BR Support

The X550 supports the IEEE 802.1BR specification. It allows forwarding to pools based on unicast or multicast E-tags and allow insertion and removal of the E-tag using a per pool policy.

To allow L2 filtering on top of the E-tag forwarding, the flow director may be configured to MAC, VLAN filtering and non matching packets may be dropped.

1.5.2.3 VXLAN and NVGRE Support

The X550 supports detection and off-loading of NVGRE and VXLAN packets. It provides transmit and receive checksum off-load on both inner and outer IP headers and on TCP header. It also allows forwarding to a specific VM within a tenant using a new flow director mode.

In the regular IP mode of the flow director, VXLAN and NVGRE flows can be differentiated from regular IP packets and filtering based on the inner IP/L4 header is supported.

1.5.3 IEEE 1588 Improvements

The X550 improves the support for IEEE 1588 by adding the following features:

- Sampling based on a fixed clock, allowing operation independent from the link speed.
- Clock representation is divided to seconds, nanoseconds and sub-nano parts - allowing easier handling by software.
- Enabling of sub-ns periodic corrections.
- Gradual time adjustment of frequency corrections preventing single large correction. An interrupt is provided when the adjustment is done.
- Support for two different target times for SDP toggling.
- Each SDP can be associated with any 1588 functionality.
- Allow timestamp to be received in register or embedded in packet.

1.5.4 Manageability

1.5.4.1 DMTF MCTP Protocol Over PCIe

The X550 enables reporting and controlling all information exposed in a LOM device via NC-SI using the MCTP protocol over PCIe in addition to SMBus. The MCTP interface over PCIe is used by the MC to control the NIC and for pass-through traffic. In addition, the MCTP over SMBus interface can also be used for pass-through traffic. For more information, refer to Section 11.7.

1.5.4.2 NVM Structures

Management related NVM structures were updated. For further information see Chapter 6, "Non-Volatile Memory Map".

1.5.4.3 Simplified SMBus TCO Status and Filter Setting

The TCO status in an SMBus received packet was reduced to 8 bytes and most of the information was removed to keep only the information relevant to the MCs. See Section 11.5.11.2.1.1 for details.

In addition, a generic command was added to enable the setting of most common filtering options independently of the actual filters implementation. See Section 11.5.11.1.7 and Section 11.5.11.1.8 for details.

1.5.4.4 Diagnostic Commands

A command was added to the legacy SMBus interface to enable querying the identity of the X550 and the firmware versions currently running on the X550. See Section 11.5.11.2.6 for details. This command is the SMBus counterpart of the NC-SI command described in Section 11.6.3.13.2.

1.6 Conventions

1.6.1 Terminology and Acronyms

See Section 16, "Glossary and Acronyms".

1.6.2 Byte Ordering

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

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

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

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

1.7 References

The X550 implements features from the following specifications:

IEEE Specifications:

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

PCI-SIG Specifications:

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

IETF Specifications:

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

IETF Drafts:

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

Manageability Documents:

- DSP0222 — DMTF Network Controller Sideband Interface (NC-SI) Specification rev 1.0.1, January 2013
- DSP0236 — DMTF Management Component Transport Protocol (MCTP) Base Specification, rev 1.2.0, January 2013
- DSP0237 — DMTF Management Component Transport Protocol (MCTP) SMBus/I^2C Transport Binding Specification, rev 1.0.0, July 2009
- DSP0238 — DMTF Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding Specification, rev 1.0.1, December 2009
- DSP0239 — DMTF Management Component Transport Protocol (MCTP) IDs and Codes, rev 1.2.0, August 2012
- DSP0261 — DMTF NC-SI Over MCTP Binding Specification - Work in Progress,
- System Management Bus (SMBus) Specification, SBS Implementers Forum, Ver. 2.0, August 2000
- UM10204 — I^2C-bus specification and user manual Rev. 5 — 9 October 2012

Proxy Documents:

- proxZZZy™ for sleeping hosts, February 2010 (ECMA-393)

Other:

- Advanced Configuration and Power Interface Specification, Rev 2.0b, October 2002
- EUI-64 specification, http://standards.ieee.org/regauth/oui/tutorials/EUI64.html.
- Definition for new PAUSE function, Rev. 1.2, 12/26/2006.
- GCM spec — McGrew, D. and J. Viega, “The Galois/Counter Mode of Operation (GCM)”, Submission to NIST, January 2004.

  http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf
- FRAMING AND SIGNALING-2 (FC-FS-2) Rev 1.00
- Fibre Channel over Ethernet Draft Presented at the T11 on May 2007
- Per Priority Flow Control (by Cisco Systems) — Definition for new PAUSE function, Rev 1.2, EDCS-472530
- NBASE-T Physical Layer Specification version 1.1

1.8 Architecture and Basic Operation

1.8.1 Transmit (Tx) Data Flow

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

Table 1-8. Tx Data Flow

Step	Description
1	The software device driver creates a descriptor ring and configures one of the X550's transmit queues with the address location, length, head, and tail pointers of the ring (one of 128 available Tx queues).
2	The software device driver is requested by the TCP/IP stack to transmit a packet, it gets the packet data within one or more data buffers.
3	The software device driver initializes the descriptor(s) that point to the data buffer(s) and have additional control parameters that describes the needed hardware functionality. The host places that descriptor in the correct location in the appropriate Tx ring.
4	The software device driver updates the appropriate Queue Tail Pointer (TDT).
5	The X550's DMA senses a change of a specific TDT and as a result sends a PCIe request to fetch the descriptor(s) from host memory.
6	The descriptor(s) content is received in a PCIe read completion and is written to the appropriate location in the descriptor queue.
7	The DMA fetches the next descriptor and processes its content. As a result, the DMA sends PCIe requests to fetch the packet data from system memory.
8	The packet data is received from PCIe completions and passes through the transmit DMA that performs all programmed data manipulations (various CPU offloading tasks as checksum offload, TSO offload, etc.) on the packet data on the fly.
9	While the packet is passing through the DMA, it is stored into the transmit FIFO. After the entire packet is stored in the transmit FIFO, it is then forwarded to transmit switch module.
10	The transmit switch arbitrates between host and management packets and eventually forwards the packet to the security module.
11	The security module optionally encrypts the packet and authenticates it using IPsec and passes the packet to the MAC.
12	The MAC appends the L2 CRC to the packet and delivers the packet to the integrated PHY.
13	The PHY performs the PCS encoding, scrambling, Low-Density Parity Check (LDPC) encoding, and the other manipulations required to deliver the packet over the copper wires at the selected speed.
14	When all the PCIe completions for a given packet are complete, the DMA updates the appropriate descriptor(s).
15	The descriptors are written back to host memory using PCIe posted writes. The head pointer is updated in host memory as well if the X550 is configured to do so.
16	An interrupt is generated to notify the software device driver that the specific packet has been read to the X550 and the driver can then release the buffer(s).

1.8.2 Receive (Rx) Data Flow

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

Table 1-9. Rx Data Flow

Step	Description
1	The software device driver creates a descriptor ring and configures one of the X550's receive queues with the address location, length, head, and tail pointers of the ring (one of 128 available Rx queues).
2	The software device driver initializes descriptor(s) that point to empty data buffer(s). The software device driver places these descriptor(s) in the correct location at the appropriate Rx ring.
3	The software device driver updates the appropriate Queue Tail Pointer (RDT).
4	A packet enters the PHY through the copper wires.
5	The PHY performs the required manipulations on the incoming signal such as LDPC decoding, de-scrambling, PCS decoding, etc.
6	The PHY delivers the packet to the Rx MAC.
7	The MAC forwards the packet to the security block.
8	If the packet is identified as a IPsec packet, it is decrypted.
9	If the packet matches the pre-programmed criteria of the Rx filtering, it is forwarded to an Rx FIFO.
10	The receive DMA fetches the next descriptor from the appropriate host memory ring to be used for the next received packet.
11	After the entire packet is placed into an Rx FIFO, the receive DMA posts the packet data to the location indicated by the descriptor through the PCIe interface. If the packet size is greater than the buffer size, more descriptor(s) are fetched and their buffers are used for the received packet.
12	When the packet is placed into host memory, the receive DMA updates all the descriptor(s) that were used by the packet data.
13	The receive DMA writes back the descriptor content along with status bits that indicate the packet information including what offloads were done on that packet.
14	The X550 initiates an interrupt to the software device driver to indicate that a new received packet is ready in host memory.
15	The software device driver reads the packet data and sends it to the TCP/IP stack for further processing. The software device driver releases the associated buffer(s) and descriptor(s) once they are no longer in use.

Chapter 2 Pin Interface

2.1 Signal Type Definition

Signal	Definition	DC Specification
In	Standard 3.3 V I/O buffer, functions as input-only signal.	
Out (O)	Standard 3.3 V I/O buffer, functions as output-only signal.	
T/s	Tri-state is a 3.3 V bi-directional, tri-state input/output pin.	
O/d	Open drain enables multiple devices to share as a wire-OR.	Section 12.4.2
A-in	Analog input signals.	Section 12.4.5 and Section 12.4.6
A-out	Analog output signals.	Section 12.4.5 and Section 12.4.6
A-Inout	Bi-directional analog signals.	
B	Input BIAS.	
NCSI-in	NC-SI 3.3 V input signal.	Section 12.4.3
NCSI-out	NC-SI 3.3 V output signal.	Section 12.4.3
In-1p2	1.2 V input-only signal. 3.3 V tolerance.	
In-Only	Standard 3.3 V buffer input-only signal.	
Out-Only	Standard 3.3 V buffer output-only signal.	
LVDS-O	Low voltage differential signal - output.	
Pup	Pull up.	
Pdn	Pull down.	

2.2 Pin Assignments

2.2.1 PCIe

See AC/DC specifications in Section 12.4.5.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
PET_0_p PET_0_n	T5 T6	AC3 AD3	A-Out			**PCIe Serial Data Output**: A serial differential output pair running at 8 Gb/s, 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	T8 T9	AC4 AD4	A-Out			
PET_2_p PET_2_n	T11 T12	AC9 AD9	A-Out			
PET_3_p PET_3_n	T14 T15	AC10 AD10	A-Out			
PET_4_p PET_4_n	N/A	AC15 AD15	A-Out			**PCIe Serial Data Output**: A serial differential output pair running at 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end. Available only in the X550-BT2.
PET_5_p PET_5_n	N/A	AC16 AD16	A-Out			
PET_6_p PET_6_n	N/A	AC21 AD21	A-Out			
PET_7_p PET_7_n	N/A	AC22 AD22	A-Out			
PER_0_p PER_0_n	P6 P7	AB2 AB1	A-In			**PCIe Serial Data Input**: A serial differential input pair running at 8 Gb/s, 5 Gb/s, or 2.5 Gb/s. This input carries both data and an embedded 8 GHz, 5 GHz, or 2.5 GHz clock that is recovered along with data at the receiving end.
PER_1_p PER_1_n	P9 P10	AD6 AC6	A-In			
PER_2_p PER_2_n	P12 P13	AD7 AC7	A-In			
PER_3_p PER_3_n	P15 P16	AD12 AC12	A-In			
PER_4_p PER_4_n	N/A	AD13 AC13	A-In			**PCIe Serial Data Input**: A serial differential input pair running at 5 Gb/s or 2.5 Gb/s. This input carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end. Available only in the X550-BT2.
PER_5_p PER_5_n	N/A	AD18 AC18	A-In			
PER_6_p PER_6_n	N/A	AD19 AC19	A-In			
PER_7_p PER_7_n	N/A	AB23 AB24	A-In			
PE_CLK_p PE_CLK_n	N4 P4	Y2 Y1	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.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
PE_RBIAS	T3	Y4	A-Inout			**BIAS**: A 4.75 KΩ ±0.1% resistor should be connected between RBIAS and RSENSE pins. Connect resistor as close as possible to the chip. Resistor is used for internal impedance compensation and BIAS current generation circuitry.
PE_RSENSE	R3	Y5	A-Inout			
PE_WAKE_N	L3	W1	O/d		Pup[1]	**Wake**: Pulled low to indicate that a Power Management Event (PME) is pending and the PCIe link should be restored. Defined in the PCIe specifications.
PE_RST_N	M3	W2	In	Pup		**Power and Clock Good Indication**: Indicates that power and the PCIe reference clock are within specified values. Defined in the PCIe specifications. Also called PCIe Reset.

1. Pup value should be considered as 10 KΩ.

2.2.2 MDI

See AC/DC specifications in Section 12.4.6.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
MDI0_0_p	A2	A3	A-Inout			**Port 0 pair A+ of the Line Interface**: Connects to the Pair A+ input of the transformer. On reset, set to high impedance.
MDI0_0_n	B2	B3	A-Inout			**Port 0 pair A- of the Line Interface**: Connects to the Pair A- input of the transformer. On reset, set to high impedance.
MDI0_1_p	A3	A5	A-Inout			**Port 0 pair B+ of the Line Interface**: Connects to the Pair B+ input of the transformer. On reset, set to high impedance.
MDI0_1_n	B3	B5	A-Inout			**Port 0 pair B- of the Line Interface**: Connects to the Pair B- input of the transformer. On reset, set to high impedance.
MDI0_2_p	A5	A7	A-Inout			**Port 0 pair C+ of the Line Interface**: Connects to the Pair C+ input of the transformer. On reset, set to high impedance.
MDI0_2_n	B5	B7	A-Inout			**Port 0 pair C- of the Line Interface**: Connects to the Pair C- input of the transformer. On reset, set to high impedance.
MDI0_3_p	A6	A9	A-Inout			**Port 0 pair D+ of the Line Interface**: Connects to the Pair D+ input of the transformer. On reset, set to high impedance.
MDI0_3_n	B6	B9	A-Inout			**Port 0 pair D- of the Line Interface**: Connects to the Pair D- input of the transformer. On reset, set to high impedance.
MDI0_4_p	A8	A11	A-Inout			**Port 0 Analog Test+**: Connects to the pair E+ input of the transformer.
MDI0_4_n	B8	B11	A-Inout			**Port 0 Analog Test-**: Connects to the pair E- input of the transformer.
MDI1_0_p	B16	A22	A-Inout			**Port 1 pair A+ of the Line Interface**: Connects to the Pair A+ input of the transformer. On reset, set to high impedance.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
MDI1_0_n	C16	B22	A-Inout			**Port 1 pair A- of the Line Interface**: Connects to the Pair A- input of the transformer. On reset, set to high impedance.
MDI1_1_p	A14	A20	A-Inout			**Port 1 pair B+ of the Line Interface**: Connects to the Pair B+ input of the transformer. On reset, set to high impedance.
MDI1_1_n	B14	B20	A-Inout			**Port 1 pair B- of the Line Interface**: Connects to the Pair B- input of the transformer. On reset, set to high impedance.
MDI1_2_p	A12	A18	A-Inout			**Port 1 pair C+ of the Line Interface**: Connects to the Pair C+ input of the transformer. On reset, set to high impedance.
MDI1_2_n	B12	B18	A-Inout			**Port 1 pair C- of the Line Interface**: Connects to the Pair C- input of the transformer. On reset, set to high impedance.
MDI1_3_p	A11	A16	A-Inout			**Port 1 pair D+ of the Line Interface**: Connects to the Pair D+ input of the transformer. On reset, set to high impedance.
MDI1_3_n	B11	B16	A-Inout			**Port 1 pair D- of the Line Interface**: Connect to the pair D- input of the transformer. On reset, set to high impedance.
MDI1_4_p	A9	A14	A-Inout			**Port 1 Analog Test+**: Connects to the pair E+ input of the transformer.
MDI1_4_n	B9	B14	A-Inout			**Port 1 Analog Test-**: Connects to the pair E- input of the transformer.
BG_REXT	B15	D12	A-Inout			Connection point for the band-gap reference resistor. Should be a precision 1% 2 KΩ resistor tied to ground.
XTAL_I	D16	D23	A-In			Positive 50.0 MHz crystal oscillator input.
XTAL_O	E16	D24	A-Out			Positive 50.0 MHz crystal oscillator output.
RSVDF5_VSS	N/A	F5	In			

2.2.3 Serial Flash

See AC/DC specifications in Section 12.4.4.4.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
FLSH_SI	K3	K2	Out	Pup		**Flash Serial Output**: Serial data output to the Flash (used as ENCRYPTION_EN strap in X550-AT2).
FLSH_SO	J3	K1	In			**Flash Serial Input**: Serial data input from the Flash.
FLSH_SCK	L1	J1	Out			**Flash serial clock**: Operates at the maximum frequency of 25 MHz. This pin acts as a JTAG_DIS strap.
FLSH_CE_N	L2	J2	Out		Pup[1]	**Flash Chip Select Output**

1. Pup value should be considered as 3.3 KΩ.

2.2.4 SMBus

See the AC/DC specifications in Section 12.4.4.3.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
SMBCLK	F3	L2	O/d		Pup[1]	**SMBus Clock**: One clock pulse is generated for each data bit transferred.
SMBD	F2	L1	O/d		Pup[1]	**SMBus Data**: Stable during the high period of the clock (unless it is a start or stop condition).
SMBALRT_N	F1	M2	O/d		Pup[1]	**SMBus Alert**: Acts as an interrupt pin of a slave device on the SMBus.

1. Pup value should be considered as 10 KΩ.

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

2.2.5 NC-SI

See AC specifications in Section 12.4.4.5.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn[1]	Name and Function
NCSI_CLK_IN	D2	G2	NCSI-In		Pdn	**NC-SI Reference Clock Input**: Synchronous clock reference for receive, transmit, and control interface. It is a 50 MHz clock ±100 ppm.
NCSI_TX_EN	B1	G4	NCSI-In		Pup[2]	**MC Transmit Enable**: Indicates that received data from MC is valid.
NCSI_TXD0 NCSI_TXD1	D3 D4	H2 G3	NCSI-In		Pdn or Pup[2]	**MC Transmit Data**: Data signals from the MC to the X550.
NCSI_CRS_DV	E3	H1	NCSI-Out		Pdn[3]	**Carrier Sense/Receive Data Valid (CRS/DV) to MC**: Indicates that the data transmitted from the X550 to MC is valid.
NSCI_RXD0 NCSI_RXD1	E1 E2	H3 G1	NCSI-Out		Pup[2]	**MC Receive Data**: Data signals from the X550 to the MC.
NCSI_ARB_IN	C1	F1	NCSI-In		Pdn[3]	**NC-SI Arbitration In**
NCSI_ARB_OUT	D1	F2	NCSI-Out			**NC-SI Arbitration Out**

1. Pdn or Pup value should be considered as 10 KΩ.
2. Should be pulled up if NC-SI interface is disabled or if set to multi drop configuration.
3. Should be pulled down if NC-SI interface is disabled.

2.2.6 Software Defined Pins (SDPs)

See AC specifications in Section 12.4.4.1.

See Section 3.5 for more details on configurable SDPs.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
SDP0_0 SDP0_1 SDP0_2 SDP0_3	H1 G1 H2 G2	R4 P3 T4 R3	T/s	Pdn Pdn Pdn Pdn		**General Purpose SDPs — 3.3 V I/Os for Function 0**: Can be used to support IEEE1588 auxiliary devices. Input for external interrupts, PCIe function disablement, etc. See Section 3.5 for possible usages of the pins.
SDP1_0 SDP1_1 SDP1_2 SDP1_3	H16 G16 H15 G15	T21 T22 U21 U22	T/s	Pdn Pdn Pdn Pdn		**General Purpose SDPs — 3.3 V I/Os for Function 1**: Can be used to support IEEE1588 auxiliary devices. Input for external interrupts, PCIe function disablement, etc. See Section 3.5 for possible usages of the pins.

2.2.7 LEDs

See AC specifications in Section 12.4.4.1.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
LED0_0	K1	H4	Out	Pdn		**Port 0 LED0**: Programmable LED. By default, indicates link up.
LED0_1	J1	J3	Out	Pdn		**Port 0 LED1**: Programmable LED. By default, indicates 10 Gb/s link.
LED0_2	K2	J4	Out	Pdn		**Port 0 LED2**: Programmable LED. By default, indicates link/activity.
LED0_3	J2	K4	Out	Pdn		**Port 0 LED3**: Programmable LED. By default, indicates 1 Gb/s link.
LED1_0	K16	J21	Out	Pdn		**Port 1 LED0**: Programmable LED. By default, indicates link up.
LED1_1	J16	J22	Out	Pdn		**Port 1 LED1**: Programmable LED. By default, indicates 10 Gb/s link.
LED1_2	K15	K21	Out	Pdn		**Port 1 LED2**: Programmable LED. By default, indicates link/activity.
LED1_3	J15	K22	Out	Pdn		**Port 1 LED3**: Programmable LED. By default, indicates 1 Gb/s link.

2.2.8 RSVD and No-Connect Pins

Connecting RSVD pins based on naming convention:

- NC — Pin is not connected in the package.
- RSVD_NC — Reserved pin. Should be left unconnected.
- RSVD_VSS — Reserved pin. Should be connected to GND.
- RSVD_VCC — Reserved pin. Should be connected to VCC3P3.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Name and Function
RSVDD14_NC	N/A	D14	A-Inout	Reserved/No-connect pin.
RSVDG21_NC RSVDG22_NC	N/A	G21 G22	A-Inout A-Inout	Reserved/No-connect pins.
RSVDL4_NC RSVDL3_NC RSVDM4_NC RSVDM3_NC RSVDN4_NC RSVDN3_NC RSVDN2_NC RSVDP4_NC RSVDL22_NC RSVDL21_NC RSVDM22_NC RSVDM21_NC RSVDN22_NC RSVDN21_NC RSVDP22_NC RSVDP21_NC	N/A	L4 L3 M4 M3 N4 N3 N2 P4 L22 L21 M22 M21 N22 N21 P22 P21	Out Out Out Out Out Out Out Out Out Out Out Out Out Out Out Out	Reserved/No-connect pins.
RSVDT23_VSS RSVDT24_VSS RSVDP23_NC RSVDP24_VSS	N/A	T23 T24 P23 P24	Inout Inout Inout Inout	Reserved VSS pins.
RSVDL15_NC RSVDL16_NC	L15 L16	N/A	Inout Inout	Reserved/No-connect pins.
RSVDR22_NC	N/A	R22	Out	Reserved/No-connect pin.
RSVDAA6_NC RSVDAA8_NC RSVDAA10_NC RSVDAA14_NC RSVDAA16_NC RSVDAA18_NC RSVDK3_NC RSVDM24_NC RSVDT2_NC RSVDV21_NC RSVDY21_NC	N/A	AA6 AA8 AA10 AA14 AA16 AA18 K3 M24 T2 V21 Y21		Reserved/No-connect pins.
RSVDG11_NC	N/A	G11	PWR	Reserved/No-connect pin.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Name and Function
RSVDU3_NC RSVDR21_NC RSVDU2_NC RSVDT3_NC RSVDV24_NC RSVDU24_NC RSVDU23_NC RSVDV23_NC	N/A	U3 R21 U2 T3 V24 U24 U23 V23	T/s T/s T/s T/s T/s T/s T/s T/s	Reserved/No-connect pin.
RSVDC1_NC RSVDD1_NC RSVDE1_NC RSVDE23_NC RSVDE24_NC RSVDF24_NC RSVDG24_VSS RSVDH22_VSS RSVDY20_NC	N/A	C1 D1 E1 E23 E24 F24 G24 H22 Y20		Reserved/No-connect pin.
RSVDN24_NC RSVDW24_NC	N/A	N24 W24		Reserved/No-connect pin.
RSVDL23_NC	N/A	L23	O/d	Reserved/No-connect pin.
RSVDJ23_VSS	N/A	J23	In-Only	Reserved VSS pin.
RSVDJ24_VSS	N/A	J24	In	Reserved VSS pin.
RSVDH24_VSS	N/A	H24	In	Reserved VSS pin.
RSVDG23_VSS	N/A	G23	In-Only	Reserved VSS pin.
RSVDAD1_VSS	N/A	AD1	In	Reserved VSS pin.v
RSVDD13_NC RSVDC13_NC RSVDC11_NC RSVDD11_NC RSVDA1_NC RSVDA24_NC RSVDAD24_NC RSVDT1_NC RSVDU1_NC RSVDN13_NC RSVDU4_NC	N/A	D13 C13 C11 D11 A1 A24 AD24 T1 U1 N13 U4	A-Inout A-Inout A-Out A-Out In-Only In-Only In-Only PWR-O PWR-O PWR-O PWR-O	Reserved/No-connect pin.
RSVDR24_NC RSVDR23_NC RSVDN1_NC	N/A	R24 R23 N1	LVDS-O LVDS-O Out-Only	Reserved/No-connect pin.
RSVDP1_NC RSVDR1_NC	N/A	P1 R1	Inout Out	Reserved/No-connect pin.
RSVDM23_NC RSVDN23_NC	N/A	M23 N23	Inout Out	Reserved/No-connect pin.
RSVDL14_NC RSVDM14_NC	L14 M14	N/A	Inout Out	Reserved/No-connect pin.
RSVDV4_NC RSVDV3_NC	N/A	V4 V3	A-Out	Reserved/No-connect pin.
RSVDT1_NC RSVDR1_NC	T1 R1	N/A	A-Out	Reserved/No-connect pin.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Name and Function
RSVDV1_NC RSVDV2_NC	N/A	V1 V2	A-Out	Reserved/No-connect pin.
RSVDR2_NC RSVDP2_NC	R2 P2	N/A	A-Out	Reserved/No-connect pin.

2.2.8.1 Pin Differences in the X550-AT Single Port Device

NC = Pin is not connected in the package.

Name	Ball # (X550-AT2)	Connection (X550-AT)
MDI1_0_p	B16	NC
MDI1_0_n	C16	NC
MDI1_1_p	A14	NC
MDI1_1_n	B14	NC
MDI1_2_p	A12	NC
MDI1_2_n	B12	NC
MDI1_3_p	A11	NC
MDI1_3_n	B11	NC
MDI1_4_p	A9	NC
MDI1_4_n	B9	NC
LAN1_DIS_N	M16	NC

For additional information on these pins in the X550-AT2, see Section 2.2.2.

2.2.9 Miscellaneous

See AC/DC specifications in Section 12.4.4.1.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
LAN_PWR_GOOD	N/A	L24	In	Pup	Pup[1]	**LAN Power Good**: A 3.3 V input signal. A transition from low to high initializes the X550 into operation. If not used (BYPASS_POR = 0b), an internal Power-on-Reset (POR) circuit triggers the X550 power up.
BYPASS_POR	N/A	H23	In	Pdn	Pdn[2]	**Bypass POR**
AUX_PWR	H3	P2	In		Note[3]	**Auxiliary Power Available**: When set, indicates that auxiliary power is available and the X550 should support $D3_{COLD}$ power state if enabled to do so. This pin is latched at the rising edge of LAN_PWR_GOOD.
MAIN_PWR_OK	G3	R2	In		Note[4]	**Main Power Good**: Indicates that platform main power is up. Must be connected externally.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
LAN1_DIS_N	M16 (Strap)	K24	In[5]	Pup	Pup[1]	**LAN 1 Disable**: This pin is a strapping pin latched at the rising edge of LAN_PWR_GOOD or PE_RST_N or In-Band PCIe Reset. If this pin is not connected or driven high during initialization, LAN 1 is enabled. If this pin is driven low during initialization, LAN 1 port is disabled.
LAN0_DIS_N	M15 (Strap)	K23	In[5]	Pup	Pup[1]	**LAN 0 Disable**: This pin is a strapping option pin latched at the rising edge of LAN_PWR_GOOD or PE_RST_N or In-Band PCIe Reset. If this pin is not connected or driven high during initialization, LAN 0 is enabled. If this pin is driven low during initialization, LAN 0 port is disabled.
ENCRYPTION_EN	Strap on FLSH_SI (K3)	M1	In	Pup	Pdn/Pup[1]	**Encryption Enable**: Enable/disable for the internal IPsec engines. When pulled up, encryption features are enabled.
THERM_D0_P THERM_D0_N	C3 C2	F4 F3	A-Inout A-Inout			**Thermal Diode Reference**: Can be used to measure on-die temperature.

1. Pup value should be considered as 1 KΩ.
2. Pdn value should be considered as 1 KΩ.
3. Connect AUX_PWR signal to Pup if AUX power is available. Connect Pdn if AUX power is not available. Pup/Pdn value should be considered as 1 KΩ.
4. Connect MAIN_PWR_OK signal to Main Power through Pup resistor. Pup value should be considered as 10 KΩ.
5. For the X550-AT and X550-AT2, this pin is input during PCIe_RST_N assertion only, and is output after that.

2.2.10 JTAG

See AC specifications in Section 12.4.4.2.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
JTCK	F16	Y22	In-Only	Pup	Pdn[1]	**JTAG Clock Input**
JTDI	F14	W22	In-Only	Pup	Pup[2]	**JTAG Data Input**
JTDO	F15	V22	Out		Pup[3]	**JTAG Data Output**
JTMS	G14	W21	In-Only	Pup	Pup[2]	**JTAG TMS Input**
JTRST_N	H14	W23	In-Only	Pup	Pdn[1]	**JTAG Reset Input**: Active low reset for the JTAG port.

1. Pdn value should be considered as 470 Ω.
2. Pup value should be considered as 10 KΩ.
3. Pup value should be considered as 3.3 KΩ.

Note: If the JTAG is disconnected, use the external pull-up or pull-down values listed.

2.2.11 Power Supplies

See AC specifications in Section 12.3.1.

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Name and Function
RSVDH21_VSS	N/A	H21	PWR	Reserved power pin.
RSVDJ14_VSS	J14	N/A	PWR	Reserved power pin.
VSS	A1, A4, A7, A10, A13, A15, A16, B4, B7, B10, B13, C5, C7, C9, C11, C13, C15, D6, D8, D10, D12, D14, E5, E13, E15	B1, B2, D2, E2, C3, D3, B4, E4, C5, D5, B6, E6, C7, D7, B8, E8, C9, D9, B10, E10, B12, E12, B13, C14, E14, B15, D15, C16, E16, B17, D17, C18, E18, B19, D19, C20, E20, B21, D21, F21, C22, D22, E22, F22, B23, F23, B24, L6, L8, L10, L12, L14, L16, L18, L20, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, G6, G8, G10, G12, G14, G16, G18, G20, H5, H7, H9, H11, H13, H15, H17, H19, J6, J8, J10, J12, J14, J16, J18, J20, K5, K7, K9, K11, K13, K15, K17, K19	PWR_ALG	1.2 V ground.
VSS	N6, N7, N9, N10, N12, N13, N15, N16, P3, P5, P8, P11, P14, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, T2, T4, T7, T10, T13, T16	U6, V5, AA1, AC1, AA2, AC2, AD2, W3, W4, W6, Y3, AA3, AA4, AB4, AA5, AC5, AD5, AB6, Y7, AA7, AB7, AC8, AD8, Y9, AA9, AB9, AB10, Y11, AA11, AC11, AD11, AA12, AB12, Y13, AA13, AB13, AC14, AD14, Y15, AA15, AB15, AB16, Y17, AA17, AC17, AD17, AB18, Y19, AA19, AB19, AA20, AC20, AD20, AA21, AB21, AA22, AA23, AA24, AC23, AD23, AC24	PWR_ALG	Ground for PCIe.
VSS	F4, F6, F8, F10,F12, G5, G7, G9, G11, G13, H4, H6, H8, H10, H12, J5, J7, J9, J11, J13, K4, K6, K8, K10, K12, K14, L5, L7, L9, L11, M6, M8, M10, M12	M5, P5, T5, N6, R6, M7, P7, T7, V7, N8, R8, U8, W8, M9, P9, T9, V9, N10, R10, U10, W10, M11, P11, T11, V11, N12, R12, U12, W12, M13, P13, T13, V13, N14, R14, U14, W14, M15, P15, T15, V15, N16, R16, U16, W16, M17, P17, T17, V17, N18, R18, U18, W18, M19, P19, T19, V19, J20, N20, R20, U20, W20, Y23, Y24	PWR	Ground
NC	N/A	G5, G7, G9, G15, G17, G19, H6, H8, H10, H12, H14, H16, H18, H20, J5, J7, J9, J11, J15, J17, J19, K6, K8, K10, K12, K14, K16, K18, K20, L7, L9, L11, L13, L15, L17, L19		No-connect pins.
VCC0P83	F5, F7, F9, F11, F13, G4, G6, G8, G10, G12, H5, H7, H9, H11, H13, J4, J6, J8, J10, J12, K5, K7, K9, K11, K13, L6, L8, L10, L12, M5, M7, M9, M11	N7, R7, U7, W7, M8, P8, T8, V8, N9, R9, U9, W9, M10,M12, P10, T10, V10, N11, R11, U11, W11, P12, T12, V12, J13, R13, U13, W13, M14, P14, T14, V14, N15, R15, U15, W15, M16, P16, T16, V16, N17, R17, U17, W17, M18, P18, T18, V18	PWR	0.83 V
VCC1P2A	M1, M2, N1, N2	V6, W5	PWR_ALG	1.2 V
VCC1P2	E4, E6, E8	D10, D4, D8, E11, E7, E9, D6, E5	PWR_ALG	1.2 V
VCC1P2	E10, E12, E14	E19, D20, D16, D18, E13, E15, E17	PWR_ALG	1.2 V

Pin Name	Ball # (X550-AT2)	Ball # (X550-BT2)	Type	Name and Function
VCC2P1A0	C4, C6, C8, D5, D7	A2, A12, C2, A4, C4, A6, C6, A8, C8, A10, C10	PWR_ALG	2.1 V
VCC2P1A1	C10, C12, C14, D9, D11, D13	A13, A15, C15, A17, C17, A19, C19, A21, C21, A23, C23	PWR_ALG	2.1 V
VCC2P1A	N/A	C12	PWR_ALG	2.1 V
VSS	E11	E21	PWR_ALG	Ground
VSS	E7	E3	PWR_ALG	Ground
DVDD0P83	E9	G13	PWR_ALG	0.83 V
VCC3P3	L4, M4, L13, M13	L5, N5, R5, M6, P6, T6, N19, R19, U19, W19, M20, P20, T20, V20	PWR	3.3 V
VCCA	N3, N5, N8, N11, N14	AB3, AB5, Y6, Y8, AB8, Y10, AB11, Y12, Y14, AB14, Y16, AB17, Y18, AB20, AB22	PWR	1.0 V
VCC2P1	D15	C24	PWR_ALG	2.1 V
VCC2P1	P1	U5	PWR	2.1 V

2.3 Pull-Up/Pull-Down Information

2.3.1 External Pull-Ups

Table 2-1 lists external pull-up resistors and their functionality.

Table 2-1. External Pull-Up Resistors

Signal Name	External Pull-Up/ Pull-Down	Notes
RESET_N	Pull-up	In X550-BT2 only.
PE_WAKE_N	Pull-up	
FLSH_SCK	Pull-down	Should be pulled down for normal operation.
FLSH_SI	Pull-down/Pull-up	Serial data output to the Flash. In the X550-BT2, no need for a pull-up or a pull-down. In the X550-AT2, when pulled down, IPsec encryption features are disabled.
SMBD	Pull-up	
SMBCLK	Pull-up	
SMBALRT_N	Pull-up	
NCSI_CLK_IN	Pull-down	Should be pulled down if NC-SI interface is disabled.
NCSI_CRS_DV	Pull-down	Only if NC-SI is unused or set to multi drop configuration.
NCSI_RXD[1:0]	Pull-up	
NCSI_TX_EN	Pull-down	Should be pulled down if NC-SI interface is disabled.
NCSI_TXD[1:0]	Pull-down	
JTCK	Pull-down	These resistors should be connected if JTAG is not used. See Section 2.2.10 for details.
JTDI	Pull-up	
JTDO	Pull-up	
JTMS	Pull-up	
JTRST_N	Pull-down	
ENCRYPTION_EN	Pull-up	When pulled up, IPsec encryption features are enabled.

2.4 Strapping Options

Table 2-2 lists the different strapping pins and their sampling point in both packages.

Table 2-2. Strapping Options

Strap	X550-BT2 Location	X550-AT2 Location	Latched At
LAN0_DIS_N	LAN0_DIS_N (K23)	LAN0_DIS_N (M15)	LAN_PWR_GOOD, PE_RST_N, In-band Reset.
LAN1_DIS_N	LAN1_DIS_N (K24)	LAN1_DIS_N (M16)	LAN_PWR_GOOD, PE_RST_N, In-band Reset.
ENCRYPTION_EN	ENCRYPTION_EN (M1)	FLSH_SI (K3)	LAN_PWR_GOOD

2.5 Ball Out — Top View Through Package

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A	VSS	MDI0_0_p	MDI0_1_p	VSS	MDI0_2_p	MDI0_3_p	VSS	MDI0_4_p	MDI1_4_p	VSS	MDI1_3_p	MDI1_2_p	VSS	MDI1_1_p	VSS	VSS	A
B	NCSI_TX_EN	MDI0_0_n	MDI0_1_n	VSS	MDI0_2_n	MDI0_3_n	VSS	MDI0_4_n	MDI1_4_n	VSS	MDI1_3_n	MDI1_2_n	VSS	MDI1_1_n	BG_REXT	MDI1_0_p	B
C	NCSI_ARB_IN	THERM_D0_N	THERM_D0_P	VCC2P1A0	VSS	VCC2P1A0	VSS	VCC2P1A0	VSS	VCC2P1A1	VSS	VCC2P1A1	VSS	VCC2P1A1	VSS	MDI1_0_n	C
D	NCSI_ARB_OUT	NCSI_CLK_IN	NCSI_TXD0	NCSI_TXD1	VCC2P1A0	VSS	VCC2P1A0	VSS	VCC2P1A1	VSS	VCC2P1A1	VSS	VCC2P1A1	VSS	VCC2P1	XTAL_I	D
E	NSCI_RXD0	NCSI_RXD1	NCSI_CRS_DV	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	DVDD0P83	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	XTAL_O	E
F	SMBALRT_N	SMBD	SMBCLK	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	JTDI	JTDO	JTCK	F
G	SDP0_1	SDP0_3	MAIN_PWR_OK	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	JTMS	SDP1_3	SDP1_1	G
H	SDP0_0	SDP0_2	AUX_PWR	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	JTRST_N	SDP1_2	SDP1_0	H
J	LED0_1	LED0_3	FLSH_SO	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	RSVDJ14_VSS	LED1_3	LED1_1	J
K	LED0_0	LED0_2	FLSH_SI ENCRYPTION_EN	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	LED1_2	LED1_0	K
L	FLSH_SCK	FLSH_CE_N	PE_WAKE_N	VCC3P3	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VCC3P3	RSVDL14_NC	RSVDL15_NC	RSVDL16_NC	L
M	VCC1P2A	VCC1P2A	PE_RST_N	VCC3P3	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC3P3	RSVDM14_NC	LAN0_DIS_N	LAN1_DIS_N	M
N	VCC1P2A	VCC1P2A	VCCA	PE_CLK_p	VCCA	VSS	VSS	VCCA	VSS	VSS	VCCA	VSS	VSS	VCCA	VSS	VSS	N
P	VCCP2P1	RSVDP2_NC	VSS	PE_CLK_n	VSS	PER_0_p	PER_0_n	VSS	PER_1_p	PER_1_n	VSS	PER_2_p	PER_2_n	VSS	PER_3_p	PER_3_n	P
R	RSVDR1_NC	RSVDR2_NC	PE_RSENSE	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	R
T	RSVDT1_NC	VSS	PE_RBIAS	VSS	PET_0_p	PET_0_n	VSS	PET_1_p	PET_1_n	VSS	PET_2_p	PET_2_n	VSS	PET_3_p	PET_3_n	VSS	T
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	

Figure 2-1. X550 Package Layout (X550-AT2)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
A	RSVDA1_NC	VCC2P1A0	MDI0_0_p	VCC2P1A0	MDI0_1_p	VCC2P1A0	MDI0_2_p	VCC2P1A0	MDI0_3_p	VCC2P1A0	MDI0_4_p	VCC2P1A0	VCC2P1A1	MDI1_4_p	VCC2P1A1	MDI1_3_p	VCC2P1A1	MDI1_2_p	VCC2P1A1	MDI1_1_p	VCC2P1A1	MDI1_0_p	VCC2P1A1	RSVDA24_NC	A
B	VSS	VSS	MDI0_0_n	VSS	MDI0_1_n	VSS	MDI0_2_n	VSS	MDI0_3_n	VSS	MDI0_4_n	VSS	VSS	MDI1_4_n	VSS	MDI1_3_n	VSS	MDI1_2_n	VSS	MDI1_1_n	VSS	MDI1_0_n	VSS	VSS	B
C	RSVDC1_NC	VCC2P1A0	VSS	VCC2P1A0	VSS	VCC2P1A0	VSS	VCC2P1A0	VSS	VCC2P1A0	RSVDC11_NC	VCC2P1A	RSVDC13_NC	VSS	VCC2P1A1	VSS	VCC2P1A1	VSS	VCC2P1A1	VSS	VCC2P1A1	VSS	VCC2P1A1	VCC2P1	C
D	RSVDD1_NC	VSS	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	RSVDD11_NC	BG_REXT	RSVDD13_NC	RSVDD14_NC	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VSS	XTAL_I	XTAL_O	D
E	RSVDE1_NC	VSS	VSS	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VSS	VSS	RSVDE23_NC	RSVDE24_NC	E
F	NCSI_ARB_IN	NCSI_ARB_OUT	THERM_DD_N	THERM_DD_P	RSVDF5_VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	RSVDF24_NC	F
G	NCSI_RXD1	NCSI_CLK_IN	NCSI_TXD1	NCSI_TX_EN	NC	VSS	NC	VSS	NC	VSS	RSVDG11_NC	VSS	DVDDOPB3	VSS	NC	VSS	NC	VSS	NC	VSS	RSVDG21_NC	RSVDG22_NC	RSVDG23_VSS	RSVDG24_VSS	G
H	NCSI_CRS_DV	NCSI_TXD0	NSCI_RXD0	LED0_0	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	RSVDH21_VSS	RSVDH22_VSS	BYPASS_POR	RSVDH24_VSS	H
J	FLSH_SCK	FLSH_CE_N	LED0_1	LED0_2	NC	VSS	NC	VSS	NC	VSS	NC	VSS	VCC0P83	VSS	NC	VSS	NC	VSS	NC	VSS	LED1_0	LED1_1	RSVDJ23_VSS	RSVDJ24_VSS	J
K	FLSH_SO	FLSH_SI	RSVDK3_NC	LED0_3	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	LED1_2	LED1_3	LAN0_DIS_N	LAN1_DIS_N	K
L	SMBD	SMBCLK	RSVDL3_NC	RSVDL4_NC	VCC3P3	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	NC	VSS	RSVDL21_NC	RSVDL22_NC	RSVDL23_NC	LAN_PWR_GOOD	L
M	ENCRYPTION_EN	SMBALRT_N	RSVDM3_NC	RSVDM4_NC	VSS	VCC3P3	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC3P3	RSVDM21_NC	RSVDM22_NC	RSVDM23_NC	RSVDM24_NC	M
N	RSVDN1_NC	RSVDN2_NC	RSVDN3_NC	RSVDN4_NC	VCC3P3	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	RSVDN13_NC	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC3P3	VSS	RSVDN21_NC	RSVDN22_NC	RSVDN23_NC	RSVDN24_NC	N
P	RSVDP1_NC	AUX_PWR	SDP0_1	RSVDP4_NC	VSS	VCC3P3	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC3P3	RSVDP21_NC	RSVDP22_NC	RSVDP23_VSS	RSVDP24_VSS	P
R	RSVDR1_NC	MAIN_PWR_OK	SDP0_3	SDP0_0	VCC3P3	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC3P3	VSS	RSVDR21_NC	RSVDR22_NC	RSVDR23_NC	RSVDR24_NC	R
T	RSVDT1_NC	RSVDT2_NC	RSVDT3_NC	SDP0_2	VSS	VCC3P3	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC3P3	SDP1_0	SDP1_1	RSVDT23_VSS	RSVDT24_VSS	T
U	RSVDU1_NC	RSVDU2_NC	RSVDU3_NC	RSVDU4_NC	VCC2P1	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC3P3	VSS	SDP1_2	SDP1_3	RSVDU23_NC	RSVDV24_NC	U
V	RSVDV1_NC	RSVDV2_NC	RSVDV3_NC	RSVDV4_NC	VSS	VCC1P2A	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC3P3	RSVDV21_NC	JTDO	RSVDU23_NC	RSVDV24_NC	V
W	PE_WAKE_N	PE_RST_N	VSS	VSS	VCC1P2A	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC0P83	VSS	VCC3P3	VSS	JTMS	JTDI	JTRST_N	RSVDW24_NC	W
Y	PE_CLK_n	PE_CLK_p	VSS	PE_RBIAS	PE_RSENSE	VCCA	VSS	VCCA	VSS	VCCA	VSS	VCCA	VSS	VCCA	VSS	VCCA	VSS	VCCA	VSS	RSVDY20_NC	RSVDY21_NC	JTCK	VSS	VSS	Y
AA	VSS	VSS	VSS	VSS	VSS	RSVDAA6_NC	VSS	RSVDAA8_NC	VSS	RSVDAA10_NC	VSS	VSS	VSS	RSVDAA14_NC	VSS	RSVDAA16_NC	VSS	RSVDAA18_NC	VSS	VSS	VSS	VSS	VSS	VSS	AA
AB	PER_0_n	PER_0_p	VCCA	VSS	VCCA	VSS	VSS	VCCA	VSS	VSS	VCCA	VSS	VSS	VCCA	VSS	VSS	VCCA	VSS	VSS	VCCA	VSS	VCCA	PER_7_p	PER_7_n	AB
AC	VSS	VSS	PET_0_p	PET_1_p	VSS	PER_1_n	PER_2_n	VSS	PET_2_p	PET_3_p	VSS	PER_3_n	PER_4_n	VSS	PET_4_p	PET_5_p	VSS	PER_5_n	PER_6_n	VSS	PET_6_p	PET_7_p	VSS	VSS	AC
AD	RSVDAD1_VSS	VSS	PET_0_n	PET_1_n	VSS	PER_1_p	PER_2_p	VSS	PET_2_n	PET_3_n	VSS	PER_3_p	PER_4_p	VSS	PET_4_n	PET_5_n	VSS	PER_5_p	PER_6_p	VSS	PET_6_n	PET_7_n	VSS	RSVDAD24_NC	AD

Figure 2-2. X550 Package Layout (X550-BT2)

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

Chapter 3 Interconnects

3.1 PCI Express (PCIe)

3.1.1 General Overview

The X550 supports Rev 3.0 of the PCIe base specification.

On top of the capabilities required by the PCIe specifications, The X550 supports optional functionality as listed in this section:

- All PCI functions are native PCIe functions.
- Physical Layer:
  - Support for 2.5 GT/s, 5 GT/s, and 8 GT/s
  - Interface width of 1, 4 or 8 PCIe lanes (8 lanes are supported only in the X550-BT2, and only at 5 GT/s or 2.5 GT/s speeds.)
  - Full swing and half swing signaling
  - Lane reversal
- Transaction layer mechanisms:
  - 64-bit and 32-bit memory address spaces
  - Removal of I/O BAR (optional)
  - Relaxed ordering
  - Flow control update timeout mechanism
  - ID-based ordering (IDO)
  - Packet sizes: Maximum packet size: 512, Maximum read request size: 2 KB
  - Function-Level Reset (FLR)
  - TLP Processing Hints (TPH)
- Reliability:
  - Advanced Error Reporting (AER)
  - End-to-End CRC (ECRC) generation and checking
  - Recovery from data poisoning
  - Completion Timeout
- Power management:
  - Active state power management (L1 state)
  - Wake capability
  - Latency Tolerance Reporting (LTR)
- DFT and DFM support for high-volume manufacturing.

- The X550 supports the following Extended Capabilities:
  — Advanced Error Reporting (AER)
  — Device Serial Number
  — 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 Transaction Types Accepted by the X550

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

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

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

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

3.1.2.1.1 Size of Target Accesses

3.1.2.1.1.1 Memory Accesses

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

Write accesses:

- — Zero-length writes have no internal impact (nothing written, no effect such as clear-by-write). The transaction is treated as a successful operation (no error event).
- — CSR writes are 32-bit or 64-bit only. Larger or partial CSR writes are handled as Completer Abort - data is dropped and an error is generated per PCIe rules.

- Read accesses:
  - — Partial reads with at least one byte disabled are handled as a full read. Any side effect of the full read (such as clear by read) is also applicable to partial reads. The completion on PCIe obeys the specification rules regarding the number of bytes reported in the completion.
  - — Zero-length reads generate a completion, but the register is not accessed and undefined data is returned.
  - — CSR reads are 32-bit or 64-bit only. Larger CSR read requests are handled as Completer Abort. The completion includes a CA status and an error is generated per PCIe rules.
  - — Some 64-bit reads are handled atomically (i.e. not interleaved with any other requests). This applies mainly to reading counters, where all 64-bit need to be read simultaneously. Such registers are explicitly marked in their description.

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 obeys 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.

3.1.2.1.1.2 I/O Accesses

Rules for accesses to the I/O BAR:

- Write accesses:
  - Write accesses are 32-bit wide.
  - Zero-length writes have no internal impact (nothing written, no effect such as clear-by-write). The transaction is treated as a successful operation (no error event).
  - Other accesses (partial writes, larger writes) are handled as 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 obeys 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.

3.1.2.1.1.3 Messages

MCTP messages may contain payload of up to 64 bytes.

3.1.2.1.2 Support for Dynamic Changes

The X550 captures the Bus Number and Device Number per each Configuration Write Request. However, dynamic change of Bus Number or Device Number is not supported. Rather, the PCIe link should be quiescent prior to such a change, including reception of all completion for previous requests.

3.1.2.2 Transaction Types Initiated by the X550

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

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

1. MCTP messages contain payload.

Configuration values:

- Max Payload Size — The value of the *Max Payload Size Supported* field in the Device Capabilities Register is loaded from NVM.
  - Hardware default is 512.
  - 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 X550 supports read requests of up to 2 KB.
  - The actual maximum size of a read request is defined as the minimum {2 KB, *Max Read Request Size* field in the Device Control Register}.

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

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

3.1.2.2.1 Data Alignment

Requests must never specify an address/length combination that causes a memory space access to cross a 4 KB boundary. The X550 therefore breaks requests into 4 KB-aligned requests (if needed). This does not place any requirement on software. However, if software allocates a buffer across a 4 KB boundary, hardware issues multiple requests for the buffer. Software should consider aligning buffers to a 4 KB boundary in cases where it improves performance. The maximum size of a read request is defined as the minimum {2 KB bytes, *Max_Read_Request_Size*}

The general rules for packet alignment are as follows. Note that these apply to all the X550 requests (read/write):

- The length of a single request does not exceed the PCIe limit of *MAX_PAYLOAD_SIZE* for write and *MAX_READ_REQ* for read.
- The length of a single request does not exceed the X550 internal limitations.
- A single request does not span across different memory pages as noted by the 4 KB boundary alignment previously mentioned.

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

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

The system cache line size is controlled by the PCI_CNF2.*CACHELINE_SIZE* bit, loaded from the NVM. Note that the Cache Line Size Register in the PCI configuration space is not related to the PCI_CNF2.*CACHELINE_SIZE* and is solely for software use.

3.1.2.3 Messages

3.1.2.3.1 Received Messages

Message packets are special packets that carry a message code. The upstream device transmits special messages to the X550 by using this mechanism. The transaction layer decodes the message code and responds to the message accordingly.

Table 3-3. Supported Messages in the X550 — as a Receiver

Message Code [7:0]	Routing r2r1r0	Message	X550 Later Response
0x14	100b	PM_Active_State_NAK	Accepted
0x19	011b	PME_Turn_Off	Accepted
0x40, 0x41, 0x43, 0x44, 0x45, 0x47, 0x48	100b	Ignored messages (used to be hot-plug messages)	Silently drop
0x50	100b	Slot power limit support (has one DWord data)	Silently drop.
0x7E	000b, 010b, 011b, 100b	Vendor defined type 0	Drop and handle as an Unsupported request.
0x7F	100b	Vendor defined type 1	Silently drop.
0x7F	010b, 011b, 000b	Vendor defined type 1	Send to MCTP reassembly if Vendor ID = 0x1AB4 (DMTF) and VDM code = 0000b (MCTP). Otherwise, silently drop.
0x00	011b	Unlock	Silently drop.

3.1.2.3.2 Transmitted Messages

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

Table 3-4. Supported Messages in the X550 — as a Transmitter

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

Table 3-4. Supported Messages in the X550 — as a Transmitter [continued]

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

3.1.2.3.3 Vendor Defined Messages (VDM)

The following vendor defined messages are supported:

- DMTF MCTP

3.1.2.3.3.1 MCTP VDMs

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

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

Table 3-5. MCTP Over PCIe Header Format

<table>
<tr><td>FMT
011</td><td>Type
10r2r1r0[1]</td><td>R</td><td>TC
000</td><td>R</td><td>Attr[2]</td><td>R</td><td>TH[2]</td><td>TD[2]</td><td>EP[2]</td><td>Attr[1:0][2]</td><td>AT
00</td><td colspan="2">Length
00_000x_xxxx</td></tr>
<tr><td colspan="8" rowspan="2">PCI Requester ID</td><td colspan="3">PCI Tag Field</td><td rowspan="2">Message Code
Vendor Defined = 0111_1111b</td></tr>
<tr><td>R</td><td>Pad Len</td><td>MCTP VDM code - 0000b</td></tr>
<tr><td colspan="8">PCI Target ID (For Route by ID messages, otherwise = Reserved)</td><td colspan="4">Vendor ID = 0x1AB4 (DMTF)</td></tr>
</table>

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

3.1.2.4 Transaction Attributes

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

The X550 only supports TC = 0 and VC = 0 (default).

3.1.2.4.2 TLP Processing Hints (TPH)

The X550 supports the TPH capability defined in the PCI Express specification. It does not support Extended TPH requests.

Existence of a TLP Processing Hint (TPH) is indicated on the PCIe link by setting the TH bit in the TLP header. Using the PCIe TLP Steering Tag (ST) and Processing Hints (PH) fields, The X550 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 X550 exposes a PCIe TPH capability structure (See Section 9.2.4.5) with no steering table.

See Section 7.5 for details on the usage of TPH.

3.1.2.5 Ordering Rules

The X550 meets the PCIe ordering rules by following the PCI simple device model:

1. Deadlock Avoidance — The X550 meets the PCIe ordering rules that prevent deadlocks:
    a. Posted writes overtake stalled read requests. This applies to both target and master directions. For example, if master read requests are stalled due to lack of credits, master posted writes are allowed to proceed. On the target side, it is acceptable to timeout on stalled read requests to allow later posted writes to proceed.
    b. Target posted writes overtake stalled target configuration writes.
    c. 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 X550 are allowed to proceed.
2. Descriptor/Data Ordering — The X550 ensures that a Rx descriptor is written back on PCIe only after the data that the descriptor relates to is written to the PCIe link.MSI and MSI-X Ordering Rules. System software might change the MSI or MSI-X tables during run-time. Software expects that interrupt messages issued after the table has been updated are using the updated contents of the tables.
    a. Since software does not know when the tables are actually updated in the X550, a common scheme is to issue a read request to the MSI or MSI-X table (a PCI configuration read for MSI and a memory read for MSI-X). Software expects that any message issued following the completion of the read request, is using the updated contents of the tables.
    b. Once an MSI or MSI-X message is issued using the updated contents of the interrupt tables, any consecutive MSI or MSI-X message does not use the contents of the tables prior to the change.
3. The X550 meets the rules relating to independence between target and master accesses:
    a. The acceptance of a target posted request does not depend upon the transmission of any TLP.
    b. The acceptance of a target non-posted request does not depend upon the transmission of a non-posted request.
    c. Accepting a completion does not depend upon the transmission of any TLP.

3.1.2.5.1 Relaxed Ordering

The X550 takes advantage of the relaxed ordering rules in PCIe. By setting the relaxed ordering bit in the packet header, the X550 enables the system to optimize performance in the following cases:

1. Relaxed ordering for descriptor and data reads — When the X550 masters a read transaction, its split completion has no ordering relationship with the writes from the CPUs (same direction). It should be allowed to bypass the writes from the CPUs.
2. Relaxed ordering for receiving data writes — When the X550 masters receive data writes, it also enables them to bypass each other in the path to system memory because software does not process this data until their associated descriptor writes are done.
3. The X550 cannot relax ordering for receive descriptor writes or an MSI write.

Relaxed ordering is enabled in the X550 by clearing the CTRL_EXT.*RO_DIS* bit. Relaxed ordering is further controlled through the *Enable Relaxed Ordering* bit in the PCIe Device Control Register (Section 9.2.3.6.5).

3.1.2.5.2 ID-Based Ordering (IDO)

ID-based ordering was introduced in the PCIe rev. 2.1 specification. When enabled, the X550 sets IDO in all applicable TLPs defined in the PCIe specification.

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

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

- The NVM PCI_CAPSUP.*IDO Enable* bit is set (Section 6.2.6.12 and Section 8.2.2.5.11).
- The PCIe *IDO Request Enable* bit (for requests) or the *IDO Completion Enable* bit (for completions) in Device Control 2 Register is set (Section 9.2.3.6.11).

3.1.2.6 Flow Control

3.1.2.6.1 Flow Control Rules

The X550 only implements the default Virtual Channel (VC0). A single set of credits is maintained for VC0.

Table 3-6. Flow Control Credits Allocation

Credit Type	Operations	Number of Credits (Dual Port)
Posted Request Header (PH)	Target write Message (one unit)	16 credit units to support tail write at wire speed.
Posted Request Data (PD)	Target Write (Length/16 bytes = one) Message (one unit)	max{MAX_PAYLOAD_SIZE/16, 32}.
Non-Posted Request Header (NPH)	Target read (one unit) Configuration read (one unit) Configuration write (one unit)	Four credit units (to enable concurrent target accesses to both LAN ports).
Non-Posted Request Data (NPD)	Configuration write (one unit)	Four credit units.
Completion Header (CPLH)	Read completion (N/A)	Infinite (accepted immediately).
Completion Data (CPLD)	Read completion (N/A)	Infinite (accepted immediately).

Rules for FC updates:

- UpdateFC packets are sent immediately when a resource becomes available.
- The X550 follows the PCIe recommendations for frequency of UpdateFC FCPs.
- Specific rules apply in L0 or L0s link states. See PCIe specification.

3.1.2.6.2 Flow Control Timeout Mechanism

The X550 implements the optional flow control update timeout mechanism. See the PCIe specification.

3.1.2.7 End-to-End CRC (ECRC)

The X550 supports ECRC as defined in the PCIe specification. The following functionality is provided:

- Inserting ECRC in all transmitted TLPs:
  - The X550 indicates support for inserting ECRC in the *ECRC Generation Capable* bit of the PCIe configuration registers. This bit is loaded from the *ECRC Generation* NVM bit.
  - Inserting ECRC is enabled by the *ECRC Generation Enable* bit of the PCIe configuration registers. VFs follow the behavior of their PF.
    - ECRC is not added to MCTP messages (per MCTP specification) unless the PCI_CAPSUP.*ECRC_MCTP_GEN* bit is set.
- ECRC is checked on all incoming TLPs. A packet received with an ECRC error is dropped. Note that for completions, a completion timeout occurs later (if enabled).
  - The X550 indicates support for ECRC checking in the *ECRC Check Capable* bit of the PCIe configuration registers. This bit is loaded from the *ECRC Check* NVM bit.
  - Checking of ECRC is enabled by the *ECRC Check Enable* bit of the PCIe configuration registers. ECRC checking is done if enabled by at least one physical function (enablement is not done via VFs).
- ECRC errors are reported on all physical functions (PFs) enabled for ECRC checking.
- System software can configure ECRC independently per each physical function.

3.1.3 Link Layer

3.1.3.1 ACK/NAK Scheme

The X550 supports two alternative schemes for ACK/NAK rate:

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

3.1.3.2 Supported DLLPs

The following DLLPs are supported by the X550 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 X550 as a transmitter:

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

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

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

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

3.1.4 Physical Layer

3.1.4.1 Link Speed

The X550 supports PCIe 2.1 and PCIe 3.0.

The following configuration controls link speed:

- **PCIe Supported Link Speeds bit** — Indicates the link speeds supported by the X550.
- **PCIe Current Link Speed bit** — Indicates the negotiated link speed.
- **PCIe Target Link Speed bit** — used to set the target compliance mode speed when software is using the *Enter Compliance* bit to force a link into compliance mode. The default value is loaded from the highest link speed supported defined by the above *Supported Link Speeds*.

The X550 does not initiate a hardware autonomous speed change.

The X550 supports entering compliance mode at the speed indicated in the *Target Link Speed* field in the PCIe Link Control 2 register (Section 9.2.3.6.13). Compliance mode functionality is controlled via the PCIe Link Control 2 register.

3.1.4.2 Link Width

The X550 supports a maximum link width of x8, x4, or x1.The maximum link width is loaded into the *Max Link Width* field of the PCIe Capability register (*LCAP[11:6]*). Hardware default is the x4 link for the X550-AT2, and x8 link for the X550-BT2.

During link configuration, the platform and the X550 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, the X550 negotiates to either x8, x4 or x1.
- If Maximum Link Width = x4, the X550 negotiates to either x4 or x1
- If Maximum Link Width = x1, the X550 only negotiates to x1

When negotiating for x4, or x1 link, the X550 may negotiate the link to reside starting from physical lane 0 or starting from physical lane 4.

The X550 does not initiate a hardware autonomous link width change.

When operating in x8 link width the X550 does not support Gen3 link speed (8GT/s).

The x8 link width is available only in the X550-BT2.

3.1.4.3 Lane Configurations

The X550 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 X550 does not initiate lane reversal. The link partner (LP) may initiate lane reversal (to end up with an optimal lane width) and the X550 consents with the lane reversal.
- If lane 0 is found invalid, the X550 initiates lane reversal. Lane reversal succeeds if the link partner supports link reversal.
- If the lanes at both ends of the port (i.e. lanes 0 & 7 for x8, lanes 0 & 3 for x4, lane 0 for x1) are invalid, a link is not established.

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

Figure 3-1, Figure 3-2, and Figure 3-3 depict the initial link width configuration and link degradation options. In Figure 3-1 and Figure 3-2, the upper part of the Figure describes link options where the Link Partner (LP) and the X550 (NIC) are aligned. The bottom part of the Figure describes link options where the Link Partner and the X550 are reversed in order.

- Figure 3-1 applies when either the Link partner or the X550 is physically set to x8.
- Figure 3-2 applies when either the Link partner or the X550 is physically set to x4 and both are not physically set x8.
- Figure 3-3 applies when both the Link partner or the X550 is physically set to x1.

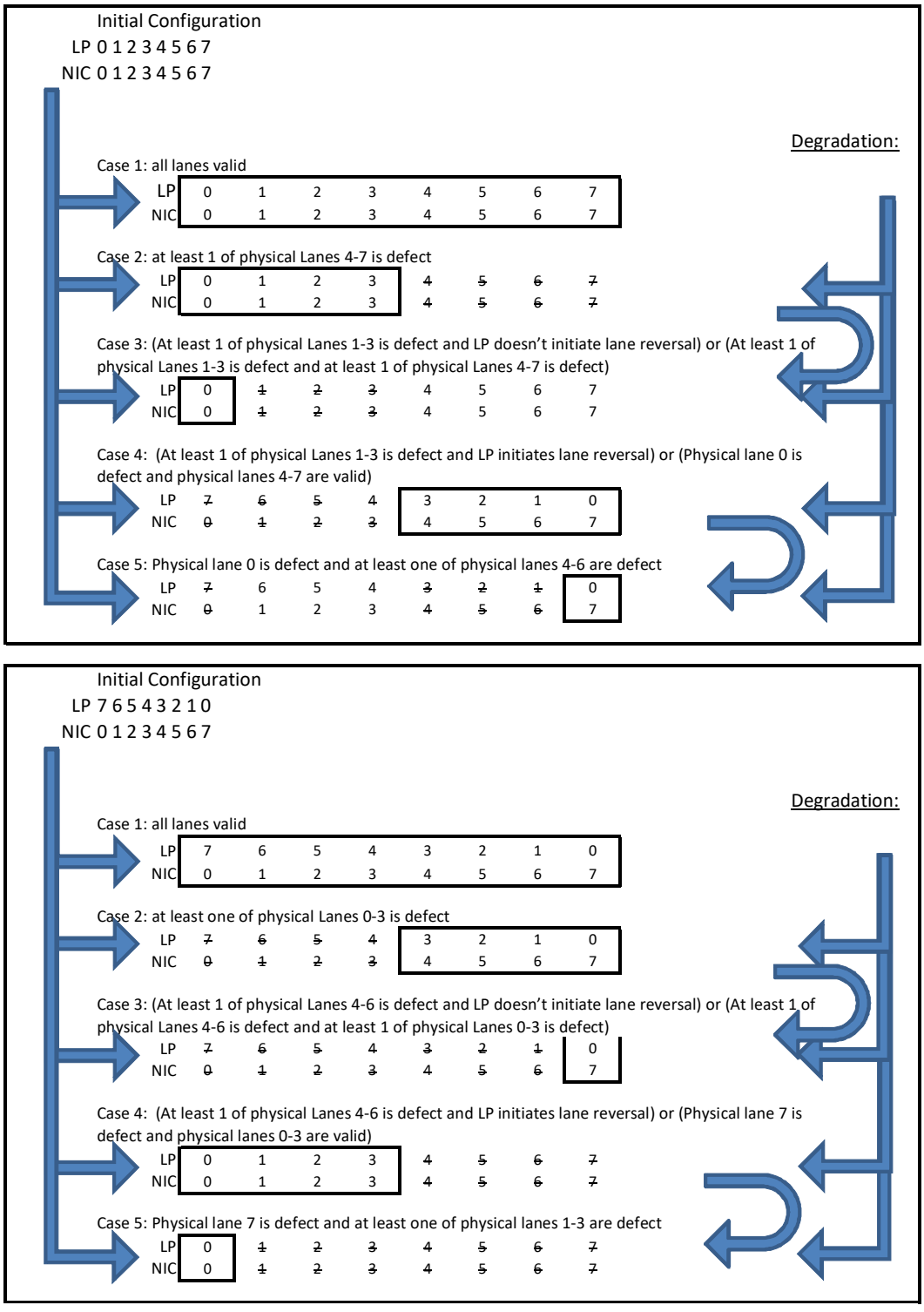


Figure 3-1. Link Width Configurations for a x8 Port

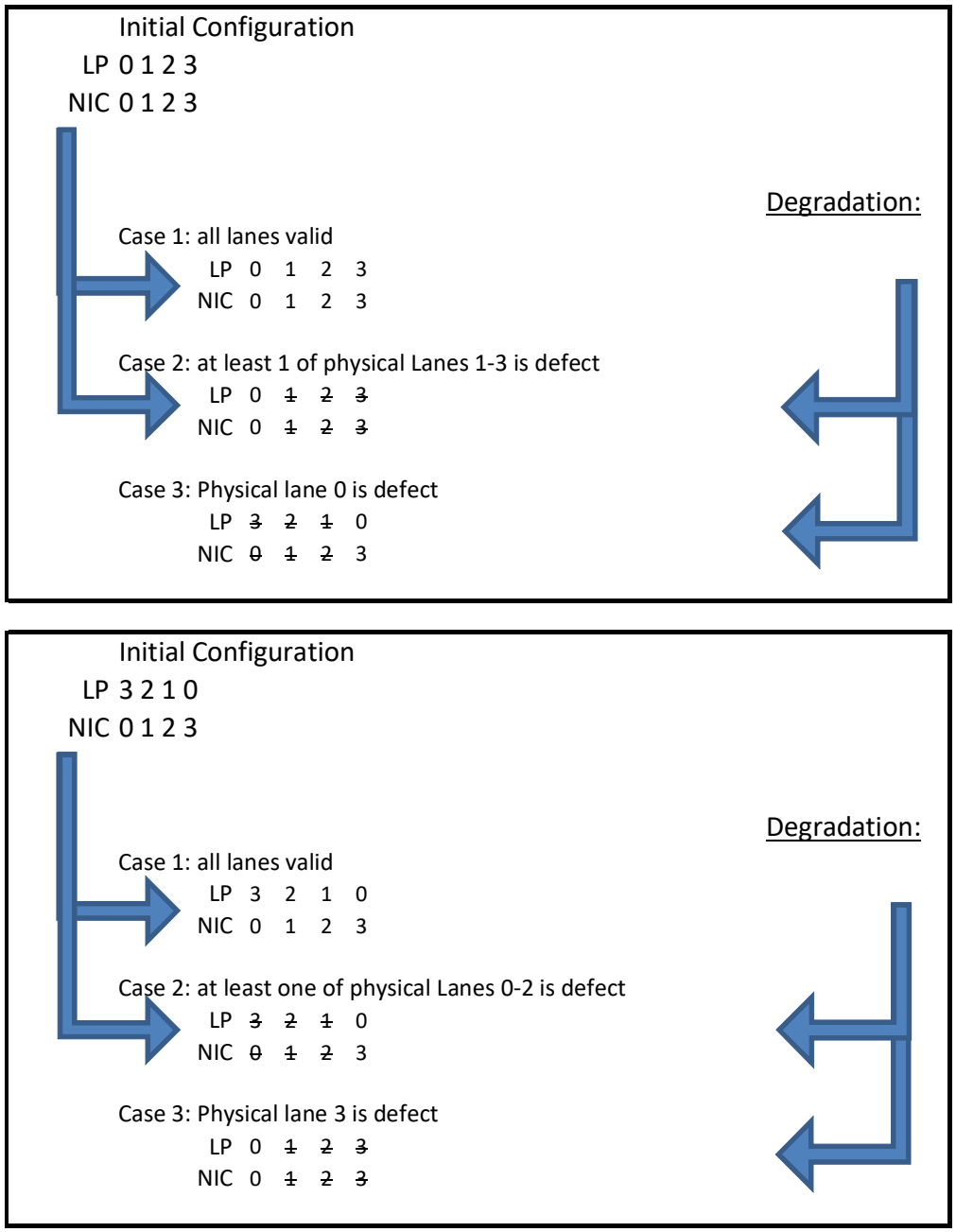


Figure 3-2. Link Width Configurations for a x4 Port

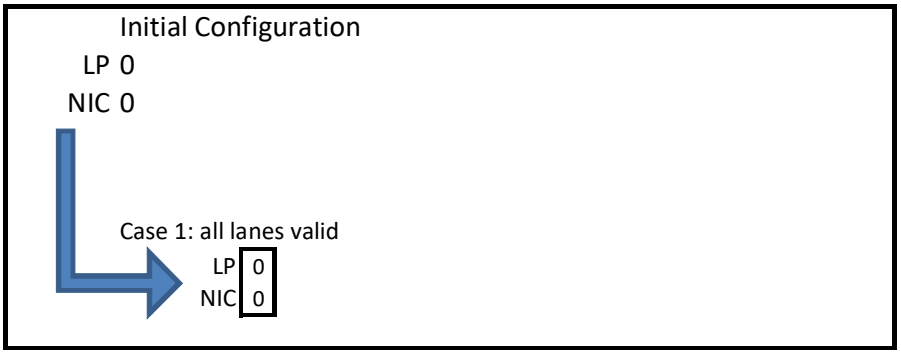


Figure 3-3. Link Width Configurations for a x1 Port

3.1.4.4 Receiver Framing Requirements

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

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

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

3.1.5 Error Events and Error Reporting

3.1.5.1 General Description

PCIe defines three error reporting paradigms: the baseline capability, the Advanced Error Reporting (AER) capability, and a proprietary mechanism. The baseline error reporting capabilities are required of all PCIe devices and define the minimum error reporting requirements. The AER capability is defined for more robust error reporting and is implemented with a specific PCIe capability structure. Both mechanisms are supported by the X550. The proprietary error reporting mechanism used for error better handled by the software device driver using internal CSRs is described in Section 3.1.5.8.

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

In a multi-function device, PCIe errors that are not related to any specific function within the device are logged in the corresponding status and logging registers of all functions in that device. Figure 3-4 shows, in detail, the flow of error reporting in the X550.

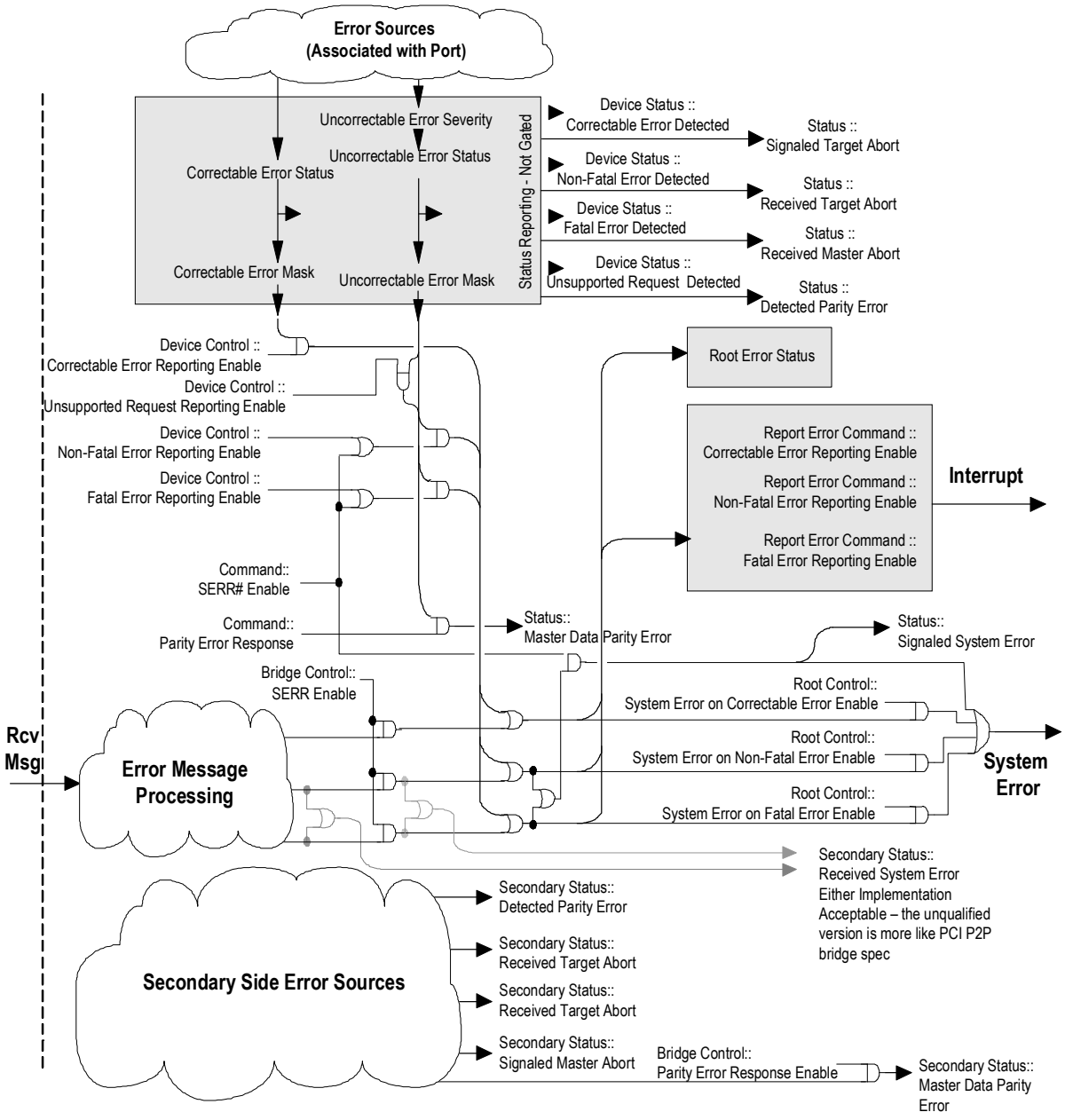


Figure 3-4. Error Reporting Mechanism

3.1.5.2 Error Events

Table 3-7 lists the error events identified by the X550 and the response in terms of logging, reporting, and actions taken. Refer to the PCIe specification for the effect on the PCI Status register.

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

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

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

Error Name	Error Events	Default Severity	Action
Completer Abort	• Received target access with illegal data size per Section 3.1.2.1.1.	Uncorrectable. ERR_NONFATAL Log header	Send completion with CA.
Unexpected Completion	• Received completion without a request for it (Tag, ID, etc.).	Uncorrectable ERR_NONFATAL Log Header	Discard TLP.
Receiver Overflow	• Received TLP beyond allocated credits.	Uncorrectable ERR_FATAL	Receiver behavior is undefined.
Flow Control Protocol Error	• Minimum initial flow control advertisements. • Flow control update for infinite credit advertisement.	Uncorrectable. ERR_FATAL	Receiver behavior is undefined.
Malformed TLP (MP)	• Data payload exceed *Max_Payload_Size.* • Received TLP data size does not match length field. • TD field value does not correspond with the observed size. • PM messages that do not use TC0. • Usage of unsupported VC. • Target request crosses a 4KB boundary.	Uncorrectable ERR_FATAL Log Header	Drop the packet, free FC credits.
Completion with Unsuccessful Completion Status		No Action (already done by originator of completion)	Free FC credits.

3.1.5.3 Completion Timeout Mechanism

The X550 supports completion timeout as defined in the PCIe specification.

The X550 controls the following aspects of completion timeout:

- Disabling or enabling completion timeout.
  - The PCIe *Completion Timeout Disable Supported* bit in the Device Capabilities 2 Register (Section 9.2.3.6.10) is hardwired to 1b to indicate that disabling completion timeout is supported
  - The PCIe *Completion Timeout Disable* bit in Device Control 2 Register controls whether completion timeout is enabled
- A programmable range of timeout values.
  - The X550 supports all four ranges as programmed in the *Completion Timeout Ranges Supported* field of the Device Capabilities 2 Register. The actual completion timeout value is written in the *Completion Timeout Value* field of Device Control 2 Register (Section 9.2.3.6.10)

The following sequence takes place when completion timeout is detected:

- The appropriate message is sent on PCIe as described in Table 3-7.
- The affected queue or client takes action based on the nature of the original request.
- An interrupt is issued to the respective PF.

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 X550 then reacts as listed in Table 3-7.

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

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

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

3.1.5.5 Completion with Unsuccessful Completion Status

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

3.1.5.6 Error Pollution

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

3.1.5.7 Blocking on Upper Address

The PCI_UPADD register blocks 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 (in other words, no TLPs are blocked by this mechanism).

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

Processing a blocked transaction:

- Write transaction
  - The transaction is dropped.
  - Set the “Exceeded upper address limit (write requests)” event in the PCIe errors register (see Section 3.1.5.8).
  - An interrupt is issued as described in Section 3.1.5.8.

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

3.1.5.8 Proprietary Error Reporting

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

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

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

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

- The device checks if the respective bit is set in the PCIe Errors Reported register. If cleared, done. Else, continue
- The respective bit is set in the "PCIe Interrupt Cause" register
- If the respective bit is set in the "PCIe Interrupt Enable" register, an interrupt is issued to the PCI function. The *PCI_EXCEPTION* cause is used (see the EICR register - Section 8.2.2.6.1).

Table 3-8. PCIe Errors Reported to Device Software

Error Event	Index	Description and Comments	Function Association
Exceeded upper address limit (read requests)	00	See Section 3.1.5.7	Sent to PF
Exceeded upper address limit (write requests)	01	See Section 3.1.5.7	Sent to PF
Reserved	02	Reserved entries	N/A
Poisoned TLP received	03	See Section 3.1.5.4	Sent to PF
Reserved	04-05	Reserved entries	N/A
ECRC error detected	06	ECRC check failed on a received TLP. See Section 3.1.5.7	Sent to all PFs
Unsupported Request - Request Type	07	Request causes an Unsupported Request due to receipt of TLP with unsupported Request Type	Sent to PF
Unsupported Request - Vendor Message	08	Request causes an Unsupported Request due to receipt of an Unsupported Vendor Defined Type 0 Message	Sent to PF unless r[2:0] = *Broadcast from Root Complex,* in which case sent to all PFs

Table 3-8. PCIe Errors Reported to Device Software [continued]

Error Event	Index	Description and Comments	Function Association
Unsupported Request - Message Code	09	Request causes an Unsupported Request due to receipt of an invalid Message Code	Sent to PF
Unsupported Request - Function Number	10	Request causes an Unsupported Request due to receipt of a not-supported Function Number	Sent to all PFs
Unsupported Request - Address Range	11	Request causes an Unsupported Request due to receipt of a not-supported Address Range	Sent to PF
Unsupported Request - D3hot	12	Request causes an Unsupported Request due to receipt of a Request TLP during D3hot, other than Configuration and Message requests	Sent to PF
Completer abort - target size	13	Received Target Access with illegal data size per Section 3.1.2.1.1 (CA)	Sent to PF
Reserved	14 - 31	Reserved entries	N/A

3.1.6 Performance and Statistics Counters

The X550 incorporates counters to track the behavior and performance of the PCIe interconnect. The X550 implements several types of counters:

- Transaction layer event counters — Section 3.1.6.1
- Link and Physical layer event counters — Section 3.1.6.2
- Bandwidth counters — Section 3.1.6.3
- Latency counters — Section 3.1.6.4

General characteristics of the counters:

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

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

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

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

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

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

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

3.1.6.1 Event Counters - Transaction Layer

Counters operate in one of the following modes:

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

The list of events supported by the X550 are listed in Table 3-9.

Table 3-9. PCIe Statistic Events Encoding

Events	Event Mapping (Hex)	Description
Cycles	0x00	Increment on each PCIe clock tick
Transaction Layer Events		
Bad Request TLPs	0x10	Number of bad TLPs arriving to the transaction layer.These include: • Request caused UR • Request caused CA • Malformed TLP
Bad Completions	0x11	Number of bad Completions received. These include: • Unexpected Completion • UR status • CA status
Completion Timeout	0x12	Number of completion timeout events
Poisoned TLP	0x13	Number of TLPs received with poisoned data
ECRC Check	0x14	Number of TLPs that foil ECRC check
Link Layer Events		
Retry Buffer Timeout	0x31	Number of replay events that happen due to timeout (does not count replay initiated due to NACK)
Retry Buffer Replay Roll-Over	0x32	Increment when a replay is initiated for more than 3 times
Physical Layer Events		
Receive Error	0x50	Increment when one of the following occurs: 1. Decoder error occurred during training in the PHY. It is reported only when training ends. 2. Decoder error occurred during link-up or till the end of the current packet (in case the link failed). This error is masked when entering/exiting EI.
Surprise Link Down	0x51	Increment when link is unpredictably down (Not because of reset or DFT)

3.1.6.1.1 Count Mode

The following CSR fields control operation of the Count mode:

- Four 32-bit counters PCI_GSCN_0_3 track events and increment on each occurrence of an event.
  - The four 32-bit counters can also operate in a two 64-bit mode to count long intervals or large payloads.
    - Registers PCI_GSCN_0_3[0] and PCI_GSCN_0_3[1] form the first 64-bit counter. Registers PCI_GSCN_0_3[2] and PCI_GSCN_0_3[3] form the second 64-bit counter.
    - The PCI_GSCL_1.GIO_64_BIT_EN selects between 32-bit and 64-bit modes.
- The PCI_GSCL_1.GIO_COUNT_EN_[3:0] bits enable each of the 4 counters.
  - The enable bits for the two 64-bit counters are PCI_GSCL_1.*GIO_COUNT_ EN_0* and PCI_GSCL_1.*GIO_COUNT_ EN_2*, respectively.
- The PCI_GSCL_1.*GIO_COUNT_START* bit starts event counting of enabled counters.
- The PCI_GSCL_1.*GIO_COUNT_STOP* bit stops event counting of running counters.
- The PCI_GSCL_1.*GIO_COUNT_RESET* bit resets the event counters.
- The PCI_GSCL_2 associates an event with each of the 4 counters.
  - In 64-bit mode, the *GIO_EVENT_NUM_[2,0]* fields are used.

3.1.6.1.2 Leaky Bucket Mode

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

- One of four 16-bit Leaky Bucket Counters (LBC) is enabled via the *LBC_ENABLE_[3:0]* bits in the PCIe Statistic Control register #1.
- The LBC is controlled by the *GIO_COUNT_START, GIO_COUNT_STOP, GIO_COUNT_RESET* bits in the PCIe Statistic Control register #1.
- The LBC increments every time the respective event occurs.
- The LBC is decremented every T μs as defined in the *LBC_TIMER_N* field in the PCIe Statistic Control registers #5...#8 (*PCI_GSCL_5_8*).
- When an event occurs and the value of the LBC meets or exceeds the threshold defined in the *LBC_THRESHOLD_N* field in the PCIe Statistic Control registers #5...#8 (*PCI_GSCL_5_8*), the respective statistics counter increments, and the LBC counter is cleared to zero.

3.1.6.2 Event Counters - Link and Physical Layers

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

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

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

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

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

- Resetting the counters configuration:
  - Set the XPPERFCON.*GRST* bit.
  - Clear the XPPERFCON.*GRST* bit (otherwise the logic stays in reset).
- Setting an event:
  - Write 0x0...0 to the XPPMCL[1:0] registers
  - Set the XPPMR[1:0].*CENS* field to 0x1
  - Set the XPPMR[1:0].*CNTMD* field to 0x1
  - Set the XPPMER[1:0].*XPRSCA* field to 0x1
  - Set the event according to Table 3-10.
- Starting a count:
  - Set the XPPERFCON.*GCE* bit.
- Stopping a count:
  - Clear the XPPERFCON.*GCE* bit.
- Reading the count (note: reading the counter clears their values):
  - Read the respective XPPMDH[1:0] and XPPMDL[1:0] register pair in a single 64-bit aligned access.

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

Table 3-10. Link and Physical Layer Performance Events

Event	Description	Register Field
Uncorrectable Errors	Counts the total number of Uncorrectable Errors.	XPPMER[1:0].*CNTUCERR*
Correctable Errors	Counts the total number of Correctable Errors.	XPPMER[1:0].*CNTCERR*
Tx L0s state utilization	Counts the number of entries to L0s on the Tx lanes.	XPPMER[1:0].*TXL0SU*
Rx L0s state utilization	Counts the number of entries to L0s on the Rx lanes.	XPPMER[1:0].*RXL0SU*
Link Utilization	Counts clocks that a port is receiving data. If one counter counts receiver errors and another counter counts Link Utilization, a bit error rate can be calculated.	XPPMER[1:0].*LNKUTIL*
Recovery State Utilization	Counts the number of entries to Recovery state.	XPPMER[1:0].*RECOVERY*
ASPM L1 state utilization	Counts the number of entries to ASPM L1 state (i.e. initiated by the device).	XPPMER[1:0].*L1*
SW L1 state utilization	Counts the number of entries to L1 state initiated by software.	XPPMER[1:0].*SWL1*
Tx and Rx L0s utilization	Counts number of events where both Tx and Rx are in L0s state.	XPPMER[1:0].*RXL0STXL0SU*
NAK DLLP received	Counts number of received NAK DLLPs.	XPPMER[1:0].*NAKDLLP*

3.1.6.3 Bandwidth Counters

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

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

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

Registers fields used exclusively by the bandwidth counters:

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

3.1.6.3.1 Register Map

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

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

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

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

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

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

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

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

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

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

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

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

Field	Bit(s)	Init.	Description
TXRXSEL	1:0	00b	**Tx/Rx Select** This field selects the traffic direction to monitor: 1xb = Transmit x1b = Receive (from PCIe bus) 11b = Either Transmit or Receive direction.
FCCSEL	4:2	000b	**Flow Control Class Select** This field selects which flow class for resource event xx1b = Posted x1xb = Non-Posted 1xxb = Completion ***Note:*** Setting to 111b counts posted, non-posted, and completion traffic combined.
RSVD	12:5	0x0	Reserved.
LNKUTIL	16:13	0x0	**Link Utilization** Set to 0x1 to enable.
XPRSCA	20:17	0x0	**XP Resource Assignment** 0001b = Set All other values are reserved.
RXL0SU	21	0b	**Rx L0s State Utilization Event** 0b = Disabled 1b = Enabled
TXL0SU	22	0b	**Tx L0s State Utilization Event** 0b = Disabled 1b = Enabled
CNTCERR	23	0b	**Count Correctable Errors** 0b = Disabled 1b = Enabled
CNTUCERR	24	0b	**Count Uncorrectable Errors** 0b = Disabled 1b = Enabled
RECOVERY	25	0b	**Recovery State Utilization** 0b = Disabled 1b = Enabled
L1	26	0b	**ASPM L1 State Utilization** 0b = Disabled 1b = Enabled
SWL1	27	0b	**SW L1 State Utilization** 0b = Disabled 1b = Enabled

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

Field	Bit(s)	Init.	Description
RXL0STXL0SU	28	0b	**Tx and Rx L0s Utilization** 0b = Disabled 1b = Enabled
NAKDLLP	29	0b	**NAK DLLP Received** 0b = Disabled 1b = Enabled
RSVD	31:30	0x0	Reserved.

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

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

3.1.6.4 Latency Counter

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

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

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

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

The registers are accessed via the PCI_LCBADD and PCI_LCBDATA registers.

The register fields that control the latency counter operation are:

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

Field	Bit(s)	Init.	Description
Reserved	3:0	0x4	Reserved.
PERFMNTRAVG	7:4	0x1	**Performance Monitor Average Rate** This field sets the averaging rate for all latency average monitors. See definition of NPQRTDLY1.*ARTDLY* (Table 3-18). This field divided by 16 is the weight W in an exponential moving average. The possible values are 1, 2, 4 or 8, which correspond to averaging rates of 0.0625, 0.125, 0.25 or 0.5, respectively.
PERFMNTREN	8	0b	**Performance Monitor Enable** This bit should set to enable latency counters. Clearing this bit clears the latency counters.

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

Field	Bit(s)	Init.	Description
RTMNTREN	9	0b	**Latency Counting Enable** This bit should set to enable latency counters. When set, completion timeout events are ignored.
Reserved	31:10	0x0	Reserved.

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

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

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

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

Latencies are measured in cycle counts, where a cycle duration is per Table 3-20.

Table 3-20. Resolution of the Latency Counters

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

3.2 Management Interfaces

The X550 contains three possible interfaces to an external BMC.

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

3.2.1 SMBus

SMBus is an optional interface for pass-through and/or configuration traffic between an external BMC and the X550. The SMBus channel behavior and the commands used to configure or read status from the X550 are described in Section 11.5.

The X550 also enables reporting and controlling the device using the MCTP protocol over SMBus. The MCTP interface is used by the BMC to control the NIC and for pass-through traffic. All network ports are mapped to a single MCTP endpoint on SMBus. For information, refer to Section 11.5.

3.2.1.1 Channel Behavior

The SMBus specification defines a maximum frequency of 100 KHz. However, when acting as a slave, the X550 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 (Section 6.2.17.3).

3.3 Network Controller — Sideband Interface (NC-SI)

The NC-SI interface in the X550 is a connection to an external MC. It operates as a single interface with an external BMC, where all traffic between the X550 and the BMC flows through the interface.

The X550 NC-SI interface meets the NC-SI version 1.0.0 specification as a PHY-side device.

3.3.1 Electrical Characteristics

The X550 complies with the electrical characteristics defined in the NC-SI specification.

3.3.2 NC-SI Transactions

The NC-SI link supports both pass-through traffic between the BMC and the X550 LAN functions, as well as configuration traffic between the BMC and the X550 internal units as defined in the NC-SI protocol. Refer to Section 11.6.2 for information.

3.3.3 MCTP (Over PCIe or SMBus)

The X550 supports MCTP protocol for management. MCTP runs over PCIe or SMBus. The X550 implements NC-SI over MCTP protocol for command and pass-through traffic. See Section 11.7 for details.

3.4 Non-Volatile Memory (NVM)

3.4.1 General Overview

The X550 uses a Flash device to store product configuration information. The Flash is divided into three general regions:

- **Hardware Accessed** — Loaded by the the X550 hardware after power-up, PCI reset de-assertion, D3 to D0 transition, or software reset. Different hardware sections in the Flash are loaded at different events. For more details on power-up and reset sequences, see Section 4.1 and Section 4.2.
- **Firmware Area** — Includes firmware code and structures used by the firmware for management configuration in its different modes.
- **Software Accessed** — This region is used by software entities such as LAN drivers, option ROM software and tools, PCIe bus drivers, VPD software, etc.

3.4.1.1 NVM Protection

The NVM protection method implemented in the X550 relies on an "authenticate on update" concept. It means that the protected modules are not authenticated after initialization, but prior to committing a module update operation only. NVM protection is guaranteed by an inductive authentication chain, that assumes an initial secured NVM image and requires that any NVM update must be secure as well. This method mandates the following limitations and restricting working assumptions:

1. An initial 'good' image is loaded into the flash at the manufacturing site which is assumed to be safe.
   - 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 that the NIC/LOM is physically protected from modification after deployment.
   - The possibility exists that unauthorized firmware may be loaded into the NVM via physical modification post manufacturing, as well as through 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 a normal operating mode, NVM write accesses are controlled by the device (firmware) and cannot be performed via the memory mapped accesses, EEWR register, or bit-banging. Memory mapped NVM access remains available for NVM read accesses only. For simplicity and flexibility reasons, NVM write accesses from the host can be initiated via host interface commands (Section 11.8.3.3), VPD write interface, or via a BMC command, which are all handled by firmware. Memory BAR writes, EEWR and FLA bit-banging accesses are blocked by hardware when the NVM is protected.

3. All the supported Flash parts share the same set of op-codes as described in Section 3.4.1.2. 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 firmware involvement via the FLA bit-banging or Flash BAR interfaces. This mode is indicated by FLA.*LOCKED* = 0b.

3.4.1.2 Flash Device Requirements

The X550 supports Flash devices with a sector erase size of 4 KB and an address width of 24 bits (up to 4 MB). Note that many Flash vendors are using the term sector differently. This document uses the term Flash sector for a logic section of 4 KB.

The X550 supports Flash devices that are either write-protected by default after power-up or not. The X550 is responsible to remove the protection by sending the write-protection removal Op-Code to the Flash after power up.

The following Op-Codes are supported by the X550 as they are common to all supported Flash devices:

1. Write Enable (0x06)
2. Read Status Register (0x05) - used by hardware internally.
3. Write Status Register (0x01). The written data is 0x00 to cancel the Flash default protection. Used by hardware internally.
4. Read Data (0x03). Burst read is supported.
5. Byte Program (0x02). To program a data byte.
6. 4 KB Sector-Erase (0x20)
7. Chip-Erase (0xC7)
8. Read JEDEC ID (0x9F)

3.4.1.2.1 Flash Identification

The Flash connected to the X550 can be identified by its JEDEC ID that can be read using the *Flash Info* host interface command (Section 11.8.3.3.9). This identification is available only if a valid Flash is installed. If the Flash is empty or with an invalid signature, software can read the JEDEC ID by applying an RDID command (opcode 0x9F) or a Read SFDP command (opcode 0x5A) via the bit-bang interface.

3.4.2 Shadow RAM

The first eight 4 KB sectors of the X550's Flash are allocated to create two 16 KB sections (section 0 and section 1), for the configuration content. At least one of these two sections must be valid at any given time or else the X550 is set to hardware default. Following a Power On Reset (POR), the X550 copies the valid section of the Flash device into an internal Shadow RAM. Any further read accesses of the software or firmware to the lower 16 KB addresses of the NVM (not through flash BAR) are directed to the internal Shadow RAM. Modifications made to the Shadow RAM content are copied by the X550 into the other 16 KB section of the NVM, circularly flipping the valid section between sections 0 and 1 in the NVM.

This mechanism provides the following advantages:

1. A way to protect the image-update procedure from power down events by establishing a double-image policy. See Section 3.4.8.1 for a description of the double-image policy. It relies on having pointers to NVM modules stored in the NVM section mirrored in the internal Shadow RAM.
2. A way to ensure hardware auto-load during a PCIe reset event can be completed within PCIe timing constraints (100 ms), even if the Flash is busy performing an erase operation initiated prior to that reset event.

Figure 3-5 shows the Shadow RAM mapping and interface.

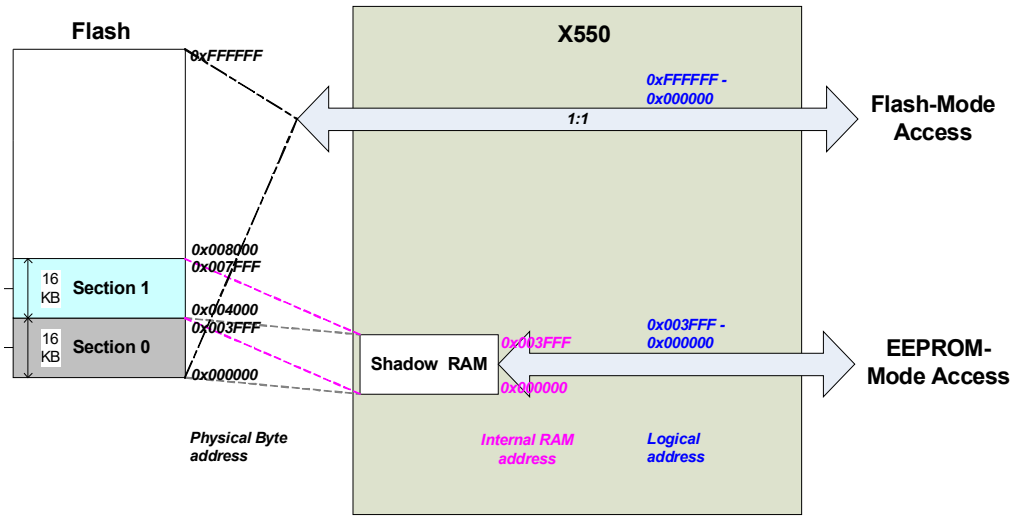


Figure 3-5. NVM Shadow RAM

3.4.2.1 Shadow RAM Update Flow

Following a write access by the software device driver to update the Shadow RAM, the data should be updated in the Flash as well. The X550 updates the Flash from the Shadow RAM when software explicitly requests to update the Flash using the Shadow RAM Dump host interface command (Section 11.8.3.3.8). To reduce Flash update operations, software is expected to request a dump only once its last Shadow RAM update access completes. The X550 then copies the content of the Shadow RAM to the non valid configuration section and makes it the valid one.

3.4.3 NVM Clients and Interfaces

There are different software clients that can access the NVM: driver, tools, BIOS, VPD, etc.

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

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

Client	NVM Access Method (Data Width)	NVM Access Target	Logical Byte Address Range	NVM Access Interface	Protection and Enforcement
Host Software	Memory BAR (parallel 32-bit)	Flash	0x000000 - 0xFFFFFF	Memory mapped read/write via BARs.	Write access is limited to a single byte, and is allowed only if hardware protection is disabled (FLA.*LOCKED* = 0).
	FLA bit-banging (serial 1-bit)	Flash	0x000000 - 0xFFFFFF	Software accesses to Flash by toggling the SPI pins.	FLA interface access is available only if hardware protection is disabled (FLA.*LOCKED* = 0).
	Host interface Shadow RAM Read/Write command	Shadow RAM	0x000000 - 0x003FFF	Access to Shadow RAM through the Shadow RAM Read/Write command	Requires a valid firmware image. Write protection is enforced by firmware.
	Host interface Flash read command	Flash	0x000000 - 0xFFFFFF	Software read from Flash via Flash Read command.	Requires a valid firmware image.
	Host interface Flash write command/Flash block erase command	Flash	0x008000 - 0xFFFFFF	Software write to sector/Flash erase	Requires a valid firmware image. Writes to protected areas are dropped when protection (enforced by firmware) is enabled. Protected sector erase or a complete Flash erase requests are rejected when protection (enforced by firmware) is enabled.
	VPD access (parallel 32-bit)	Shadow RAM	0x000000 - 0x0003FF	VPD Address and Data registers.	Write accesses are enabled to the R/W area of the VPD If the VPD structure is not valid, the entire 1024 bytes area is RO.
Firmware or Software	FLMNG Parallel (Read: 32-bits Write: 8-bits)	Flash	0x000000 - 0xFFFFFF	FLMNGCTL and FLMNGDATA registers accesses to the FLASH	Software access is available only when protection is disabled.

3.4.3.1 Memory Mapped Host Interface

Using the legacy Flash transactions, the Flash is accessed by the X550 each time the host CPU performs a read or a write operation to a memory location mapped to the Flash address space, 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 Expansion ROM Base Address Register (PCIe Control Register at offset 0x30).
- The X550 is responsible to map accesses via the Expansion ROM BAR to the physical NVM. The offset in the NVM of the Expansion ROM module is defined by the *PCIe Expansion/Option ROM Pointer* (NVM word address 0x05). This pointer is loaded by the X550 from the NVM before enabling any access to the Expansion ROM memory space.

The X550 controls accesses to the Flash when it decodes a valid access. Attempt to out of range read access the PCIe Expansion/Option ROM module (according to *NVM Size* field in *NVM Control Word 1*) would return a value of 0xDEADBEEF. Attempt to memory-mapped write accesses to the Flash when protection is enabled or via expansion ROM BAR are ignored.

3.4.4 Flash Access Contention

Flash accesses initiated through different LAN functions might occur concurrently. The X550 does not synchronize between entities accessing the Flash, so a contention caused from one entity reading and another modifying the same location is possible.

To avoid such contention between software LANs or between software and firmware accesses, these entities are required to make use of the semaphore registers. Refer to Section 11.8.4. Any read or write access to the NVM made by software/firmware must be preceded by acquiring ownership over the NVM. This is also useful to avoid the time out of a PCIe transaction made to a memory mapped Flash address while the Flash is busy performing a sector erase operation.

However, two software entities cannot use this semaphore mechanism: BIOS access through expansion ROM and VPD software.

- Since VPD software accesses only the VPD module, which is located in the configuration section of the NVM, VPD accesses are always performed against the Shadow RAM. Firmware must take NVM ownership before dumping the VPD changes to the Flash. The Shadow RAM dump sequence is described in Section 3.4.2.1.
- No contention can occur between the BIOS access through expansion ROM and other software entities (including VPD) as it accesses the NVM while the operating system is down.
- Contentions between BIOS and firmware can however happen if a system reboot occurs while the MC is accessing the NVM.
  - If a system reboot is caused by a user pressing the standby button, it is required to route the wake-up signal from the standby button to the MC and not to the chipset. The MC issues a system reboot signal to the chipset only after the NVM write access completes. Firmware is responsible to respond with a "busy" error code to MC NC-SI commands while other NVM writes are in progress.
  - If a system reboot is issued by a local user on the host, there is no technical way to prevent NVM access contentions between the BIOS and the MC.

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

Notes: The PHY auto-load process from the Flash device is made up of short read bursts (32-bits) that can be inserted by hardware in between other NVM clients' accesses, at the lowest priority. It is the user's responsibility to avoid initiating a PHY auto-load while updating the PHY NVM modules.

The MAC auto-load from the Flash device itself occurs after power-up only, before host or firmware can attempt to access the Flash. The host must wait until PCIe reset is de-asserted (after ~1 second, which is enough time for the MAC auto-load to complete), and firmware starts its auto-load after the EEC.*MNG_READY* bit is asserted by hardware.

Other MAC auto-load events are performed from the internal Shadow RAM and do not compete with memory mapped accesses to the Flash device. During such MAC auto-loads, accesses from other clients via EEPROM-Mode registers are delayed until the auto-load process completes.

Software and firmware should avoid holding Flash ownership (via the dedicated semaphore bit) for more than 500 ms.

3.4.4.1 Flash Deadlock Avoidance

The Flash is a shared resource between the following clients:

1. Hardware auto-load of Shadow RAM (at power up).
2. LAN port 0 and LAN port 1 software accesses.
3. Manageability/firmware accesses.
4. Software tools.

All clients can access the Flash using parallel access. Hardware implements the actual access to the Flash. Hardware arbitrates between the different clients and schedules these accesses, avoiding starvation of any client.

However, the software and firmware clients can access the Flash using bit-banging. In this case, there is a request/grant mechanism that locks the Flash to the exclusive use of one client. If one client is stuck without releasing the lock, the other clients can no longer access the Flash. To avoid this deadlock, the X550 implements a time-out mechanism, which revokes the grant from a client that does not toggle the Flash bit-bang interface (FLA.*FL_SCK* bit) for more than 2 seconds. If any client fails to release the Flash interface, hardware clears its grant, enabling the other clients to use the interface.

The deadlock timeout mechanism is enabled by the *Deadlock Timeout Enable* bit in *NVM Control Word 2* in the Flash.

3.4.5 Signature Field

The only way the X550 can tell if a Flash is present is by trying to read the Flash. The X550 first reads the Control word at word address 0x0 and at word address 0x2000. It then checks the signature value at bits 7 and 6 in both addresses.

If bit 7 is 0b and bit 6 is 1b in (at least) one of the two addresses, it considers the Flash to be present and valid. It then reads the additional Flash words from that section and programs its internal registers based on the values read. Otherwise, it ignores the values read from that location and does not read additional words.

If the signature bits are valid at both addresses the X550 assumes that the first section is the valid one.

3.4.6 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 0xFFFF means VPD is not supported and the PCI_CAPCTRL.*VPD_EN* bit should be cleared in the PCI NVM section (Section 6.2.6.9), to prevent the VPD capability from appearing in the configuration space.

The maximal area size is 1024 bytes but 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 Table 3-22 and Table 3-23.

Table 3-22. Small Resource Structure

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

Table 3-23. Large Resource Structure

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

The X550 parses the VPD structure during the auto-load process following PCIe reset to detect the read only and read/write area boundaries. The X550 assumes the following VPD fields with the limitations listed:

Table 3-24. 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 should start with a Tag = 0x82. If the X550 does not detect a value of 0x82 in the first byte of the VPD area or the structure does not follow the description of Table 3-24, it assumes the area is not programmed and the entire 1024 bytes area is read only.
- The RO area and RW area are both optional and can appear in any order. A single area is supported 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 structure should end with a Tag = 0x78. The tag must be word aligned.
- The VPD area can be accessed through the PCIe configuration space VPD capability structure listed in Table 3-24. Write accesses to a read only area or any access to an offset outside of the VPD area via this structure are ignored.
- VPD area must be mapped in the first 16 KB section of the Flash mapped to the Shadow RAM.
- VPD software does not check the semaphores before attempting to access the Flash via dedicated VPD registers. Even if the Flash is owned by another entity, VPD software read access to the VPD area in the Flash might complete immediately since it is first performed against the Shadow RAM. However, VPD software write access might not complete immediately since the VPD changes are committed to the Flash device at the X550's initiative, once the other entity releases Flash ownership, which may take up to several seconds.

3.4.6.1 VPD Access Flows

3.4.6.1.1 First VPD Area Programming

The VPD capability is exposed in the PCIe configuration space only if the *PCI_CAPCTRL.VPD_EN* bit is set, regardless of any other sanity checks that are performed on the VPD area contents.

The VPD content and pointer can be written on a blank Flash without any limitation, such as for any other NVM module when in the blank Flash programming mode. After protection is enabled, if *VPD Write Enable* bit in *NVM Control Word 1* is cleared, only the RW area of the VPD is writable and only via the VPD interface.

3.4.6.1.2 VPD Area Update Flow

1. The host initiates a VPD write by programming the offset and data fields of the VPD capability register set, and then setting the capability's *Flag* bit.(bit 15 in VPD Address Register - 0xE2).
2. Firmware checks if the VPD write is allowed - it checks that the write offset falls within the VPD-RW area.
   a. If writing is not allowed, firmware clears the VPD flag in the configuration space to notify the VPD software that the transaction completed, and then it exits the flow.
3. Firmware indicates VPD access completion by clearing the VPD flag in the configuration space.

Note: In case the Flash is occupied with a previous sector erase operation, or if NVM ownership is held by software, the completion indication (Step 3) might be delayed. Additional writes should not be attempted before the completion of Step 3.

3.4.7 NVM Read, Write, and Erase Sequences

Refer to Section 3.4.8.1 for the flow required to update secure NVM modules.

Any software flow described in this section must be preceded by taking NVM ownership via semaphores as described in Section 11.8.4.

3.4.7.1 Flash Block Erase Flow by Software

1. Send a Flash Block Erase host interface command (Section 11.8.3.3.7) with the aligned address of the block to erase.
2. Wait for the command to complete before releasing the NVM semaphore.

3.4.7.2 X550 Software Flow to the Bit-Banging Interface

To directly access the Flash when Flash is blank or not protected, software should follow these steps:

1. Write a 1b to the *Flash Request* bit (FLA.*FL_REQ*).
2. Poll the *Flash Grant* bit (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 FLA register. The exact protocol used depends on the Flash placed on the board.

4. Write a 0b to the *Flash Request* bit (FLA.*FL_REQ*).
5. Following a write or erase instruction, software 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 Flash data sheet for the op-code to be used for reading the STATUS register.

Note: The bit-banging interface is blocked during normal operation (protection enabled). Software should use the Host Interface commands. The firmware can use this interface all the time.

3.4.7.3 Erase Flow Using the FLA Register

To directly erase a sector in the Flash when Flash is blank or not protected, software should follow these steps:

1. Take ownership of NVM semaphore.
2. Set the *Flash Address* (FLA.*FL_ADDR*) to the index of the 4 KB sector to erase and *Sector Erase* bit (FLA.*FL_SER*) bit.
3. Read the FLA register until *Flash Busy* bit (FLA.*FL_BUSY*) is cleared.
4. Release ownership of NVM semaphore.

To directly erase the entire Flash when Flash is blank or not protected, software should follow these steps:

1. Take ownership of NVM semaphore
2. Set *Device Erase* bit (FLA.*FL_DER*) bit.
3. Read the FLA register until *Flash Busy* bit (FLA.*FL_BUSY*) is cleared.
4. Release ownership of NVM semaphore.

3.4.7.4 Software Access Flow to Shadow RAM

3.4.7.4.1 Read Interface

Software can read from the Shadow RAM using the following flow:

1. Send a Shadow RAM Read host interface command (Section 11.8.3.3.2) with the address and length to read.
2. Wait for the command to complete.
3. Read the data from the response buffer.

3.4.7.4.2 Write Interface

Software can write to the Shadow RAM using the following flow:

1. Send a Shadow RAM Write host interface command (Section 11.8.3.3.4) with the address, length and data to write.
2. Wait for the command to complete.

3.4.7.5 Software Access to Flash via Memory Mapped Interface

3.4.7.5.1 Read Access

Software can always use the Flash BAR for read accesses.

Note: Software should take semaphore ownership before executing the flow.

3.4.7.5.2 Write Access

When Flash is blank or protection is disabled, software might initiate a write cycle via the Flash BAR as follows:

1. Take semaphore ownership before executing the flow.
2. Write the data byte to the Flash through the Flash BAR.
3. Poll the *FL_BUSY* flag in the FLA register until cleared.
4. Repeat Step 2 and Step 3 to write additional bytes.
5. Release NVM semaphore ownership

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

1. Set the *FL_BUSY* bit in the 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 FLA register.

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

3.4.7.6 Software Flash Programming via Host Interface Command

Software must take semaphore ownership before executing the flow.

Software can write to non write protected areas of the flash using the following flow:

1. Send a *Flash Write* host Interface command (Section 11.8.3.3.3) with the address, length and data to write.
2. Wait for the command to complete.

3.4.7.7 Software Flash Read via Host Interface Command

Software must take semaphore ownership before executing the flow.

Software can read from the flash using the following flow:

1. Send a *Flash Read* host Interface command (Section 11.8.3.3.1) with the address and length of the data to read.
2. Wait for the command to complete.
3. Read the data from the completion of the command.

3.4.8 Extended NVM Flows

3.4.8.1 Flow for Updating Secured Modules

This section describes the flow to use to update the firmware image (Section 6.1.4), Option ROM (Section 6.1.5) or PHY module (Section 6.1.6).

To protect the Flash update procedure from power-down events, a double image policy is used for each of the updated modules. The software flow to update a module is as follows:

1. Take ownership over the NVM via the semaphore bits. Refer to Section 11.8.4.
   a. If SW_FW_SYNC.*NVM_UPDATE_STARTED* bit is read as clear, set this bit together with setting NVM semaphore bit. It is used to notify other entities that a long NVM update process which may take up to several minutes has started. During this time, other entities can not perform a write access to the firmware or PHY modules, but reading these modules in between update write bursts is allowed using the flash memory mapping. Legacy EEPROM Modules are not concerned by this limitation.
   b. Otherwise, release NVM semaphore ownership and restart the update process later on.
2. Read the pointer to the free provisioning area (NVM word 0x40). Check that the free provisioning area size read from NVM word 0x41 is greater or equal to the size of the new firmware/PXE/PHY image to be loaded in NVM.
   a. If not, release NVM semaphore ownership, clear the SW_FW_SYNC.*NVM_UPDATE_STARTED* bit and exit the flow.
3. Initiate sector erase instructions (Section 3.4.7.1) to the entire free space provisioning segment.
   a. To guarantee NVM semaphore ownership time does not exceed the 1 sec timeout, it is recommended to perform at this step no more than four 4 KB sector erase operations at once in a burst, releasing semaphore ownership for 200 µs in between. This way, other entities can insert NVM read accesses in between burst without waiting for the entire update process completion, which might take minutes.
4. Write the new firmware/Option ROM/PHY module to the free provisioning area via Flash-mode access.
   a. Same as Step 3a, it is recommended to write at this step no more than four 4 KB sectors at once in a burst, releasing semaphore ownership for 200 µs in between.

5. Send a Flash Module Update host Interface command (Section 11.8.3.3.5) with the module ID of the section to update.The encoding of the modules is:

Module	ID	Section Number
Firmware code	0x1	6.1.4
Reserved	0x2	
Reserved	0x3	
Reserved	0x4	
PHY Firmware Image	0x5	6.1.6
Reserved	0x6-0xFD	
Option ROM	0xFE	6.1.5
Reserved (Full Shadow RAM)	0xFF	

6. Release the NVM semaphore and clear the SW_FW_SYNC.*NVM_UPDATE_STARTED* bit.
   a. Software must avoid taking the NVM semaphore again until the firmware command has completed. Any attempt to write the NVM until then is not performed by the device.
7. Firmware swaps between the Free Provisioning Area Pointer (word 0x40) and the Firmware Code Module pointer, PXE module pointer or the PHY pointer located at the Shadow RAM word address 0x3A/0x5/0x4 respectively
8. Software waits for the command to complete.
   a. If the update process failed due to a security check failure or a flash write fault, an *Authentication Error* (0x80) or Data Error (0x6) respectively is returned. Software must then exit the flow, prior to attempt another update.
9. If the updated module is the PHY image, the PHY should be instructed to reload the image using the following flow:
   a. Update the PHY firmware version number at NVM word address 0x19 (Section 6.2.2.26) using the Legacy EEPROM Module Update flow (Section 3.4.7.6).
   b. Read-modify-write SRAMREL register for setting *LATCH_IMAGE_VLD* bit to 1b.
   c. Read-modify-write SRAMREL register for setting *LATCH_IMAGE_VLD* bit to 0b.
   d. Write PHY register bit 1E.C474.0 bit to 0b.
   e. Force PHY image reload by setting PHY register bit 1E.C442.0 to 1b.
10. If the updated module is the firmware image, the firmware should be instructed to reload the image using the *Apply Update* command (Section 11.8.3.3.6).

3.4.8.2 Flow for Updating One of the RW Legacy EEPROM Modules

When updating one or several fields from a legacy EEPROM module there is a risk that a hardware auto-load event occurs in the middle of the operation (fox example, due to a sudden PCIe reset), leading to the auto-load of an invalid or inconsistent content from the internal Shadow RAM into the device registers or memory. Therefore unless the field(s) can be updated by a single EEPROM-mode access, the updating software must repeatedly use the following procedure for each legacy EEPROM module to be updated:

1. Take ownership over the NVM via semaphore bits. Refer to Section 11.8.4.
2. Invalidate the pointer to the module to be modified by setting it to 0xFFFF using *Shadow RAM Write* command. This way, if a hardware auto-load of the module is attempted, the associated register defaults are loaded instead. Do not invalidate pointers to firmware modules, only to hardware auto-load modules.
3. Modify the contents of the module via *Shadow RAM Write* command.
4. Restore the pointers modified in Step 2 via *Shadow RAM Write* command.
5. Compute and update the software checksum (word 0x3F) if the contents covered by the software checksum was modified.
6. Release the NVM semaphore.
7. Send a Shadow RAM dump command (Section 11.8.3.3.8) to ask the device firmware to load the internal Shadow RAM into the Flash.

Note: Depending on the modified RO items, a system reset is generally required for loading the modifications into the device.

3.4.9 NVM Authentication Procedure

NVM update integrity feature ensures that only Intel approved firmware code (or other protected NVM module) is able to be updated on the X550 devices after manufacturing. This procedure is performed whenever attempting to update one of the protected modules.

Integrity validation of NVM updates is provided by a digital signature. The digital signature is a SHA256 Hash computed over the protected content (long by 256-bits) which is then encrypted by a 2048-bits RSA encryption using an Intel private key. This digital signature is stored in 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.

To verify the authenticity of the digital signature, firmware must first verify that the RSA Modulus and RSA Exponent fields in the new firmware image loaded are identical to those in the old firmware image. If the RSA Modulus and Exponent fields are the same, firmware decrypts the digital signature using the 2048-bit RSA Modulus and Exponent fields stored in the manifest of the old firmware image to extract the expected SHA256 Hash of content (stored hash). Firmware 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 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 device flash.

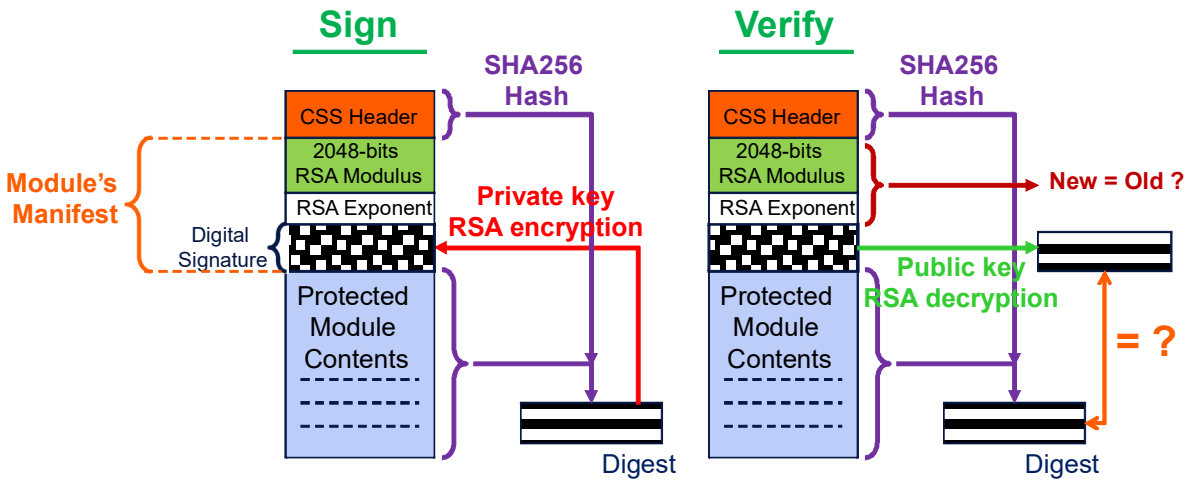


Figure 3-6. Sign & Verify Procedures for Authenticated NVM Modules

3.4.9.1 Digital Signature Algorithm Details

As described 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 X550.

The different 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

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

The X550 has four software-defined pins (SDP pins) per port that can be used for miscellaneous hardware or software-controllable purposes. Unless specified otherwise, these pins and their function are bound to a specific LAN device. The use, direction, and values of SDP pins are controlled and accessed by the Extended SDP Control (ESDP) register. To avoid signal contention, following power-up, all four pins are defined as input pins.

Some SDP pins have specific functionality:

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

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

The bit mappings are shown in Table 3-25 for clarity.

Table 3-25. GPI to SDP Bit Mappings

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

- SDP1 pins can also be used to (electrically) disable both PCIe functions altogether. Also, if the MC is present, the MC-to-LAN path(s) remain fully functional. This PCIe-Function-Off mode is entered when SDP1 pin of a port is driven high while PE_RST_N is de-asserted. For correct capturing, it is therefore recommended to set SDP1 pins to their desired levels while the PE_RST_N pin is driven low and to maintain the setting on the (last) rising edge of PE_RST_N. This ability is enabled by setting bit 11 in *NVM Control Word 2* (global NVM offset 0x01 - Section 6.2.2.1).

  Note: The PCIe-Function-Off is activated regardless of the SDP1 direction defined in the NVM SDP Control word.

- The lowest SDP pins (SDP0_0) of port 0 can be used to encode the NC-SI package ID of the X550. This ability is enabled by setting bit 15 in *NC-SI Configuration 2* word (offset 0x05 - Section 6.2.17.6) of the NVM. The 3-bit package ID is encoded as follows: Package ID = {0, SDP0_0, 0}.
- When the SDP pins are used as IEEE1588 auxiliary signals they can generate an interrupt on any transition (rising or falling edge), refer to Section 7.7.4.
- A pair of SDPs can be used to create an I^2C interface as described in Section 3.5.1.

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

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

Table 3-26 summarize the setup required to achieve each of the possible SDP configurations.

Table 3-26. SDP Settings

SDP	Usage	NVM Setting	ESDP			
			SDPx_NATIVE	SDPx_IODIR	SDP1_Function	SDP23_function
0	SDP	N/A	0	Input/Output	N/A	
	NC-SI package ID (Port #0 only)	Bit 15 in *NC-SI Configuration 2* NVM word	0	Input		
	1588 functionality as defined by the TSSDP register	N/A	1	Input/Output		
1	SDP	N/A	0	Input/Output	0	N/A
	PCI disable	NVM Control Word 2, SDP_FUNC_OFF_EN bit	N/A	N/A	N/A	
	1588 functionality as defined by the TSSDP register	N/A	1	Input/Output	0	
	Thermal Sensor	TS NVM-based Mode Enable bit in the Common Firmware Parameters word	0	Output	1	
	Reserved	N/A	1	N/A	1	
2	SDP	N/A	0	Input/Output	N/A	N/A
	1588 functionality as defined by the TSSDP register		1	Input/Output		0
	I^2C clock		1	N/A		1
3	SDP	N/A	0	Input/Output	N/A	N/A
	1588 functionality as defined by the TSSDP register		1	Input/Output		0
	I^2C data		1	N/A		1

3.5.1 I²C Over SDP

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

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

The I²C interface can be used in two methods, a hardware based access, where the device initiate a transaction following a software device driver request via the I2CCMD register (Section 8.2.2.1.16) or a software controlled bit-banging using the I2CPARAMS register (Section 8.2.2.1.17).

3.5.1.1 Hardware Based I²C Access

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

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

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

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

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

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

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

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

See Section 3.5.1.3 below for the I²C commands supported when using the built in read and write commands. All the transactions uses a clock of 100 KHz. When using the bit-bang method any command can be given to the I²C device.

3.5.1.2 Bit-Bang Based I²C Access

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

Pad	Field Controlling the Output Value	Field Reflecting the Input Value	Field Controlling the Output Enable Value[1]
SDPx_2 (I²C clock)	CLK_OUT	CLK_IN	CLK_OE_N
SDPx_3 (I²C data)	DATA_OUT	DATA_IN	DATA_OE_N

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

3.5.1.3 Supported Commands

Note: The gray columns below denotes cycles driven by the I²C device. White columns denotes cycles driven by the X550.

When a word read command (I2CPARAMS.*ACCESS_WIDTH* = 1b, I2CCMD.*OP* =1b) is given the following sequence is done by the X550:

Table 3-27. I²C Read Transaction - Dummy Write

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

Table 3-28. I²C Read Transaction - Word Read

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

When a byte read command (I2CPARAMS.*ACCESS_WIDTH* = 0b, I2CCMD.*OP* =1b) is given the following sequence is done by the X550:

Table 3-29. I²C Read Transaction - Dummy Write

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

Table 3-30. I²C Read Transaction - Byte Read

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

When a word write command (I2CPARAMS.*ACCESS_WIDTH* = 1b, I2CCMD.*OP* =0b) is given the following sequence is done by the X550:

Table 3-31. I²C Write Transaction - Word Write

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

When a byte write command (I2CPARAMS.*ACCESS_WIDTH* = 0b, I2CCMD.*OP* =0b) is given the following sequence is done by the X550:

Table 3-32. I²C Write Transaction - Byte Write

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

3.6 LEDs

The X550 implements four output drivers intended for driving external LED circuits per port. Each of the four LED outputs can be individually configured to select the particular event, state, or activity, which is indicated on that output. In addition, each LED can be individually configured for output polarity as well as for blinking versus non-blinking (steady-state) indication.

The configuration for LED outputs is specified via the LEDCTL register. Furthermore, the hardware-default configuration for all LED outputs can be specified via NVM fields (Section 6.2.5.5 and Section 6.2.5.6) thereby supporting LED displays configurable to a particular OEM preference.

Each of the four LED's can be configured to use one of a variety of sources for output indication. For more information on the MODE bits see LEDCTL register (see Section 8.2.2.1.10).

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

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

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

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

The mapping in Table 3-33 is used to specify the LED control source (MODE) for each LED output:

Table 3-33. LED Mapping

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

3.7 Network Interface

3.7.1 Overview

The X550 provides dual-port network connectivity with copper media. Each port includes integrated MAC-PHY functionality and can be operated at either 10 GbE, 1 GbE, 5GBASE-T, 2.5GBASE-T, or 100BASE-T(X) link speed. In terms of functionality there is no primary and secondary port as each port can be enabled or disabled independently from the other, and they can be set at different link speeds.

The integrated PHYs support the following specifications:

- 10GBASE-T as per the IEEE 802.3an standard.
- 1000BASE-T and 100BASE-TX as per the IEEE 802.3 standard.
- 2.5 and 5 Gb/s as per the NBASE-T specification.

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

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

3.7.2 Internal MDIO Interface

The X550 implements an internal IEEE 802.3 Management Data Input/Output Interface (MDIO Interface or MII Management Interface) between each MAC and its attached integrated PHY. This interface provides firmware and software the ability to monitor and control the state of the PHY. It provides indirect access to an internal set of addressable PHY registers. It complies with the new protocol defined by Clause 45 of IEEE 802.3 std. No backward compliance with Clause 22.

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

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

Software is notified that PHY initialization and/or reset has completed by either polling or by PHY reset done interrupt (see Section 3.7.3.4.4).

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

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

1. Address Cycle - Register MSCA is initialized with the appropriate PHY register address in *MDIADD DEVADD*, and *PORTADD* fields, the *OPCODE* field set to 00b and *MDICMD* bit set to 1b.
2. Poll MSCA.*MDICMD* bit until it is read as 0b.
3. Write Data Cycle - Data to be written is programmed in field MSRWD.*MDIWRDATA*.
4. Write Command Cycle - *OPCODE* field in the MSCA register is set to 01b for a write operation and the MSCA.*MDICMD* bit is set to 1b.
5. Wait for the MSCA.*MDICMD* bit to reset to 0b, which indicates that the transaction on the internal MDIO interface completed.

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

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

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

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

3.7.3 Integrated Copper PHY Functionality

3.7.3.1 PHY Performance

3.7.3.1.1 Reach

Table 3-34. BER and Ranges vs. Link Speed and Cable Types

Speed	Cable	Committed Reach	Committed BER
10GBASE-T	CAT-7	Full reach: 100 m	$< 10^{-12}$
	CAT-6a	Full reach: 100 m	
	CAT-6a	Short reach: 30 m	
	CAT-6a	Jumper mode / direct attach: 1 m	
	CAT-6	55 m	
	CAT-5e	1m	
1000BASE-T	CAT-5e	Full reach: 100 m	$< 10^{-10}$
100BASE-TX	CAT-5e	Full reach: 100 m	$< 10^{-8}$
2.5GBASE-T	CAT-5e	Full Reach: 100 m	$< 10^{-12}$
	CAT-6	Full Reach: 100 m	
5GBASE-T	CAT-5e	Full Reach: 100 m	$< 10^{-12}$
	CAT-6	Full Reach: 100 m	

Note: Reaches specified in the table refer to real cable lengths and not to the IEEE standard model.

3.7.3.1.2 MDI/Magnetics Spacing

The X550 supports a variable distance of from 1.5 to 4 inches with the magnetics.

3.7.3.1.3 Cable Discharge

The X550 is capable of passing the Intel cable discharge test. Contact your Intel representative for more details.

3.7.3.2 Auto-Negotiation and Link Setup

Link configuration is always determined by PHY auto-negotiation with the link partner. The X550 does not support parallel detect 100BASE-TX.

3.7.3.2.1 Automatic MDI Cross-Over and Lane Inversion

Twisted pair Ethernet PHYs must be correctly configured for MDI (no cross-over) or MDI-X (cross-over) operation to inter operate. This has historically been accomplished using special patch cables, magnetics pinouts or Printed Circuit Board (PCB) wiring. The PHY supports the automatic MDI/MDI-X configuration (like automatic cross-over detection) originally developed for 1000Base-T and standardized in IEEE 802.3 clause 40, at any link speed and also during auto-negotiation. Manual (non-automatic) MDI/MDI-X configuration is still possible via bits 1:0 of Auto-Negotiation Reserved Vendor Provisioning 1 register at address 7.C410.

In addition to supporting MDI/MDI-X, the PHY supports lane inversion (MDI swap) of the ABCD pairs to DCBA. It is useful for tab up or tab down RJ45 or integrated magnetics modules on the board. It is configurable via PHY register 1.E400.

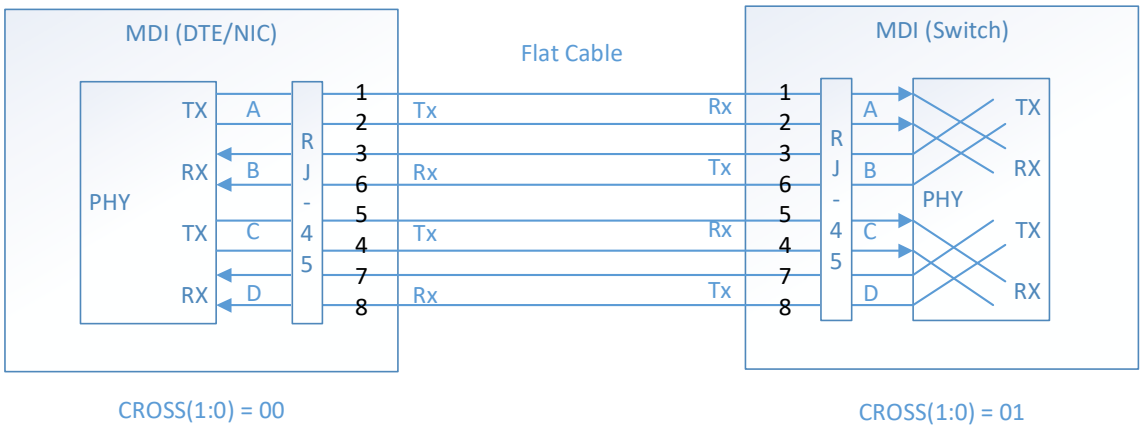


Table 3-35. Cross-Over Function (7.C410.1:0 = 00b)

3.7.3.2.2 Auto-Negotiation Process

The integrated copper PHY performs the auto-negotiation function. Auto-Negotiation provides a method for two link partners to exchange information in a systematic manner to establish a link configuration providing the highest common level of functionality supported by both partners. Once configured, the link partners exchange configuration information to resolve link settings such as:

- Speed: 100/1000 Mb/s or 10 Gb/s
- Link flow control operation (known as PAUSE operation)

Notes: When operating in Data Center Bridging (DCB) mode, generally, priority flow control is used instead of link flow control, and it is negotiated via higher layer protocol (DCBx protocol) and not via auto-negotiation. Refer to Section 3.7.4.

Each PHY is capable of successfully auto-negotiating with any device that supports 100 Mb/s or higher Ethernet, regardless of its method of Power over Ethernet (PoE) detection.

The X550 supports only full duplex mode of operation at any speed.

PHY specific information required for establishing the link is also exchanged.

If link flow control is enabled in the X550, the settings for the desired flow control behavior must be set by software in the PHY registers and auto-negotiation is restarted. After auto-negotiation completes, the software device driver must read the PHY registers to determine the resolved flow control behavior of the link and reflect these in the MAC register settings (FCCFG.*TFCE* and MFLCN.*RFCE*).

Once PHY auto-negotiation completes, the PHY asserts a link-up indication to the MAC that might notify software by an interrupt if the Link Status Change (LSC) interrupt is enabled. The resolved speed is also indicated by the PHY to the MAC. The status of both is directed to software via LINKS.*LINK_UP* and LINKS.*LINK_SPEED* bits.

3.7.3.2.2.1 Speed Resolution and Partner Presence

At the end of the auto-negotiation process, the link speed is automatically set to the highest common denominator between the abilities advertised by the link partners.

If there is no common denominator, the PHY asserts the *Device Present* bit (Auto-Negotiation Reserved Vendor Status 1: Address 7.C810, bit E) if it detected valid link pulses during auto-negotiation even though there is no common link speed with the link partner. This bit is valid only if auto-negotiation is enabled.

If the PHY training sequence cannot complete properly in spite of auto-negotiation completing, the PHY retries auto-negotiation for a programmable number of times (set by PHY register 7.C400: 3:0) before downshifting cyclically. Downshifting is enabled by PHY register 7.C400: 4. Automatic downshifting events are reported by the *Automatic Downshift* bit in PHY register 7.CC00.

3.7.3.2.2.2 Link Flow Control Resolution

Flow control is a function that is described in Clause 31 of the IEEE 802.3 standard. It allows congested nodes to pause traffic. Flow control is essentially a MAC-to-MAC function. PHYs indicate their MAC ability to implement flow control during auto-negotiation. These advertised abilities are controlled through two bits in the auto-negotiation registers (Auto-Negotiation Advertisement Register: Address 7.10), bits 5 and 6 for PAUSE and Asymmetric PAUSE, respectively.

After auto-negotiation, the link partner's flow control capabilities are indicated in Auto-Negotiation Link Partner Base Page Ability Register: Address 7.13, bits 5 and 6.

There are two forms of flow control that can be established via auto-negotiation: symmetric and asymmetric. Symmetric flow control was defined originally for point-to-point links; and asymmetric for hub-to-end-node connections. Symmetric flow control enables either node to flow-control the other. Asymmetric flow-control enables a repeater or switch to flow-control a DTE, but not vice versa.

Generally either symmetric PAUSE is used or PAUSE is disabled, even between a end-node and a switch.

Table 3-36 lists the intended operation for the various settings of ASM_DIR and PAUSE. This information is provided for reference only; it is the responsibility of the software to implement the correct function. The PHY merely enables the two MACs to communicate their abilities to each other.

Table 3-36. Pause and Asymmetric Pause Settings

Local and Remote ASM_DIR Settings	Local Pause Setting	Remote Pause Setting	Result
Both ASM_DIR = 1b	1	1	Symmetric — Either side can flow control the other.
	1	0	Asymmetric — Remote can flow control local only.
	0	1	Asymmetric — Local can flow control remote.
	0	0	No flow control.
Either or both ASM_DIR = 0b	1	1	Symmetric — Either side can flow control the other.
	Either or both = 0		No flow control.

3.7.3.3 PHY Initialization

3.7.3.3.1 PHY Boot

Each PHY has an Embedded Microprocessor (MCP). Each MCP has its own instruction RAM (IRAM) and Data RAM (DRAM). The MCP code/data segment and the PHY default configuration are fetched from the external Flash device, right after power-on reset and also per PHY MMD register set to force a reload (Global General Provisioning 3: Address 1E.C442, bit 0).

PHY access to the Flash device is controlled by the MAC. Assuming the PHY is granted by the MAC with back-to-back access to the Flash, the PHY initialization process should take less than 200 ms, at the end of which a PHY reset done interrupt is issued and/or reported in PHY register 1E.CC00.6.

Internal MDIO interface provides access to the PHY registers but it does not provide the software with the ability to overwrite the PHY image located in the NVM. MDIO access is done via dedicated MAC registers only.

The X550 maintains a CRC-16 (standard CCITT CRC $x^{16} + x^{12} + x^5 + 1$) over the PHY image in the NVM, and checks this on NVM loads. Inversion of the CRC after calculation is not required. If a CRC error occurs, the PHY image is reloaded again. If an error also occurs on the second try, the PHY is stopped and a fatal interrupt is generated to the host.

Default configuration read from the Flash overrides the default register values of the PHY. The same MCP code/data segment is auto-loaded to both PHYs, but each PHY has its own default configuration.

MCP code/data segment and default configuration read from the Flash are stored into internal Shadow RAMs. At PHY reset events, which are either issued by software (Global Standard Control 1: Address 1E.0000, bit F) or internally by the MAC, there is a reset of the micro controller; however there is no reload of ISRAM/DSRAM from the Flash. The micro controller begins executing instructions out of internal memory loaded from the previous Flash load. The same stands for PHY registers, which retrieves their default values loaded from the previous Flash load.

3.7.3.3.2 PHY Power-Up Operations

The integrated PHY is designed to perform the following operations at boot:

1. Power-up calibration of VCOs and power supplies.
2. Provision stored default values (from Flash into internal data RAM and then into PHY registers).
3. Calibration of the Analog Front-End (AFE).
4. Cable diagnostics.

5. Auto-negotiation.
6. Perform training (as required).
7. Verify error-free operation.
8. Enter steady state.

3.7.3.3.3 PHY Reset

Each PHY protects its data RAM via parity bits and its code RAM via ECC. In the event data corruption is detected, a PHY fault interrupt is issued (see Section 3.7.3.4.1).

Each PHY supports a watchdog timer to detect a stuck micro controller. Upon failure, a PHY fault interrupt is issued as well. Watchdog timer is set to 5 seconds by default.

The PHY is also reset on the same occasions that MAC is reset, except on software reset events for which the PHY does not get reset. A dedicated PHY reset command is provided to software instead, via a PHY register (Global Standard Control 1: Address 1E.0, bit F). Refer to Table 4-5.

At PHY reset events, all the PHY functionality go to reset including the micro controller except the PHY PLLs that go to reset only at power-up.

PHY reset completion is expected to take up to 5 ms, with no MDIO access during that time. PHY reset event causes link failure, which can take up to several seconds for resuming via auto-negotiation.

3.7.3.4 PHY Interrupts

The interrupt structure of each internal PHY is hierarchical in nature, and allows masking of all interrupts, at each of the levels of the hierarchy. The PHY has two interrupt hierarchies one is fully clause 45 compliant, the other is vendor defined, which is intended to allow determining the cause of an interrupt with only two status reads.

The values of these interrupt masks are visible via the internal MDIO interface in the vendor specific areas of each MMD, and the global summary register is located in the vendor specific area of the PHY registers (Global PHY Standard Interrupt Flags: Addresses 1E.FC00 and Global PHY Vendor Interrupt Flags: Addresses 1E.FC01).

The interrupt structure of each PHY is such that all standards-based interrupts can be read and cleared using a maximum of two PHY register reads.

There are two types of PHY interrupts according to their severity, normal or fatal:

- Fatal PHY interrupts are reported together with other fatal interrupts by the *ECC* bit in the EICR register. They concern the following events:
  - — ECC error when reading PHY micro controller code
  - — CRC error on the second attempt to load the PHY image from the NVM
  - — PHY micro controller watchdog failure
- Normal PHY interrupts are reported by the *PHY_GLOBAL_INTERRUPT* bit in EICR register. They concern all other PHY interrupt causes.

Note: The PHY micro controller never resets itself to a fatal interrupt or to any other event. The host is responsible to reset the link in such situations. The link is down until then.

Many of the interrupt causes are mostly useful to debug the PHY hardware. Therefore, they are masked by default and unless a specific need arises should remain so.

By default, Link State Change and Global Fault are the only interrupts that should be unmasked by software. To enable them software should set the following bits:

- 1E.FF01.C and 1E.FF01.2 — PHY vendor mask
- 1E.D400.4 — Enable chip fault interrupt
- 1E.FF00.8 — Enable standard autoneg interrupt 1

Additionally, software can enable an interrupt on reset complete:

- 1E.D400.6 — Enable reset done interrupt

3.7.3.4.1 PHY Fault Interrupt

In the event of a PHY fatal error, 1E.CC00.4 is set and an error code is written to 1E.C850. Software should log this code and attempt to reset the PHY.

Among others, a fatal interrupt is generated on one of the following events:

- CRC error over the PHY image when trying to load it from Flash twice without success
- ECC error on one of the PHY's internal memory that contains control data
- Watchdog failure of the PHY embedded micro controller

In reaction to a fatal error, the MAC drops the link until the fatal error is cleared. Software is therefore required to reset the link (not only the PHY) and retry to reload the PHY firmware as described in Section 3.7.3.4.2.

If three fatal PHY interrupts are handled with no link-up event in between, the link is considered to be down and the port must be disabled.

3.7.3.4.2 PHY Firmware Reload Flow

First read is 1E.FC00.0 = Vendor Alarm, followed by 1E.FC01, checking for bits 2 and 0 (Global Alarms 1 and 3 respectively) being set.

1. If any of them is set, check the following bits:
   - 1E.CC02.0 *Watchdog Timer Alarm*
   - 1E.CC00.4 *Device Fault* alarm (DRAM/IRAM non-correctable parity error)
   - 1E.CC02.A *DRAM Parity Error* alarm
   - 1E.CC02.9 Multi-bits *IRAM Parity Error* alarm; errors not corrected.
2. If any of them is set, initiate daisy-chain reload on the PHY that is corrupted by setting 1E.C442.0.
3. Wait for PHY interrupt and check that the reset is done by reading PHY register 1.CC02 and checking that bit 0 (*Reset Complete*) is set.
4. Resume normal operation.

3.7.3.4.3 Link State Change Interrupt

When an interrupt is caused by a change in the link state, bit 7.1.2 is latching low. The actual link state can be found in register 1.E800.0.

Table 3-37. PHY Link State Registers

Register Bits	Name	Description
7.C800 2:1	Connect Rate	0x0 = Reserved 0x1 = 100BASE-TX 0x2 = 1000BASE-T 0x3 = 10GBASE-T
7.C800 0	Connect Type (Duplex)	0b = Half 1b = Full
7.C810 F	Energy Detect	1b = Detected
7.C800 E	Far End Device Present	1b = Present
7.C800 D:9	Connection State	0x00 = Inactive (such as low-power or high-impedance). 0x01 = Cable diagnostics. 0x02 = Auto-negotiation. 0x03 = Training (10 GbE and 1 GbE only). 0x04 = Connected. 0x05 = Fail (waiting to retry autoneg). 0x06 = Test Mode. 0x07 = Loopback Mode. 0x08 = Reserved. 0x09 = Reserved. 0x0A = Reserved. 0x0B:0x10 = Reserved.

3.7.3.4.4 Reset Done Interrupt

If software has enabled the reset done interrupt, such an event generates an interrupt, which is indicated by bit 1E.CC00.6 being set. Note that a boot complete event is simultaneous with the reset event.

3.7.3.4.5 PHY Interrupt Handling Flow

Firmware is responsible to guarantee an operative PHY even when host is down or malfunctioning, to:

- Provide a remote access to MC from the network.
- Receive WoL packets.

When the host is down, interrupts from MAC blocks which are critical for MC/WoL are also handled by the firmware:

- ECC-Error from Security Rx/Tx blocks
- ECC-Error from Rx-Filter
- ECC-Error from DMA-Tx

3.7.3.4.5.1 Firmware to Software PHY Interrupt Handling Coordination

Note: As the software device driver does not handle PHY interrupt events, the flow below is not implemented and is provided as a guideline if implemented in the future. The software device driver needs to handle PHY interrupts.

Firmware cannot be sure the host is well functioning and consequently it always handles PHY interrupts first. Once it has completed to do its handling of PHY interrupts, firmware sets the relevant EEMNGCTL.*CFG_DONE0/1* bit and notifies the host it can start its own handling by issuing EICR.*MNG* interrupt. Since the PHY interrupt flags are cleared by read, the following flow must be run by host and firmware whenever a PHY interrupt occurs:

1. Host does not attempt to take ownership over the PHY semaphore until *CFG_DONE* bit is set by firmware.
   - In case the PHY semaphore is currently owned by the host, it stops accessing PHYINT_STATUS or PHY registers and releases the PHY ownership as soon as possible. Refer to Section 11.8.4 for the maximum semaphore ownership time allowed.
2. Firmware takes ownership of PHY semaphore
3. Firmware copies the PHY interrupt flags read from PHY registers into the PHYINT_STATUS registers
   - When writing PHYINT_STATUS registers firmware must not clear bits that were not cleared by the host yet
4. Firmware handles the PHY interrupt by resetting the PHY (only if it is a fatal PHY interrupt)
5. Firmware sets *CFG_DONE* bit, releases ownership of the PHY semaphore, and issues EICR.*MNG* interrupt to host.
6. Host takes semaphore ownership over the PHY.
7. Host reads the PHYINT_STATUS registers and clears them (by writing zeros).
8. Host handles the PHY interrupts.
   - Prior to doing a PHY re-configuration that might drop the link (e.g. restart auto-negotiation), the host must wait until the *VETO* bit is read as 0b.
9. Host releases PHY semaphore.

Notes: *CFG_DONE* bits are set by firmware and cleared by software. They cannot be cleared by firmware, and cannot be set by software.

For simplifying drivers, firmware runs the above flow even if there is no MC or WoL. No wake-up of the host occurs for the fatal PHY events handled by firmware.

PHYINT_STATUS registers and EEMNGCTL.*CFG_DONE* bits are reset by hardware only at power-up events.

3.7.3.5 Cable Diagnostics

The PHY implements a powerful cable diagnostic algorithm to accurately measure all of the TDR and TDT sequences within the group of four channels. The algorithm used transmits a pseudo-noise sequence with an amplitude of less than 300 mV for a brief period of time during startup. From the results of this measurement, the length of each pair, the top four impairments along the pair, and the impedance of the cable are flagged. These measurements are accurate to ±1 m under the assumption of the ISO 11801 cable propagation characteristics of 5.46 ns / m and are presented in the Global MMD register map.

Cable diagnostics performed by the PHY are listed in Table 3-38 and apply to all operating rates and scenarios.

Note: The PHY does not distinguish more than one feature within 5 m, the dominant source of reflection can be recorded as a single feature.

Each PHY completes the TDR measurement within 5 seconds of activation from an MDIO accessible register bit (1E.C470[4]. The PHY indicates that the operation is complete using the *Connection State* field in *Auto-Negotiation Reserved Vendor Status 1* register (7.C810.[D:9]). The PHY completes the entire cable diagnostic functions following a single access from the host.

Table 3-38. Proprietary Cable Diagnostics

Test	Feature	Notes
Distance to reflections: each of four pairs.	Distance to four reflection points for each pair.	±1 m resolution.
	Un-terminated end point (>300 Ω).	
	Terminated > 115 Ω or mismatch.	
	Terminated ≈ 100 Ω or compliant connector (if reflection is visible.	
	Terminated < 85 Ω or mismatch.	
	Short circuit < 30 Ω or mismatch.	
Pair-to-pair short.	Distance to short.	
	Suspected pairs.	

Cable diagnostics is able to detect all simple shorts on a cable. If the short is from one pair to any other pair, it is necessary to indicate the detail of the pair-pair short connectivity. It is recommended to use a mechanism that includes sending a signal on one pair and listening on all pairs. If a reflection is detected (greater than the limit allowed for NEXT) on any pair other than the one with the signal, it can be assumed that this is due to an inter-pair short.

Each PHY performs cable diagnostics at startup, prior to attempting auto-negotiation. Manual re-running of diagnostics can also be initiated by setting bit 4 of Global Reserved Provisioning 1 register (1E.C470).

3.7.3.6 False Training Detection

Each PHY restricts the amplitude of the cable diagnostics sequence to be less than 300 mV to avoid false detection of the training sequence by a parallel detection auto-negotiation block.

3.7.3.7 Low Power Operation and Power Management

The PHY incorporates numerous features to maintain the lowest power possible. Refer to Section 5.3.1.

3.7.4 Ethernet Flow Control (FC)

The X550 supports flow control as defined in 802.3x, as well as the specific operation of asymmetrical flow control defined by 802.3z. The X550 also supports Priority Flow Control (PFC), known as Class Based Flow Control, as part of the DCB architecture.

Note: The X550 can either be configured to receive regular FC packets or PFC packets. The X550 does not support receiving both types of packets simultaneously.

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

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

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

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

3.7.4.1 MAC Control Frames and Reception of Flow Control Packets

3.7.4.1.1 MAC Control Frame — Other than FC

The IEEE specification reserved the EtherType value of 0x8808 for MAC control frames, which are listed in Table 3-39.

Table 3-39. MAC Control Frame Format

Field	Description
DA	The *Destination Address* field can be an individual or multicast (including broadcast) address. Permitted values for the *Destination Address* field can be specified separately for a specific control OpCode such as FC packets.
SA	Port Ethernet MAC Address (six bytes).
Type	0x8808 (two 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	Four bytes.

3.7.4.1.2 Structure of 802.3X FC Packets

802.3X FC packets are defined by the following three fields (see Table 3-40):

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

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

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

Table 3-40. 802.3X Packet Format

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

Table 3-40. 802.3X Packet Format [continued]

Field	Description
Pad	42 bytes.
CRC	4 bytes.

3.7.4.1.3 Priority Flow Control

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

PFC is implemented to prevent the possibility of receive packet buffers overflow. Receive packet buffers overflow results in the dropping of received packets for a specific TC. Implement PFC by sending a timer indication to the transmitting station TC (XOFF) of a nearly full receive buffer condition at the X550. At this point the transmitter stops transmitting packets for that TC until the XOFF timer expires or a XON message is received for the stopped TC.

Similarly, once the X550 receives a priority-based XOFF it stops transmitting packets for that specific TC until the XOFF timer expires or XON packet for that TC is received.

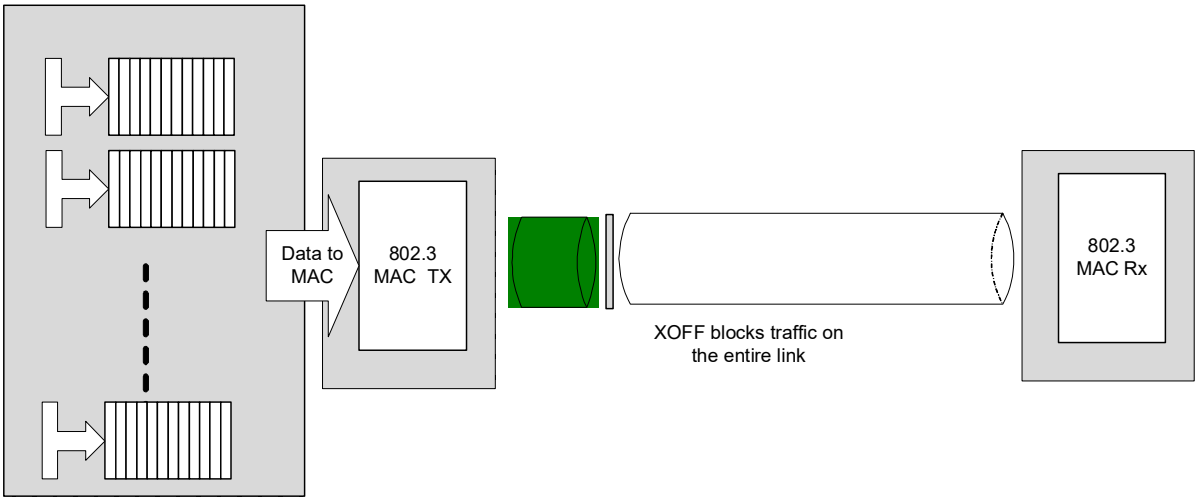


Figure 3-7. 802.3X Link Flow Control (PAUSE)

Link flow control (802.3X) causes all traffic to be stopped on the link. DCB uses the same mechanism of FC but provides the ability to do PFC on TCs, as shown in Figure 3-8.

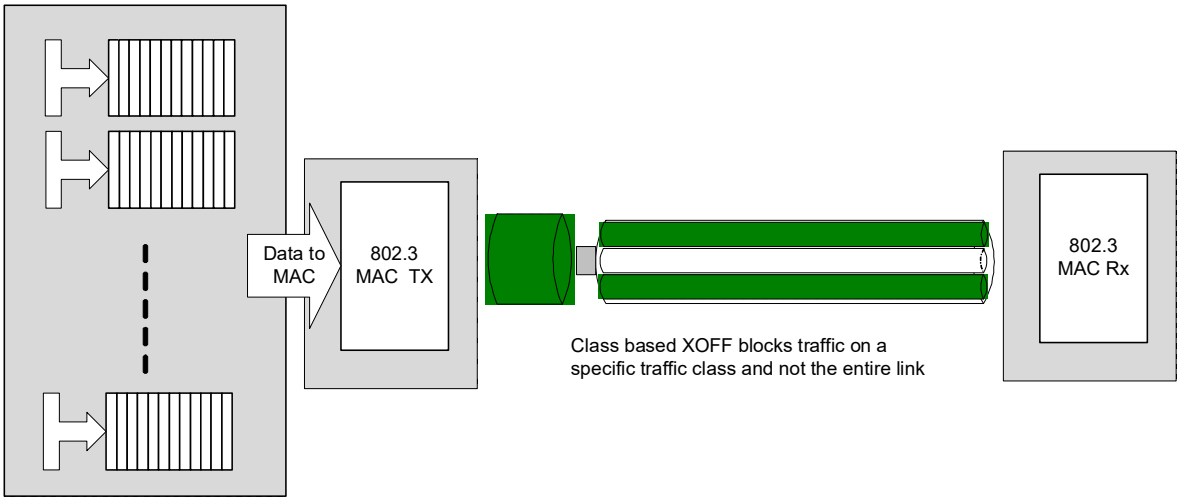


Figure 3-8. Priority Flow Control

Table 3-41. Packet Format for PFC

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

Table 3-42. Format of Priority Enable Vector

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

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

Each of the eight timers refers to a specific User Priority (UP), such as Timer 0 refers to UP 0, etc. The X550 binds a UP and timer to one of its TCs according to the UP-to-TC binding tables. Refer to the RTTUP2TC register for the binding of received PFC frames to Tx TCs, and to the RTRUP2TC register for the binding of transmitted PFC frames to Rx TCs.

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

When a PFC frame is formatted by the X550, the same values are replicated into every *Timer* field and priority enable vector bit of all the UPs bound to the associated TC. These values are configured in the RTRUP2TC register.

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

3.7.4.1.4 Operation and Rules

The X550 operates in either link FC or in PFC mode. Note that enabling both modes concurrently is not allowed:

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

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

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

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

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

Resumption of transmission can occur under the following conditions:

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

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

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

3.7.4.1.5 Timing Considerations

When operating at 10 GbE line speed, the X550 must not begin to transmit a (new) frame more than 74 pause quanta after receiving a valid Link XOFF frame, as measured at the wires (a pause quantum is 512 bit times).

When operating at 1 GbE line speed, the X550 must not begin to transmit a (new) frame more than 2 pause quanta after receiving a valid Link XOFF frame, as measured at the wires.

When operating at 100 Mb/s line speed, the X550 must not begin to transmit a (new) frame more than 1 pause quantum plus 64 bit times after receiving a valid Link XOFF frame, as measured at the wires.

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

3.7.4.2 PAUSE and MAC Control Frames Forwarding

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

- The *DPF* bit controls the transfer of PAUSE packets to the host. The same policy applies to both link FC and PFC packets as listed in Table 3-43. Note that any packet must pass the L2 filters as well.
- The *Pass MAC Control Frames* (*PMCF*) bit controls the transfer of non-PAUSE packets to the host. Note that when link FC frames are not enabled (RFCE = 0b), link FC frames are considered as MAC control frames for this case. Similarly, when PFC frames are not enabled (RPFCM = 0b), PFC frames are considered as MAC control frames as well.

Note: When virtualization is enabled, forwarded control packets are queued according to the regular switching procedure defined in Section 7.7.10.4.

Table 3-43. Transfer of PAUSE Packet to Host (DPF Bit)

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

3.7.4.3 Transmitting PAUSE Frames

The X550 generates PAUSE packets to ensure there is enough space in its receive packet buffers to avoid packet drop. The X550 monitors the fullness of its receive FIFOs and compares it with the contents of a programmable threshold. When the threshold is reached, the X550 sends a PAUSE frame. The X550 supports both link FC and PFC — but not both concurrently. When DCB is enabled, it only sends PFC, and when DCB is disabled, it only sends link FC.

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

3.7.4.3.1 Priority Flow Control (PFC)

The X550 operates in either a link 802.3X compliant mode or in a Priority Flow Control mode, but not in both at the same time.

The same watermarks mechanism is used for PFC and for 802.3X FC to determine when to send XOFF and XON packets. When PFC is used in the receive path, priority PAUSE packets are sent instead of 802.3X PAUSE packets. The format of priority PAUSE packets is described in Section 3.7.4.1.3.

Specific considerations for generating PFC packets:

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

3.7.4.3.2 Operation and Rules

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

The content of the Flow Control Receive Threshold High (FCRTH) register determines at what point the X550 transmits the first PAUSE frame. The X550 monitors the fullness of the receive FIFO and compares it with the contents of FCRTH. When the threshold is reached, the X550 sends a PAUSE frame with its pause time field equal to FCTTV.

At this time, the X550 starts counting an internal shadow counter (reflecting the pause time-out counter at the partner end). When the counter reaches the value indicated in FCRTV register, then, if the PAUSE condition is still valid (meaning that the buffer fullness is still above the low watermark), an XOFF message is sent again.

Once the receive buffer fullness reaches the low water mark, the X550 sends an XON message (a PAUSE frame with a timer value of zero). Software enables this capability with the XONE field of the FCRTL.

The X550 sends a PAUSE frame if it has previously sent one and the FIFO overflows. This is intended to minimize the amount of packets dropped if the first PAUSE frame did not reach its target.

3.7.4.3.3 Flow Control High Threshold — FCRTH

The X550 sends a PAUSE frame when the Rx packet buffer is full above the high threshold. The threshold should be large enough to overcome the worst case latency from the time that crossing the threshold is sensed until packets are not received from the link partner.

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

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

Three values are envisaged for MaxFrame:

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

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

> Std DV = MaxFrame(TC) + MaxFrame(link) + PFC Frame + 2 x Cable Delay + 2 x Interface Delay + Higher Layer Delay x Sec Y Transmit Delay
>
> Std DV (bit time units) = MaxFrame(TC) + MaxFrame(link) + 672 + 2 x 5,556 + 2 x (25,600 + 8,192 + 2 x 2,048) + 6,144 x (MaxFrame(link) + 3,200)

MaxFrame(TC) term and MaxFrame(link) term included in Sec Y Transmit Delay correspond to worst case scenarios issued by the link partner. All other terms in Std DV formula must take in account worst case incoming traffic pattern which would lead to worst case buffer utilization as per the internal architecture of Rx packet buffer in the X550.

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

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

It leads to the below revised formula for the X550:

> X550 DV (bit time units) = 1.44 x [(MaxFrame(link) + 672 + 2 x 5,556 + 2 x (25,600 + 8,192 + 2 x 2,048) + 6,144 x (3,200)] + MaxFrame(TC) + MaxFrame(TC) x MaxFrame(link)

FCRTH must be set to the size of the Rx packet buffer allocated to the traffic class minus the X550 DV.

Table 3-44. X550 Delay Values (DV) Used for FCRTH

9.5 KB Jumbo Enabled	FCoE Traffic Class	X550 DV
No	No	24 KB
No	Yes	25 KB
Yes	No	50 KB
Yes	Yes	35 KB
No	No	27 KB
No	Yes	27 KB
Yes	No	60 KB
Yes	Yes	45 KB

Note: 9.5 KB Jumbo enabled/disabled is a global setting per port which concerns all Traffic Classes except the FCoE Traffic Class.

3.7.4.3.4 FC Low Threshold — FCRTL

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

FCRTL = 2 x MaxFrame(TC) + PCIe round trip delay

PCIe round trip delay is assumed to be ~ 1 us and it must cover for worst case incoming traffic pattern (buffer utilization by 1.44 than wire rate):

FCRTL (bit time units) = 2 x MaxFrame(TC) + 1.44 x 10,000

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

Table 3-45. X550 FCRTL

9.5 KB Jumbo Enabled	FCoE Traffic Class	X550 DV
No	No	5 KB
No	Yes	7 KB
Yes	No	21 KB
Yes	Yes	7 KB

3.7.4.3.5 Packet Buffer Size

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

- For a PFC-enabled TC:
  - FCRTH = FCRTL + hysteresis budget = FCRTL + Max (MaxFrame(TC), 4.5 x 1024 B)
  - Rx Packet Buffer size = FCRTH + the X550 DV (see Section 3.7.4.3.3)
- For a best effort TC:
  - Rx Packet Buffer size = FCRTL, as the same considerations than described in Section 3.7.4.3.4 play here to avoid bubbles over PCIe

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

Table 3-46 assumes four PFC-enabled Traffic Classes are defined over the port, of which two are allocated to FCoE traffic and two for other loss less traffic types like iSCSI, etc. The table lists the recommended settings for the supported combinations. When less than 4 PFC-enabled TCs are defined, and/or when less than 8 TCs are defined, it is recommended to refer to the setting rules described in this section, in Section 3.7.4.3.3, and in Section 3.7.4.3.4. Note that reducing the number of TCs of a port to what is really needed, helps increasing the port's throughput.

Table 3-46. Some Recommended Rx Packet Buffer Settings

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

Table 3-46. Some Recommended Rx Packet Buffer Settings [continued]

Flow Director Table Size	9.5 KB Jumbo Enabled	Packet Buffer Size of Any of the 4 Best Effort TCs	Packet Buffer Size of Any of the 2 FCoE TCs	Packet Buffer Size of Any of the Other 2 PFC-Enabled TCs
64 KB	Yes	14 KB	49 KB FCRTL = 3 KB FCRTH = 4 KB	83 KB FCRTL = 18 KB FCRTH = 23 KB
128 KB	No	17 KB	46 KB FCRTL = 7 KB FCRTH = 21 KB	45 KB FCRTL = 6 KB FCRTH = 21 KB
128 KB	No	16 KB	48 KB FCRTL = 7 KB FCRTH = 21 KB	47 KB FCRTL = 6 KB FCRTH = 20 KB

Notes:

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

3.7.4.4 Link FC in DCB Mode

When operating in DCB mode, PFC is the preferred method of getting the best use of the link for all TCs. When connecting to switches that do no support (or enable) PFC, the X550 can also throttle the traffic according to incoming link FC notifications. Following is the required device setting and functionality.

- The X550 should be set to legacy link FC by setting MFLCN.*RFCE*.
- Receive XOFF pauses transmission in all TCs.
- Crossing the Rx buffer high threshold on any TC generates XOFF transmission. Each TC can have its own threshold configured by the FCRTH[n] registers.
- Crossing the Rx buffer Low threshold on any TC generates XON transmission. This behavior is undesired. Therefore, software should not enable XON in this mode by clearing FCRTL[n].*XONE* bits in all TC.
- The FCTTV of all TCs must be set to the same value.

3.7.5 Inter Packet Gap (IPG) Control and Pacing

The X550 supports transmission pacing by extending the IPG (the gap between consecutive packets). The pacing mode enables the average data rate to be slowed in systems that cannot support the full link rate (10 GbE, 1 GbE or 100 Mb/s). As listed in Table 3-47, the pacing modes work by stretching the IPG in proportion to the data sent. In this case the data sent is measured from the end of preamble to the last byte of the packet. No allowance is made for the preamble or default IPG when using pacing mode.

Example 1:

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

Example 2:

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

Table 3-47 lists the pacing configurations supported by the X550 at link rates of 10 GbE. When operating at lower link speeds the pacing speed is proportional to the link speed.

Table 3-47. Pacing Speeds at 10 GbE Link Speed

Pacing Speeds (Gb/s)	Delay Inserted into IPG	Register Value
10 (LAN)	None	0000b
9.294196 (WAN)	1 byte for 13 transmitted	1111b
9.0	1 DWord for 9 transmitted	1001b
8.0	1 DWord for 4 transmitted	1000b
7.0	3 DWords for 7 transmitted	0111b
6.0	2 DWords for 3 transmitted	0110b
5.0	1 DWords for 1 transmitted	0101b
4.0	3 DWords for 2 transmitted	0100b
3.0	7 DWords for 3 transmitted	0011b
2.0	4 DWords for 1 transmitted	0010b
1.0	9 DWords for 1 transmitted	0001b
10	None	Default

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

Chapter 4 Initialization

4.1 Power Up

4.1.1 Power-Up Sequence

Figure 4-1 shows the X550's power-up sequence from power ramp up until it is ready to accept host commands.

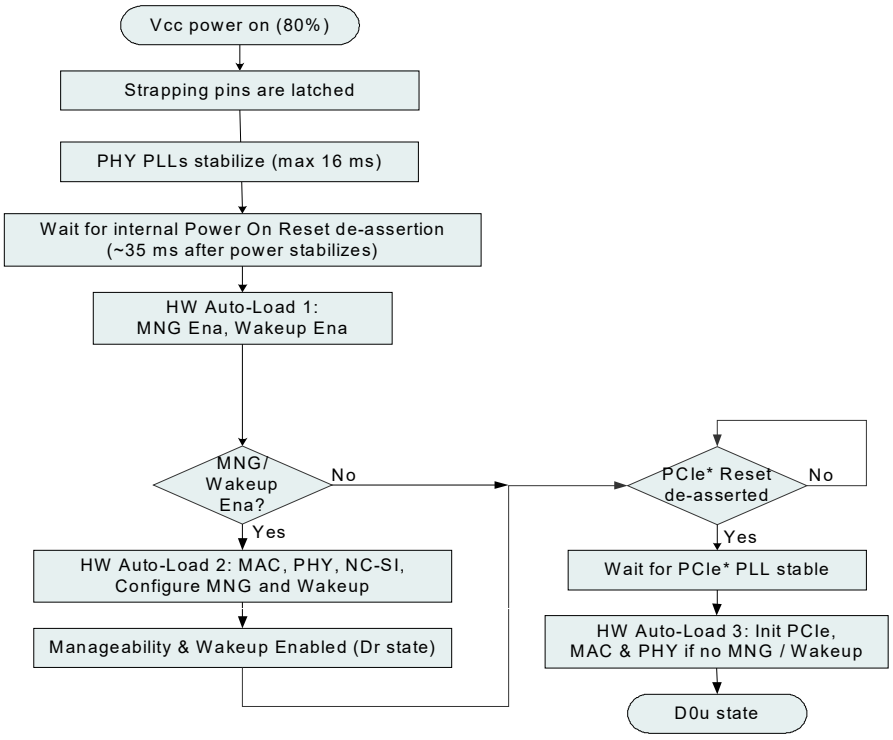


Figure 4-1. X550 Power-Up Sequence

4.1.2 Power-Up Timing Diagram

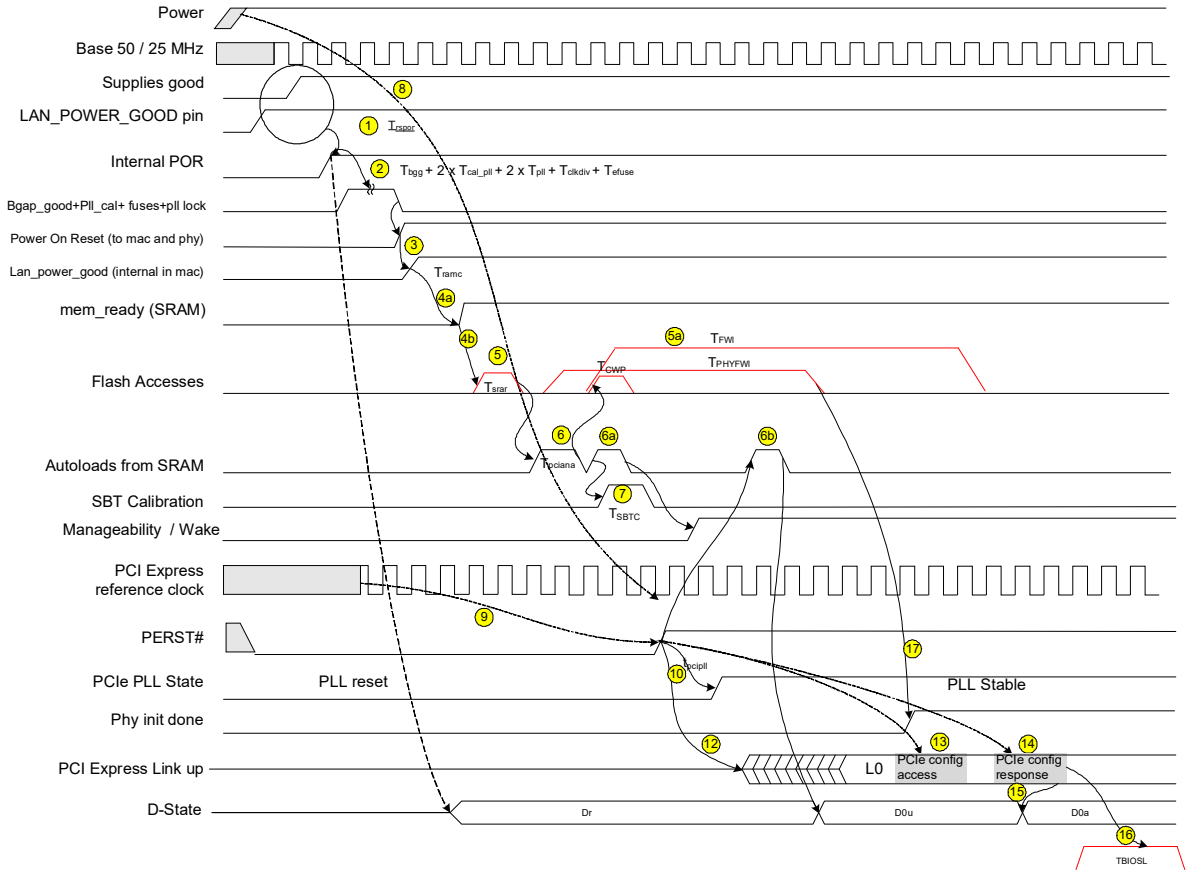


Figure 4-2. Power-Up Timing Diagram

Table 4-1. Notes for Power-Up Timing Diagram

No.	Note
1	Internal Power On Reset (POR) is released Trspor after power is stable and the reset pin has been de-asserted, releasing PLL kickstart.
2	Bgap indication is checked. If it is good, the PLL starts calibration and lock mechanism. Once PHY PLLs are locked, the efuses are read and the clock dividers are released to provide stable clocks. Finally, the PHY releases the POR signal to the MAC.
3	MAC starts its MAC-power-up-flow and internal LAN_PWR_Good signal is asserted (clocked on 50 MHz). At this stage, internal clocks are stable.
4a	Clear RAM contents
5	Shadow RAM read.
5a	Clear Flash write protection, Interleaved Manageability and PHY firmware load (in parallel). The FLASH accesses are active at the same time, but with only one SPI bus, the indicated times must be summed, arbitration determines which finishes first

Table 4-1. Notes for Power-Up Timing Diagram [continued]

No.	Note
6	NVM read starts following the read of the Shadow RAM. First Flash auto-read sequence is owned by the MAC to load PCI analog section, LAN and PCIe sections, APM enable, and other configuration words from legacy NVM initialization section. A PERST# de-assertion restarts part of the auto-load process.
5	APM wake-up and/or manageability active, based on NVM contents (if enabled).
6	In this second MAC auto-load sequence, MAC manageability and wake-up modules are loaded (if manageability / wake-up enabled).
6a	First auto-load of LAN sections: MAC module if manageability is enabled.
6b	Second auto-load of LAN sections PCIe general configuration; PCIe configuration space; LAN core modules, and MAC, module if manageability is not enabled.
7	SVR calibration - should be done before PERST# de-assertion
8	PERST# is de-asserted by minimum t_{PVPGL} after power is stable (PCIe specification).
9	The PCIe reference clock is valid $t_{PWRGD-CLK}$ before the de-assertion of PERST# (PCIe specification).
10	De-assertion of PERST# to PCIe PLL stable t_{PCIPLL}.
12	PCIe link training starts after t_{pgtrn} from PERST# de-assertion (PCIe specification).
13	A first PCIe configuration access might arrive after t^{pgcfg} from PERST# de-assertion (PCIe specification).
14	A first PCI configuration response can be sent after t_{pgres} from PERST# de-assertion (PCIe specification).
15	Setting the *Memory Access Enable* or *Bus Master Enable* bits in the PCI Command register transitions the X550 from D0u to D0 state.
16	BIOS software reads the PCIe driver and iSCSI/FCoE boot code from Flash, via expansion ROM.
17	***Note:*** PHY Init Done is a per port indication. This indication is used by Windows drivers (by either polling or interrupt notification) to know that PHY Device IDs are readable via MDIO. The requirement is that it is readable within a maximum of 200 ms after software reset.

4.1.2.1 Timing Requirements

The X550 requires the following start-up and power-state transitions.

Table 4-2. Power-Up Timing Requirements

Parameter	Description	Min	Max	Notes
t_{xog}	Base 50 MHz clock stable from power stable.		10 ms	
$t_{PWRGD-CLK}$	PCIe clock valid to PCIe power good.	100 µs	-	According to PCIe specification.
t_{PVPGL}	Power rails stable to PCIe reset inactive.	100 ms	-	According to PCIe specification.
t_{pgcfg}	External PCIe reset signal to first configuration cycle.	100 ms		According to PCIe specification.

Note: It is assumed that the external 50 MHz clock source is stable after power is applied; the timing for that is part of t_{xog}.

4.1.2.2 Timing Guarantees

The X550 guarantees the following start-up and power-state transition related timing parameters.

Table 4-3. Power-Up Timing Guarantees

Parameter	Description	Min	Max	Notes
t_{xog}	Xosc stable from power stable.		10 ms	
T_{rspor}	Internal POR from power stable.	21,000 µs	21,000 µs	Uses an internal timer based on 50MHz clock.
T_{bgg}	Bandgap good indication is checked.	5,280 µs	10,560 µs	If the bandgap does not report good, kickstart is issued and the bandgap status is checked again after T_{bgg}.
T_{cal_pll}	VCO calibration of one PLL	41 µs	5,200 µs	worst case timing is when this step needs to iterate.
T_{pll}	Lock detection of one PLL is checked.	5,280 µs	10,560 µs	If PLL is not locked, kickstart is issued and the PLL lock status is checked again after T_{pll}.
T_{clkdiv}	Release of reset to the clock dividers.	1.1 µs	1.1 µs	
T_{efuse}	Read EFUSE content.	500 µs	500 µs	
t_{ppg}	Internal MAC power good delay from valid power rail.	37,423 µs	63,581 us	Typically the time is no worse than 42 ms.
t_{fl}	Flash auto-read duration.		3 ms	
t_{opll}	PCIe reset to start of link training.		10 ms	
t_{pcipll}	PCIe reset to PCIe PLL stable.		3 ms	
t_{pgtrn}	PCIe reset to start of link training.		20 ms	According to PCIe specification.
t_{pgres}	PCIe reset to first configuration response cycle.	100 ms		According to PCIe specification.
T_{ramc}	RAM clear time	81 µs		
T_{srar}	Shadow RAM read time	10.8 ms		
T_{FWI}	Manageability firmware load	300 ms	380 ms	Assumes 12.5 MHz clock
T_{PHYFWI}	PHY firmware load	160 ms	240 ms	Assumes 22.5 MHz clock
T_{CWP}	Clear Write protect	100 µs		
T_{PCIana}	PCI Analog Section auto-load	65 ms		
T_{SBTC}	PCI Analog Section auto-load	7 ms	15 ms	
T_{BIOSL}	BIOS Expansion ROM auto-load	700 ms	900 ms	Assumes 12.5 MHz clock

4.1.3 Main-Power/Aux-Power Operation

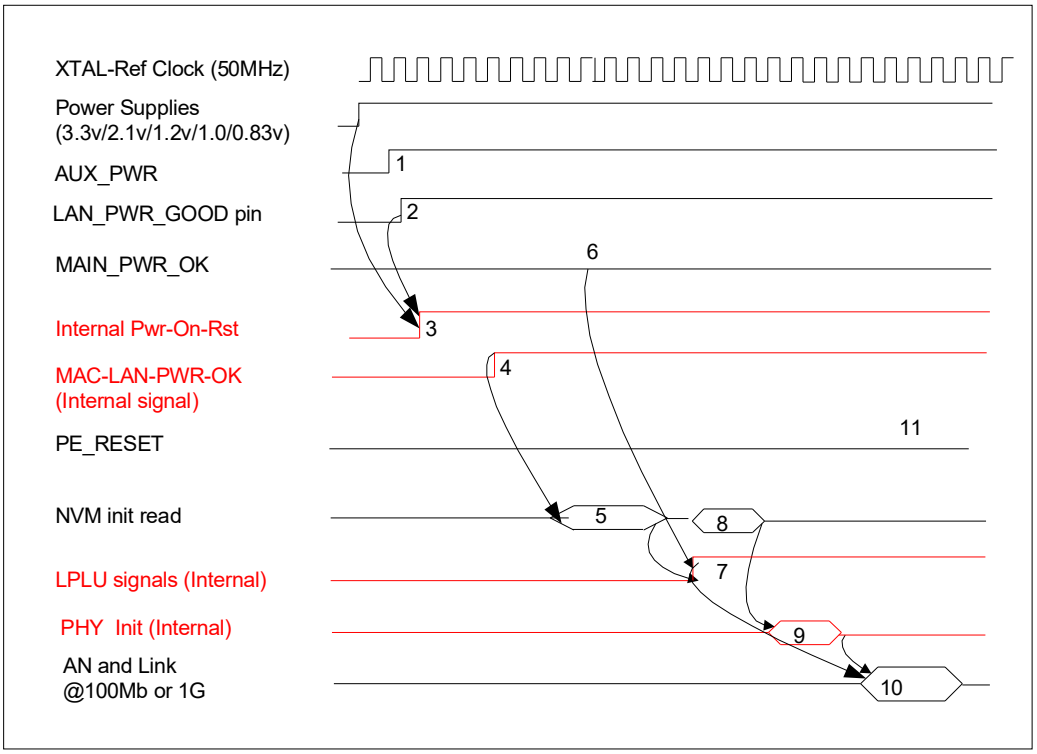


Figure 4-3. Low-Power Modes Timing Diagram

Table 4-4. Notes for Low-Power Modes Timing Diagram

No.	Note
1	AUX_PWR set to 1b, indicates that the X550 should support Aux-Power Mode.
2	LAN_PWR_GOOD signal de-asserted.
3	Internal Power-On-Reset de-asserted, indicates to all the X550 blocks that all power-rails are OK and the external Reset signal is de-asserted (reset_n).
4	MAC_LAN_PWR_OK de-asserted (internal signal), indicates that all clocks are stable.
5	MAC reads configuration from NVM.
6	Main-Power-OK still de-asserted, indicates that the system still runs from Aux-Power.
7	LPLU signals asserted from MAC to PHY, indicates the PHY needs to operate at low-speed (100 Mb/s or GbE)
8	PHY reads configuration from NVM.
9	PHY Init completes.
10	PHY auto-negotiation to 100 Mb/s or GbE and establishes link (assuming LP available).
11	PE_RESET is still de-asserted, indicates that the system is still down, keep PCIe at reset, and the MAC at low-power modes (WoL or MNG).

4.2 Reset Operation

4.2.1 Reset Sources

The X550 reset sources are described in the sections that follow. Figure 4-4 and Figure 4-5 describes the hierarchy between the different reset causes.

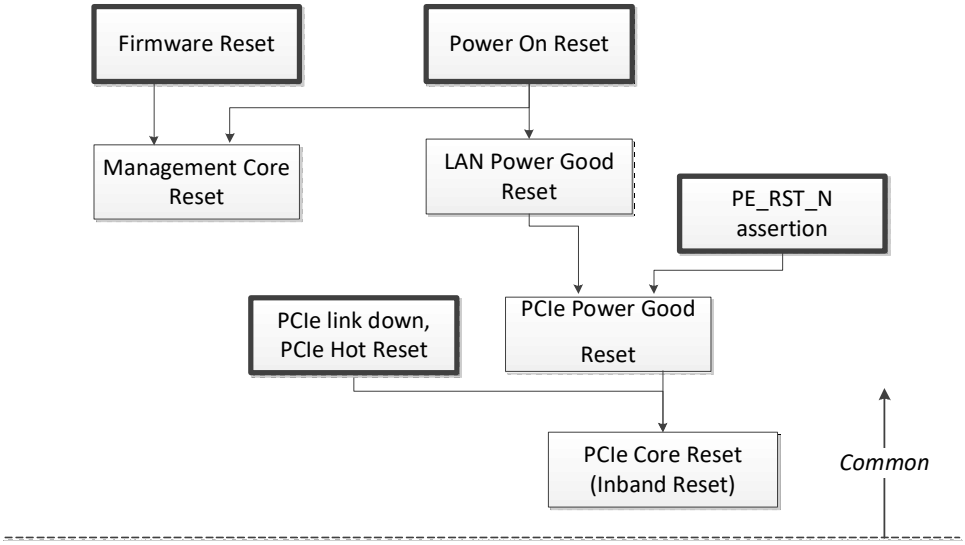


Figure 4-4. Reset Tree (Common)

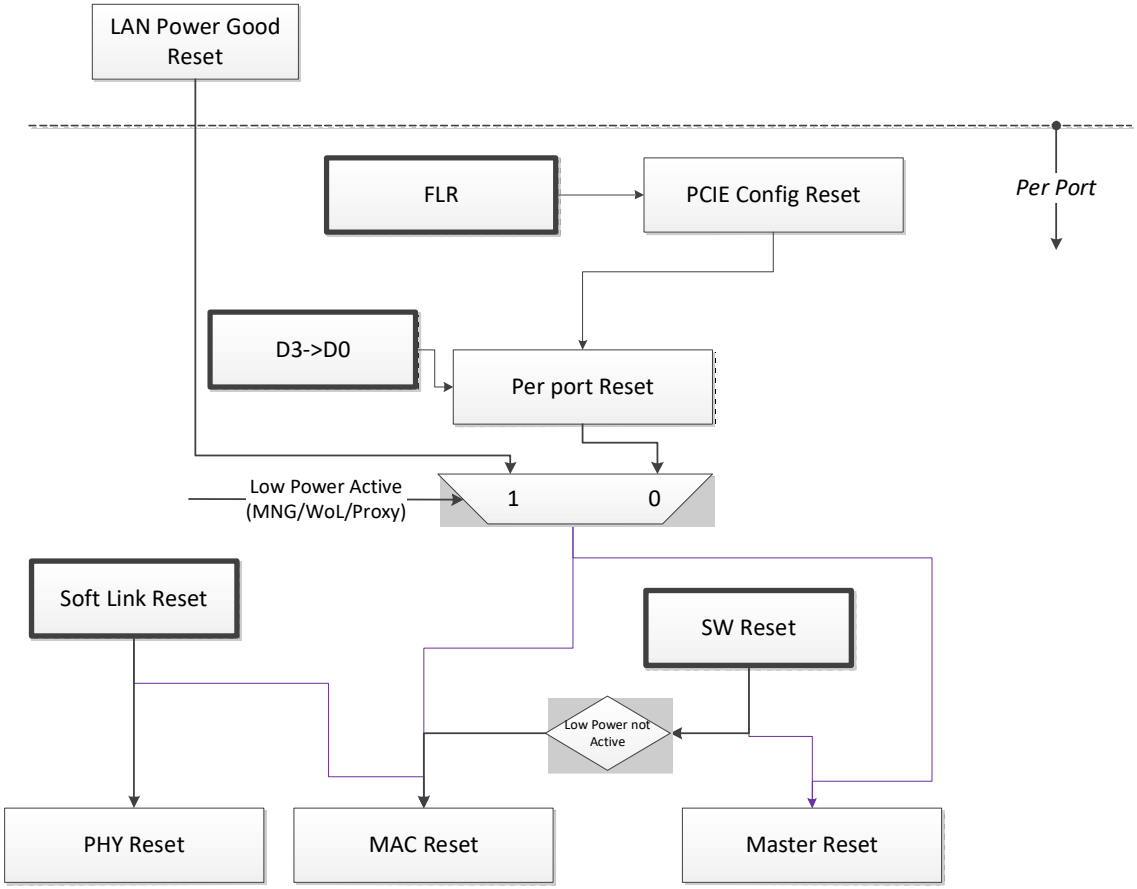


Figure 4-5. Reset Tree (per Port)

4.2.1.1 LAN_PWR_GOOD

The X550 has an internal mechanism for sensing power pins. Once power is up and stable, the X550 creates an internal reset, which acts as a master reset of the entire chip. It is level sensitive, and while it is 0b, all of the registers are held in reset. LAN_PWR_GOOD is interpreted to be an indication that device power supplies are all stable. Note that LAN_PWR_GOOD changes state during system power-up.

4.2.1.2 PE_RST_N (PCIe Reset)

De-asserting PCIe reset indicates that both the power and the PCIe clock sources are stable. This pin also asserts an internal reset after a D3cold exit. Most units are reset on the rising edge of PCIe reset. The only exception is the GIO unit, which is kept in reset while PCIe reset is asserted (level).

4.2.1.3 In-Band PCIe Reset

The X550 generates an internal reset in response to a physical layer message from PCIe or when the PCIe link goes down (entry to a polling or detect state). This reset is equivalent to PCI reset in previous PCI Gigabit Ethernet (GbE) controllers.

4.2.1.4 D3hot to D0 Transition

This is also known as ACPI reset. The X550 generates an internal reset on the transition from D3hot power state to D0 (caused after configuration writes from a D3 to D0 power state). Note that this reset is per function and resets only the function that transitions from D3hot to D0 and do not reset the config space of the function. Its effect is equivalent to a software reset (Section 4.2.1.6.1).

4.2.1.5 Function Level Reset (FLR) Capability

The *FLR* bit is required for the Physical Function (PF) and per Virtual Function (VF). Setting this bit for a VF only resets the part of the logic dedicated to the specific VF and does not influence the shared part of the port. Setting the PF *FLR* bit resets the entire function.

4.2.1.5.1 FLR in Non-IOV Mode

A FLR reset to a function is equivalent to a D0 –> D3 –> D0 transition with the exception that this reset does not require driver intervention to stop the master transactions of this function. FLR affects the device 1 parallel clock cycle from FLR assertion by default setting, or any other value defined by the *FLR Delay Disable* and *FLR Delay* fields in the *PCIe Init Configuration 2 — Offset 0x02* word in the NVM.

4.2.1.5.2 Physical Function FLR (PFLR)

A FLR reset to the PF function in an IOV mode is equivalent to a FLR reset in non-IOV mode. All VFs in the PCIe function of the PF are affected.

The affected VFs are not notified of the reset in advance. The *RSTD* bit in VFMailbox[n] is set following the reset (per VF) to indicate to the VFs that a PF FLR took place. Each VF is responsible to probe this bit (such as after a timeout).

4.2.1.5.3 Virtual Function FLR (VFLR)

A VF operating in an IOV mode can issue a FLR. VFLR resets the resources allocated to the VF (like disabling the queues and masking interrupts). It also clears the PCIe configuration for the VF. There is no impact on other VFs or on the PF.

Tx and Rx flows for the queues allocated to this VF are disabled. All pending read requests are dropped and PCIe read completions to this function can be completed as unsupported requests.

Note: Clearing the *IOV Enable* bit in the IOV structure is equivalent to a VFLR to all VFs in the same port.

A VFLR does not release queues that were blocked due to malicious events. The PF device driver needs to release them using the matching bit in WQBR_RX and WQBR_TX registers. The matching bits in these registers should be cleared after each VFLR, even if not caused due to a malicious event.

Note: PF driver should clear the VF's VFMBMEM after a VFLR is detected.

4.2.1.6 Soft Resets

4.2.1.6.1 Software Reset

Software reset is done by writing to the *Device Reset* bit of the Device Control (CTRL.*RST*) register. The X550 re-reads the per-function NVM fields after software reset. Bits that are not normally read from the NVM are reset to their default hardware values.

Note: This reset is per function and resets only the function that received the software reset.

PCI configuration space (configuration and mapping) of the device is unaffected. The MAC might or might not be reset (see Section 4.2.3).

Prior to issuing a software reset, the software driver needs to execute the master disable algorithm as defined in Section 5.2.4.3.2.

If DCB is enabled, following a software reset the steps below must be executed to prevent potential races between manageability mapping to TC before and after initialization.

- Clear the Flow Control enablement in the MAC by clearing the MFLCN.*RFCE* (or simply clear the whole register)
- Software should waits ~10µs
- The software polls the TFCS.*TC_XON*[0] = 0 (most of the time it is expected to be found at zero while max poll time is always shorted than the max expected PAUSE time before the software reset initiated)
- The software maps the Manageability transmit TC (setting the MNGTXMAP register) and then maps the user priority of Manageability traffic to the Manageability TC (setting the RTRUP2TC and RTTUP2TC registers)
- The software waits ~10µs
- The software can re-enable the Flow Control as part of the rest of the init flow

4.2.1.6.2 Physical Function Software Reset

A software reset by the PF in IOV mode has the same consequences as a software reset in a non-IOV mode.

The procedure for PF software reset is as follows:

- The PF driver disables master accesses by the device through the master disable mechanism (see Section 5.2.4.3.2). Master disable affects all VFs traffic.
- Execute the procedure described in Section 4.2.2 to synchronize between the PF and VFs.

VFs are expected to timeout and check on the *RSTD* bit to identify a PF software reset event. The *RSTD* bits are cleared on read.

4.2.1.6.3 VF Software Reset

A software reset applied by a VF is equivalent to a FLR reset to this VF with the exception that the PCIe configuration bits allocated to this function are not reset. It is activated by setting the VTCTRL.*RST* bit.

4.2.1.6.4 Force TCO

This reset is generated when manageability logic is enabled. It is only generated if enabled by the *Force TCO Reset* bit in the *Common Firmware Parameters* word in the NVM. If enabled by the NVM, the firmware triggers a port reset by setting the CTRL.*LRST* bit. In pass-through mode it is generated when receiving a Force TCO SMBus or NC-SI command with bit 0 set.

4.2.1.7 Link Reset

Also referred to as MAC reset.

Initiated by writing the *Link Reset* bit of the Device Control register (CTRL.*LRST*).

A link reset is equivalent to a software reset + reset of the MAC + reset of the PHY. The X550 re-reads the per-function NVM fields after link reset. Bits that are normally read from the NVM are reset to their default hardware values. Note that this reset is per function and resets only the function that received the link reset.

The PF in IOV mode can generate a link reset.

Prior to issuing link reset the software driver needs to execute the master disable algorithm as defined in Section 5.2.4.3.2.

4.2.1.8 PHY Resets

Software can reset each PHY separately via MDIO, by setting the corresponding *Soft Reset* bit in the Global Standard Control 1 register. It resets all PHY functionalities expected to PLLs; however, the PHY image is not reloaded from the NVM.

Software can at once reset a PHY (excluding PLLs) and cause a PHY image reload from the NVM, for each PHY separately, by setting via MDIO the corresponding PHY *Image Reload* bit in the Global Standard Control 1 register.

A PHY reset event causes a link down and restarts the auto-negotiation process. This can take a few seconds to complete, which might result in drop of the TCP sessions with the host and/or with a Manageability Controller (MC).Because the PHY can be accessed by the MC (via internal firmware) and by the driver software concurrently, the driver software should coordinate any PHY reset with the firmware using the following procedure:

1. Ensure that the MMNGC.*MNG_VETO* bit is cleared. If it is set, the MC requires a stable link and thus the PHY should not be reset at this stage. The software driver can skip the PHY reset (if it is not mandatory) or wait for this bit to be cleared by the MC. See Section 5.3.5.2 for more details on *MNG_VETO* bit.
2. Take ownership of the relevant PHY using the flow described in Section 11.8.4.
3. Set the *PHY Reset* bit in the Global Standard Control 1 register (or bit 0 in PHY register 1E.C442 for a PHY *Image Reload* event).
4. For a PHY reset, wait for 2 μs before initiating any MDIO access.
5. For a PHY image re-load and before initiating any MDIO access, do one of the following:
   a. Wait for 100 ms and poll *Global Reset Completed* bit in PHYINT_STATUS2 register until it is set by firmware.
   b. Wait to receive a PHY reset done interrupt.
6. Release ownership of the relevant PHY using the flow described in Section 11.8.4.

4.2.2 Reset in PCI-IOV Environment

Several mechanisms are provided to synchronize reset procedures between the PF and the VFs.

4.2.2.1 RSTI/RSTD

This mechanism is provided specifically for a PF software reset but can be used in other reset cases as follows.

- One of the following reset cases takes place:
  - LAN Power Good
  - PCIe reset (PE_RST_N and in-band)
  - D3hot --> D0
  - FLR
  - Software reset by the PF
- The X550 sets the *RSTI* bits in all the VFMailbox registers. Once the reset completes, each VF can read its VFMailbox register to identify a reset in progress.
  - The VF can poll the *RSTI* bit to detect if the PF is in the process of configuring the device.
- Once the PF completes configuring the device, it sets the CTRL_EXT.*PFRSTD* bit. As a result, the X550 clears the *RSTI* bits in all the VFMailbox registers and sets the *Reset Done* (*RSTD*) bits in all the VFMailbox registers.
  - The VF might read the *RSTD* bit to detect that a reset has occurred. The *RSTD* bit is cleared on read.

4.2.2.2 VF Receive Enable (PFVFRE)/VF Transmit Enable (PFVFTE)

This mechanism insures that a VF cannot transmit or receive before the Tx and Rx path have been initialized by the PF.

- The PFVFRE register contains a bit per VF. When the bit is set to 0b, assignment of an Rx packet for the VF's pool is disabled. When set to 1b, the assignment of an Rx packet for the VF's pool is enabled.
- The PFVFTE register contains a bit per VF. When the bit is set to 0b, fetching data for the VF's pool is disabled. When set to 1b, fetching data for the VF's pool is enabled. Fetching descriptors for the VF pool is maintained, up to the limit of the internal descriptor queues — regardless of PFVFTE settings.

PFVFTE and PFVFRE are initialized to zero (VF Tx and Rx traffic gated) following a PF reset. The relevant bits per VF are also initialized by a VF software reset or VFLR.

4.2.3 Reset Effects

The resets listed in Section 4.2.1 affect the following registers and logic:

Table 4-5. Reset Effects — Common Resets

Reset Activation	LAN Power Good	PCIe PE_RST_N	In-Band PCIe Reset	Firmware Reset	Notes
NVM read	See Section 6.1.3				
LTSSM (back to detect/polling)	X	X	X		
PCIe link data path	X	X	X		
PCI configuration registers (RO)	X	X	X		8
PCI configuration registers (RW)	X	X	X		8
PCIe local registers	X				
Data path	X	X	X		2, 7
MAC, TimeSync	X	X [6]	X [6]		
PHY (excluding PLLs)	X	X [6]	X [6]		15
PCIe analog, PHY PLLs	X				
Wake-up (PM) Context	X	1			3
Wake-up/manageability control/status registers	X				4, 5
Manageability unit	X			X	
LAN disable strapping pins	X	X	X		
All other strapping pins	X				
Shadow RAMs in MAC or PHY	X				

Table 4-6. Reset Effects — per Function Resets

Reset Activation	D3 or Dr	FLR or PFLR	Software Reset	Link Reset or Exit from LAN Disable	PHY Image Reload or PHY Reset	Notes
NVM read	See Section 6.1.3					
LTSSM (back to detect/polling)						
PCIe link data path						
PCI configuration registers (RO)						8
PCI configuration registers (RW)		X				8, 9
Data path and memory space, TimeSync	X	X	X	X		2, 7
MAC	X [6]	X [16]	X [16]	X		
PHY (excluding PLLs)	X [6]	X [16]		X	X	15
Virtual function resources	X	X	X			10
Wake-up (PM) context						3
Wake-up/manageability control/status registers						4, 5
Manageability unit						

Table 4-6. Reset Effects — per Function Resets [continued]

Reset Activation	D3 or Dr	FLR or PFLR	Software Reset	Link Reset or Exit from LAN Disable	PHY Image Reload or PHY Reset	Notes
Strapping pins						
Shadow RAMs in MAC or PHY						

Table 4-7. Reset Effects — Virtual Function Resets

Reset Activation	VFLR	VF Software Reset	Notes
Interrupt registers	X	X	11
Queue disable	X	X	12
VF specific PCIe configuration space	X		13
Data path			
Statistics registers			14

Notes for Table 4-5 **through** Table 4-7**:**

1. If AUX_PWR = 0b the wake-up context is reset (*PME_Status* and *PME_En* bits should be 0b at reset if the X550 does not support PME from D3cold).
2. The following register fields do not follow the general rules previously described:
   - ESDP registers — Reset on LAN Power Good only.
   - LED configuration registers — Reset on LAN Power Good and on software reset events.
   - The *Aux Power Detected* bit in the PCIe Device Status register is reset on LAN Power Good and PCIe reset only.
   - FLA — Reset on LAN Power Good only.
   - RAH/RAL[n, where n>0], MTA[n], VFTA[n], WUPM[n], FFMT[n], FFVT[n], TDBAH/TDBAL, and RDBAH/RDVAL registers have no default value. If the functions associated with these registers are enabled they must be programmed by software. Once programmed, their value is preserved through all resets as long as power is applied.
   - Statistic registers (physical function)
3. The wake-up context is defined in the PCI Bus Power Management Interface specification (sticky bits). It includes:
   - *PME_En* bit of the Power Management Control/Status Register (PMCSR).
   - *PME_Status* bit of the PMCSR.
   - *Aux_En* bit in the PCIe registers.
   - The device requester ID (since it is required for PM_PME TLP).
   - The shadow copies of these bits in the Wake-Up Control (WUC) register are treated identically.
4. Refers to bits in the WUC register that are not part of the wake-up context (the *PME_En* and *PME_Status* bits). The WUFC register is not part of the wake-up context and is reset as part of the data path.

5. The Wake-Up Status (WUS) registers include the following:
   - WUS register.
   - Wake-Up Packet Length (WUPL) register.
   - Wake-Up Packet Memory (WUPM) register.
6. The MAC cluster and the PHY are reset by the appropriate event only if the manageability unit is disabled and the host is in a low-power state with WoL disabled. WoL disabled means either AUX_PWR pin is cleared, or *APM Enable* bit in *NVM Control Word 3* is disabled, or ACPI is disabled (all wake-up filters are disabled or *PME_EN* bit is disabled in PMCSR register).
7. The contents of the following memories are cleared to support the requirements of PCIe FLR:
   - The Tx packet buffers.
   - The Rx packet buffers.
8. Sticky bits and hardware init bits (indicated as HwInit) in the PCI Configuration registers are cleared only by LAN Power Good reset.
9. The following register fields are not affected by FLR or PFLR:
   - *Max_Payload_Size* in the Device Control register
   - Active State Power Management (ASPM) Control in the Link Control register
   - Common Clock Configuration in the Link Control register
   - Link Equalization Request in the Link Status 2 register.
   - Read Completion Boundary (RCB) in the Link Control register
   - Extended Synch in the Link Control register
   - Hardware Autonomous Speed Disable in the Link Control 2 register
   - Link Equalization Control registers in the Secondary PCI Express Extended Capability structure
10. These registers include:
    - VFEICS
    - VFEIMS
    - VFEIAC
    - VFEIAM
    - VFEITR 0-2
    - VFIVAR0
    - VFIVAR_MISC
    - VFPBACL
    - VFMailbox
11. These registers include:
    — VFEICS
    - VFEIMS
    - VFEIMC
    - VFEIAC

- VFEIAM
- VFEICR
- VFEITR 0-2
- VFIVAR0
- VFIVAR_MISC
- VFPBACL
- VFMailbox
- RSCINT

12. These registers include specific VF bits in the FVRE and FVTE registers are cleared as well.
13. These registers include:
    - MSI/MSI-X enable bits
    - BME
    - Error indications
14. Rx and Tx counters might miss proper counting due to VFLR indicating more packets than those ones actually transferred. It could happen if VFLR happened after counting occurred but before Tx or Rx completed.
15. PHY reset events that exclude PLLs reset the following blocks:
    - PMA
    - PCS
    - Autoneg
    - MCP
    - PHY NVM I/F
    - Global, excluding PLLs.The PHY Image Reload command should be effective even if the PHY embedded micro controller is stuck by a faulty PHY image.
16. Concerned functionalities do not reset if the Manageability or WoL are enabled.

Note: Unless specified otherwise the X550's on-die memories are reset together with the functional block(s) they belong to.

4.3 Queue Disable

See Section 4.6.7.1.1 for details on disabling and enabling an Rx queue.

See Section 4.6.8.1.1 for details on disabling and enabling a Tx queue.

4.4 Function Disable

4.4.1 General

For a LAN on Motherboard (LOM) design, it might be desirable for the system to provide BIOS-setup capability for selectively enabling or disabling LAN functions. It enables end users more control over system resource-management and avoids conflicts with add-in NIC solutions. The X550 provides support for selectively enabling or disabling one or both LAN device(s) in the system.

4.4.2 Overview

Device presence (or non-presence) must be established early during BIOS execution, to ensure that BIOS resource-allocation (of interrupts, of memory or IO regions) is done according to devices that are present only. This is frequently accomplished using a BIOS Configuration Values Driven on Reset (CVDR) mechanism. The X550's. LAN-disable mechanism is implemented to be compatible with such a solution.

The X550 provides three mechanisms to disable LAN ports and/or PCIe functions:

- The LANx_DIS_N pins (one pin per LAN port) are sampled on reset to determine the LAN enablement.
- One of the LAN ports can be disabled using a NVM configuration.
- Both SDP1 pins are sampled on reset to determine (electrical) disablement of both PCIe functions. If the MC is present, LAN ports are still available for manageability purposes.

Disabling a LAN port affects the PCI function it resides on. When function 0 is disabled (either LAN0 or LAN1), it does not disappear from the PCIe configuration space. Rather, the function presents itself as a dummy function. The device ID and class code of this function changes to other values (dummy function device ID 0x10A6, class code 0xFF0000). In addition, the function does not require any I/O space, and does not require an interrupt line. It requires a minimal memory space (4K) that is not mapped to internal registers.

Mapping between function and LAN ports is listed in Table 4-8.

Table 4-8. PCI Functions Mapping

PCI Function #	LAN Function Select	Function 0	Function 1
Both LAN functions are enabled.	0	LAN 0	LAN 1
	1	LAN 1	LAN 0
LAN 0 is disabled.	0	Dummy	LAN1
	1	LAN 1	Disable
LAN 1 is disabled.	0	LAN 0	Disable
	1	Dummy	LAN 0
Both LAN functions are disabled.	Both PCI functions are disabled. Device is in low-power mode.		

The following rules apply to function disable:

- When function 0 is disabled, it is converted into a dummy PCI function. Function 1 is not affected.
- When function 1 is disabled, it disappears from the PCI configuration space.

- The disabled LAN port is still available for manageability purposes if disabled through the NVM or SDP1 mechanism. In this case, and if the *LPLU* bit is set, the PHY attempts to create a link at 100 Mb/s. The disabled LAN port is not available for manageability purposes if disabled through the LANx_DIS_N pin mechanism. Section 11.2.2.2 and Section 11.5.3 describe the NC-SI channels and SMBus ports, respectively, exposed in each mode.
- Dummy function mode should not be used in PCI IOV mode (since PF0 is required to support certain functionality).

The following NVM bits control function disable:

- One PCI function can be enabled or disabled according to the *LAN_PCI_DISABLE* field in NVM (reflected in DEV_FUNC_EN.*LAN_PCI_DISABLE* bit).
- The *LAN Disable Select* field in NVM (reflected in DEV_FUNC_EN.*LAN_DISABLE_SELECT* bit) indicates which function is disabled.
- The *LAN_FUNCTION_SEL* field in NVM (reflected in FACTPS.*LAN_FUNCTION_SEL* bit) defines the correspondence between LAN port and PCI function.
- The *SDP_FUNC_OFF_EN* field in NVM (reflected in DEV_FUNC_EN.*SDP_FUNC_OFF_EN* bit) enables the function disable mechanism made via SDP1 pins.

When a particular LAN is fully disabled, all internal clocks to that LAN are disabled, the device is held in reset, and the internal PHY for that LAN is powered down. In both modes, the device does not respond to PCI configuration cycles. Effectively, the LAN device becomes invisible to the system from both a configuration and power consumption standpoint.

4.4.3 Control Options

The functions have a separate enabling mechanism. Any function that is not enabled does not function and does not expose its PCI configuration registers.

LAN0 or LAN1 can be disabled in the NVM by setting the DEV_FUNC_EN.*LAN_PCI_DISABLE* bit. The *LAN_DISABLE_SELECT* in the same register in the NVM selects which LAN is disabled. Furthermore, if the LAN port at function 0 is disabled, the disabled function is filled by a dummy function.

Note: Mapping LAN0 and LAN1 to PCI function 0 and PCI function 1 is controlled by the FACTPS.*LAN_FUNCTION_SEL* field.

LAN0 and LAN1 can be disabled on the board level by driving the LAN0_Dis_N or LAN1_Dis_N pins respectively to low. These I/O pins have weak internal pull up resistors so leaving them unconnected or driving them to high enable the respective LAN port. These pins are strapping options, sampled at LAN Power Good, PCIe reset or in-band PCIe reset.

PCIe Functions 0 and 1 can be disabled at the board level by driving SDP0_1 or SDP1_1 pins respectively high. These I/O pins have weak internal pull-down resistors so leaving them unconnected or driving them to low enables the PCIe functions. These pins are strapping options, sampled at PE_RST_N de-assertion. This feature is enabled/disabled via the DEV_FUNC_EN.*SDP_FUNC_OFF_EN* bit (set from *NVM Control word 2* — Section 6.2.2.2).

4.4.4 Event Flow for Enable/Disable Functions

This section describes the driving levels and event sequence for device functionality.

Following a power on reset / LAN Power Good/ PCIe reset/ in-band reset the LANx_DIS_N signals should be driven high (or left open) for normal operation. If any of the LAN functions are not required statically, its associated disable strapping pin can be tied statically to low.

Following a PCIe reset, the SDP1 pins should be driven low (or left open) for normal operation. If both PCIe functions are not required statically, the SDP1 strapping pins can be tied statically to high.

4.4.4.1 BIOS Disable of the LAN Function at Boot Time by Using the Strapping Option

Assume that following a power-up sequence LANx_DIS_N signals are driven high.

1. PCIe is established following the PCIe reset.
2. BIOS recognizes that a LAN function in the X550 should be disabled.
3. The BIOS drives the LANx_DIS_N signal to a low level.
4. BIOS issues a PCIe reset or an in-band PCIe reset.
5. As a result, the X550 samples the LANx_DIS_N signals and disables the LAN function and issues an internal reset to this function.
6. The BIOS might start with the device enumeration procedure (the disabled LAN function is invisible; changed to dummy function).
7. Proceed with normal operation.
8. Re-enable could be done by driving the LANx_DIS_N signal high and then requesting the end user to issue a warm boot to initialize new bus enumeration.

4.4.4.2 BIOS Disable of the PCIe Functions at Boot Time by Using the Strapping Option

Assume that following a power-up sequence SDP1 signals are driven low and/or the *SDP_FUNC_OFF_EN* bit is cleared.

1. PCIe is established following the PCIe reset.
2. The BIOS recognizes that both PCIe functions in the X550 should be disabled.
3. The BIOS modifies the *DEV_FUNC_EN.SDP_FUNC_OFF_EN* NVM bit and it might eventually issue an in-band PCIe reset that causes the X550 to auto-load the PCIe general configuration from the NVM.
4. The BIOS drives the two SDP1 signals to a high level.
5. The BIOS issues a PCIe reset (via PE_RST_N).
6. The system reboots.
7. PE_RST_N might toggle a couple of times during POST, before its last de-assertion.
8. As a result, the X550 samples the SDP1 signals and disables the two PCIe functions.
9. The BIOS might start with the device enumeration procedure (the disabled PCIe function is invisible and PCIe lanes are electrically off).

4.4.4.3 Multi-Function Advertisement

If one of the LAN devices is disabled and function 0 is the only active function, the X550 is no longer a multi-function device. The X550 normally reports 0x80 in the *PCI Configuration Header* field (*Header Type*), indicating multi-function capability. However, if a LAN is disabled and only function 0 is active, the X550 reports 0x0 in this field to signify single-function capability.

4.4.4.4 Interrupt Use

When both LAN devices are enabled, the X550 uses the PCI legacy interrupts of both ports for interrupt reporting. The NVM configuration controls the *Interrupt Pin* field of the PCI configuration header to be advertised for each LAN device to comply with PCI specification requirements.

However, if either LAN device is disabled, the legacy PCI interrupt of port A must be used for the remaining LAN device, therefore the NVM configuration must be set accordingly. Under these circumstances, the *Interrupt Pin* field of the PCI header always reports a value of 0x1, indicating INTA# pin usage, which means legacy PCI interrupt of port A is used.

4.4.4.5 Power Reporting

When both LAN devices are enabled, the PCI Power Management register block has the capability of reporting a common power value. The common power value is reflected in the *Data* field of the PCI Power Management registers. The value reported as common power is specified via an NVM field, and is reflected in the *Data* field each time the *Data_Select* field has a value of 0x8 (0x8 = common power value select).

When only one LAN port is enabled and the X550 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.

4.5 Device Disable

4.5.1 Overview

To disable the device, both LANx_DIS_N signals should be tied statically to low. When sampled at a power-on reset, PCIe reset, or in-band reset, the X550 is disabled. It is held in reset and power-down mode (some clocks are still running), and digital I/O pins are at High-Z if DEV_FUNC_EN.*DEV_OFF_EN* bit was set in *NVM Control Word 1*. As an example, digital I/O pins are in an electrical off state where pull-up/pull-down resistors are at their defined values. The manageability interface is also disabled in this state.

4.5.2 BIOS Disable of the Device at Boot Time by Using the Strapping Option

Assume that following a power-up sequence LANx_DIS_N signals are driven high.

1. PCIe is established following the PCIe reset.
2. BIOS recognizes that the X550 should be disabled.
3. The BIOS drives the LANx_DIS_N signals to the low level.
4. BIOS issues a PCIe reset or an in-band PCIe reset.
5. As a result, the X550 samples the LANx_DIS_N signals and disables the LAN ports and the PCIe connection.
6. Re-enable can be done by driving high at least one of the LANx_DIS_N signals and then issuing a PCIe reset to restart the device.

4.6 Software Initialization and Diagnostics

4.6.1 Introduction

This section discusses general software notes for the X550, especially initialization steps. These include:

- General hardware power-up state
- Basic device configuration
- Interrupts initialization
- Initialization of transmit
- Receive initialization
- Link configuration
- Software reset capability
- Statistics
- Diagnostic hints
- FCoE initialization
- Virtualization support initialization
- DCB configuration
- Security (IPsec) support initialization

4.6.2 Power-Up State

When the X550 powers up, it automatically reads the NVM. The NVM contains sufficient information to bring the link up and configure the X550 for manageability and/or APM wake-up. However, software initialization is required for normal operation.

4.6.3 Initialization Sequence

The following sequence of commands is typically issued to the device by the software device driver to initialize the X550 to normal operation. The major initialization steps are:

1. Disable interrupts.
2. Issue a global reset and perform general configuration — see Section 4.6.3.2.
3. Wait for the NVM auto-read completion.
4. Wait for manageability configuration done indication (EEMNGCTL.*CFG_DONE0/1).*
5. Wait until the DMA initialization completes (RDRXCTL.*DMAIDONE*).
6. Setup the PHY and the link — see Section 3.7.3.2 and Section 3.7.3.3.
7. Initialize all statistical counters — see Section 4.6.5.
8. Initialize receive — see Section 4.6.7.
9. Initialize transmit — see Section 4.6.8.
10. Initialize FCoE — see Section 4.6.9
11. Initialize Virtualization support — see Section 4.6.10
12. Configure DCB — see Section 4.6.11
13. Configure Security (IPsec) — see Section 4.6.12.
14. Enable interrupts — see Section 4.6.3.1.

4.6.3.1 Interrupts During Initialization

Most drivers disable interrupts during initialization to prevent re-entrance. Interrupts are disabled by writing to the EIMC register. Note that the interrupts need to also be disabled after issuing a global reset, so a typical driver initialization flow is:

1. Disable interrupts.
2. Issue a global reset.
3. Disable interrupts (again).

After initialization completes, a typical driver enables the desired interrupts by writing to the EIMS register.

4.6.3.2 Global Reset and General Configuration

Global reset = software reset + link reset.

Device initialization typically starts with a software reset that puts the device into a known state and enables the device driver to continue the initialization sequence. Following a global reset the software driver should poll the *CTRL.RST* until it is cleared and then wait at least 10 ms to enable a smooth initialization flow.

To enable flow control, program the FCTTV, FCRTL, FCRTH, FCRTV and FCCFG registers. If flow control is not enabled, these registers should be written with 0x0. If Tx flow control is enabled, Tx CRC by hardware should be enabled as well (HLREG0.*TXCRCEN* = 1b). Refer to Section 3.7.4.3.2 through Section 3.7.4.3.5 for recommended settings of the Rx packet buffer sizes and flow control thresholds.

Note: FCRTH[n].*RTH* fields must be set by default regardless if flow control is enabled or not. FCRTH[n].*FCEN* should be set to 0b if flow control is not enabled as all the other registers previously indicated.

The link inter-connect configuration according to the electrical specification of the relevant electrical interface should be set prior to the link setup. This configuration is done through the PHY image section of the NVM by applying the appropriate settings to the link interconnect block.

4.6.4 100 Mb/s, 1 GbE, and 10 GbE Link Initialization

Refer to Section 3.7.3.3 and Section 3.7.3.2 for the initialization and link setup steps. The device driver uses the MDIO register to initialize the PHY and setup the link. Section 3.7.3 describes the usage of the MDIO register.

4.6.4.1 MAC Settings Automatically Based on Speed Resolved by PHY

FCCFG.*RFCE*	Must be set by software after reading flow control resolution from PHY registers.
FCCFG.*TFCE*	Must be set by software after reading flow control resolution from PHY registers.
MAC Speed	Speed setting is established from the PHY's internal indication to the MAC after the PHY has auto-negotiated a successful link up.
PHY Speed	The speed resolution by the PHY with a link partner according to the setup made in the PHY Auto-negotiation registers.
STATUS.*LINKUP*	Must be set by the PF to reflect a link indication from the LINKS register. This is useful for IOV mode.
LINKS.*LINK_STATUS*	Reflects the PHY internal indication to the MAC.
LINKS.*LINK_SPEED*	Reflects the actual speed setting negotiated by the PHY and indicated to the MAC.

4.6.5 Initialization of Statistics

Statistics registers are hardware-initialized to values as detailed in each particular register's description. The initialization of these registers begins upon transition to a D0 active power state (when internal registers become accessible, as enabled by setting the *Memory Access Enable* field of the PCIe Command register), and is guaranteed to complete within 1 ms of this transition. Note that access to statistics registers prior to this interval might return indeterminate values.

All statistical counters are cleared on read and a typical device driver reads them (thus making them zero) as a part of the initialization sequence.

Queue counters are mapped using the RQSMR registers for Rx queues, and TQSM registers for Tx queues. Refer to the RQSMR register section for RQSMR setup, and to the TQSM register section for TQSM setup. Note that if software requires the queue counters, the RQSMR and TQSM registers must be reprogrammed following a device reset.

4.6.6 Interrupt Initialization

4.6.6.1 Working with Legacy or MSI Interrupts

- Software driver associates between Tx and Rx interrupt causes and the EICR register by setting the IVAR[n] registers
- Program the SRRCTL[n].*RDMTS* per receive queue if software uses the Receive Descriptor Minimum Threshold Interrupt (RDMTI).
- All interrupts should be set to zero — no auto clear in the EIAC register. Following an interrupt software can read the EICR register to check for the interrupt causes.
- Set the auto mask in the EIAM register according to the preferred mode of operation.
- Set the interrupt throttling in the EITR[n] and GPIE according to the preferred mode of operation.
- The software enable the required interrupt causes by setting the EIMS register.

4.6.6.2 Operating with MSI-X

- The operating system/BIOS sets hardware to MSI-X mode and programs the MSI-X table as part of the device enumeration procedure.
- The software driver associates between interrupt causes and MSI-X vectors and the throttling timers EITR[n] by programming the IVAR[n] and IVAR_MISC registers.
- Program the SRRCTL[n].RDMTS (per receive queue) if software uses the receive descriptor minimum threshold interrupt.
- The EIAC[n] registers should be set to auto clear for transmit and receive interrupt causes (for best performance). The EIAC bits that control the other and TCP timer interrupt causes should be set to 0b — no auto clear.
- Set auto mask in the EIAM, EIAM[n] registers according to the preferred mode of operation.
- Set the interrupt throttling in the EITR[n] and GPIE registers according to the preferred mode of operation.
- Software clears EICR by writing all ones to clear old interrupt causes.
- Software enables the required interrupt causes by setting the EIMS[n] registers.

4.6.7 Receive Initialization

Initialize the following register tables before receive and transmit is enabled:

- Set CTRL_EXT.*Extended_VLAN* bit if needed
- Receive Address (RAL[n] and RAH[n]) for used addresses and Receive Address High — RAH[n].*VAL*=0b for unused addresses
- Unicast Table Array — PFUTA[n]
- VLAN Filter Table Array — VFTA[n]
- VLAN Pool Filter — PFVLVF[n]
- MAC Pool Select Array — MPSAR[n]

- VLAN Pool Filter Bitmap — PFVLVFB[n].

Program the Receive Address register(s) (RAL[n], RAH[n]) per the station address. This can come from the NVM or from any other means (for example, it could be stored anywhere in the NVM or even in the platform PROM for a LOM design).

Set up the Multicast Table Array — MTA registers. Assuming the entire table was zeroed by the last reset, only the desired multicast addresses should be permitted (by writing 0x1 to the corresponding bit location). Set the MCSTCTRL.*MFE* bit if multicast filtering is required.

Set up the VLAN Filter Table Array — VFTA if VLAN support is required. Assuming the entire table was zeroed by the last reset, only the desired VLAN addresses should be permitted (by writing 0x1 to the corresponding bit location). Set the VLNCTRL.*VF*E bit if VLAN filtering is required.

Initialize the flexible filters 0…7 — Flexible Host Filter Table (FHFT_FILTER) registers.

After all memories in the filter units previously indicated are initialized, enable ECC reporting by setting the RXFECCERR0.*ECCFLT_EN* bit.

Program the different Rx filters and Rx offloads via registers FCTRL, VLNCTRL, MCSTCTRL, RXCSUM, RQTC, RFCTL, MPSAR, RSSRK, RETA, SDPQF, FTQF, SYNQF, ETQF, ETQS, RDRXCTL, RSCDBU.

Note: As NFS detection is not supported, the RFCTL.*NFSW_DIS* and RFCTL.*NFSR_DIS* bits should be set to 1b.

Program RXPBSIZE, MRQC, PFQDE, RTRUP2TC, MFLCN.*RPFCE,* MFLCN.*RPFCM,* and MFLCN.*RFCE* according to the DCB and virtualization modes. Refer to Section 4.6.11.3.

Enable receive jumbo frames by setting HLREG0.*JUMBOEN* in the following case:

1. Jumbo packets are expected. Set MAXFRS.*MFS* to the expected maximum packet size.

Enable receive coalescing if required as described in Section 4.6.7.2.

4.6.7.1 Receive Queues Enable

The following should be done for each receive queue:

1. Allocate a region of memory for the receive descriptor list.
2. Receive buffers of appropriate size should be allocated and pointers to these buffers should be stored in the descriptor ring.
3. Program the descriptor base address with the address of the region (registers RDBAH, RDBAL).
4. Set the length register to the size of the descriptor ring (register RDLEN).
5. Program SRRCTL associated with this queue according to the size of the buffers and the required header control.
6. Set RXDCTL[n].*RLPML* field enabled by the RXDCTL[n].*RLPML_EN* limiting the maximum Rx packet size. This setting is optional enabling the software to use smaller buffers than the size defined by the SRRCTL[n].*BSIZEPACKET*. Software may not use smaller buffers than defined by the SRRCTL[n] on Rx queues that enables RSC.
7. If header split is required for this queue, program the appropriate PSRTYPE for the appropriate headers.
8. Program RSC mode for the queue via RSCCTL register.
9. Program RXDCTL with appropriate values including the queue *Enable* bit. Note that packets directed to a disabled queue are dropped.

10. Poll the RXDCTL register until the *Enable* bit is set. The tail should not be bumped before this bit was read as 1b.
11. Bump the tail pointer (*RDT*) to enable descriptors fetching by setting it to the ring length minus one.

Enable the receive path by setting RXCTRL.*RXEN*. This should be done only after all other settings are done. If software uses the receive descriptor minimum threshold interrupt, that value should be set.

4.6.7.1.1 Dynamic Enabling and Disabling of Receive Queues

Receive queues can be enabled or disabled dynamically using the following procedure.

Enabling:

Follow the per-queue initialization described in the previous section.

Disabling:

1. Disable the routing of packets to this queue by re-configuring the Rx filters.
2. If RSC is enabled on the specific queue and VLAN strip is enabled as well, wait two ITR expiration times (ensure all open RSCs completed).
3. All FCoE DDP flows related to this queue must be invalidated by the software device driver prior to disabling the queue.
4. Disable the queue by clearing the RXDCTL.*ENABLE* bit. The X550 stops fetching and writing back descriptors from this queue. Any further packet that is directed to this queue is dropped. If a packet is being processed, the X550 completes the current buffer write. If the packet spreads over more than one data buffer, all subsequent buffers are not written.
5. The X550 clears the RXDCTL.*ENABLE* bit only after all pending memory accesses to the descriptor ring are done. The driver should poll this bit before releasing the memory allocated to this queue.
6. Once the RXDCTL.*ENABLE* bit is cleared the driver should wait an additional amount of time (~100 µs) before releasing the memory allocated to this queue.

The Rx path can be disabled only after all the receive queues are disabled.

Note: As there could be additional packets in the receive packet buffer targeted to the disabled queue and the arbitration could be such that it would take a long time to drain these packets, if software re-enables a queue before all packets to that queue were drained, the enabled queue could potentially get packets directed to the old configuration of the queue. For example, VM goes down and a different VM gets the queue. The software device driver should delay the re-enablement of the queue until it is guaranteed there are no more packets directed to this queue in the packet buffer.

4.6.7.2 RSC Enablement

RSC enablement as well as RSC parameter settings are assumed as static. It should be enabled prior receiving and can be disabled only after the relevant Rx queue(s) are disabled.

4.6.7.2.1 RSC Global Setting

- Enable global CRC stripping via the *HLREG0* register (hardware default setting).
- Software device driver should set the RDRXCTL.*RSCACKC* bit that forces RSC completion on any change of the *ACK* bit in the Rx packet relative to the RSC context.

- The SRRCTL[n].*BSIZEHEADER* (header buffer size) bit must be larger than the packet header (even if header split is not enabled). A minimum size of 128 bytes for the header buffer addresses this requirement.

4.6.7.2.2 RSC per Queue Setting

- Enable RSC and configure the maximum allowed descriptors per RSC by setting the *MAXDESC* and *RSCEN* fields in the RSCCTL[n] register.
- Use a non-legacy descriptor type by setting the SRRCTL[n].*DESCTYPE* bit to non-zero values.
- TCP header recognition: the *PSR_TYPE4* bit in the PSRTYPE[n] registers should be set.
- The SRRCTL[n].*BSIZEPACKET* (packet buffer size) must be 2 KB at minimum.
- Interrupt setting:
  - Interrupt moderation must be enabled by setting the EITR[n].*ITR_INTERVAL* bit to a value greater than zero. The *ITR_INTERVAL* bit must be larger than the *RSC Delay* field described later. Note that if the *CNT_WDIS* bit is cleared (write enable), the *ITR_COUNTER* bit should be set to 0b.
  - The *RSC_DELAY* field in the GPIE register should be set to the expected system latency descriptor write-back cycles. 4 to 8 μs should be sufficient in most cases. If software sees cases where RSC did not complete as expected (following EITR interrupt assertion), the *RSC_DELAY* field might need to be increased.
  - Map the relevant Rx queues to an interrupt by setting the relevant IVAR registers.

4.6.7.3 Flow Director Initialization

Flow Director initialization is described in Section 7.1.3.6.11.

4.6.8 Transmit Initialization

- Program the HLREG0 register according to the MAC behavior needed.
- Program the TCP segmentation parameters via registers DMATXCTL (while keeping the *TE* bit cleared), DTXTCPFLGL, DTXTCPFLGH, and TPH parameters via TPH_TXCTRL.
- Refer to the IPG description in Section 3.7.5 for more details.
- Set RTTDCS.*ARBDIS* to 1.
- Program the DTXMXSZRQ, TXPBSIZE, TXPBTHRESH, MTQC, and MNGTXMAP registers according to the DCB and virtualization modes. Refer to Section 4.6.11.3.
- Clear RTTDCS.*ARBDIS* to 0.
- Legacy drivers that uses queue zero and assumes it is enabled by default, should at this stage, set queue zero parameters as described below.
- Enable the transmit path by setting the DMATXCTL.*TE* bit.

4.6.8.1 Transmit Queues Enable

The following steps should be done once for each transmit queue:

1. Allocate a region of memory for the transmit descriptor list.
2. Program the descriptor base address with the address of the region (TDBAL and TDBAH).
3. Set the length register to the size of the descriptor ring (TDLEN).
4. If needed, set TDWBAL/TWDBAH to enable head write back.
5. Program the TXDCTL register with the desired Tx descriptor write-back policy (see the recommended values in the register description).
6. Enable the queue using the TXDCTL.*ENABLE* bit. Poll the TXDCTL register until the *ENABLE* bit is set.

Note: Queue 0 is enabled by default when the DMATXCTL.*TE* bit is set. If transmit is already enabled (DMATXCTL.*TE* bit is set), this queue should be disabled (clear TXDCTL.*ENABLE* bit) before changing the basic queue parameters (TDBAL, TDBAH, TDLEN, TDWBAL/TWDBAH).

Note: The tail register of the queue (TDT) should not be bumped until the queue is enabled.

4.6.8.1.1 Dynamic Enabling and Disabling of Transmit Queues

Transmit queues can be enabled or disabled dynamically given the following procedure is followed.

Enabling:

Follow the per-queue initialization described in the previous section.

Disabling:

1. Stop storing packets for transmission in this queue.
2. The completion of the last transmit descriptor must be visible to software to guarantee that packets are not lost in step 5 (Section 4.6.8). Therefore, its *RS* bit must be set or *WTHRESH* must be greater than zero. If none of the previous conditions are met, software should add a null Tx data descriptor with an active *RS* bit.
3. Wait until the software head of the queue (*TDH*) equals the software tail (*TDT*) indicating the queue is empty.
4. Wait until all descriptors are written back (polling the *DD* bit in ring or polling the *Head_WB* content). It might be required to flush the transmit queue by setting the TXDCTL[n].*SWFLSH* bit if the *RS* bit in the last fetched descriptor is not set or if *WTHRESH* is greater than zero.
5. Disable the queue by clearing TXDCTL.*ENABLE*.
6. Any packets waiting for transmission in the packet buffer would still be sent at a later time.

The transmit path can be disabled only after all transmit queues are disabled.

4.6.9 FCoE Initialization Flow

Ordering between the following steps is not critical as long as it is done before transmit and receive starts.

1. The FCoE DDP context table should be initialized clearing the FCBUFF.*VALID* bit and the FCFLT.*VALID* bit of all contexts.
2. EType Queue Filter — ETQF[n]: Select a filter by setting the *FCOE* bit. The *ETYPE* field should be set to 0x8906 (FCoE Ethernet Type).
3. EType Queue Select — ETQS[n]: Each ETQF filter is associated to a queue select register. The ETQS registers can be used to direct the FCoE traffic to specific receive queues. Up to one queue per Traffic Class (TC) as programmed in the ETQF.
4. Multiple receive queues can be enabled by setting the FCRECTL.*ENA* bit and programming the FCRETA[n] registers.
5. Set the RDRXCTL.*FCOE_WRFIX* bit that forces a DDP write exchange context closure after receiving the last packet in a sequence with an active *Sequence Initiative* bit in the *F_CTL* field.
6. Follow the rules described in Section 7.10.2.1 and Section 7.10.3.1 for Tx and Rx cross-functionality requirements. These sections include requirements on Ethernet CRC and padding handling, legacy Rx buffers, etc.

4.6.10 Virtualization Initialization Flow

4.6.10.1 VMDq Mode

4.6.10.1.1 Global Filtering and Offload Capabilities

- Select one of the VMDQ pooling and queuing methods
  - Pool Selection can be based on MAC/VLAN, and optionally E-tag. The filtering mode is defined using the PFVTCTL.*POOLING_MODE* field.
- Queue in poll can be based either on DCB or RSS. Potential values for MRQC.*MRQE* to set VMDq modes are 1000b to 1111b.
- DCB should be initiated as described in Section 4.6.11. In RSS mode, the RSS key (VFRSSRK) and redirection table (VFRETA) of the pool should be programmed. The usage of DCB can be controlled per pool.
- Configure the PFVTCTL register to define the default pool.
- Enable replication via the PFVTCTL.*RPL_EN* bit.
- If needed, enable padding of small packets via the HLREG0.*TXPADEN* bit.
- The MPSAR registers are used to associate Ethernet MAC Addresses to pools. Using the MPSAR registers, software must reprogram RAL[0] and RAH[0] by their values (software could read these registers and then write them back with the same content).

4.6.10.1.2 Mirroring Rules

For each mirroring rule to be activated:

- Set the type of traffic to be mirrored in the PFMRCTL[n] register. Only one type of traffic can be selected in each rule.
- Set the mirror pool by setting the PFMRCTL[n].*MP* bit.
- For pool mirroring, set the PFMRVM[n] register with the pools to be mirrored.
- For VLAN mirroring, set the PFMRVLAN[n] register with the indexes from the PFVLVF registers of the VLANs to be mirrored.

4.6.10.1.3 Security Features

For each pool, the driver might activate the MAC, VLAN and EtherType anti-spoof features via the relevant bit in the PFVFSPOOF.*MACAS,* PFVFSPOOF.*VLANAS,* PFVFSPOOF.*ETHERTYPEAS, and* PFVFSPOOF.*ETHERTYPELB* fields, respectively.

In addition, stripping and hiding of VLAN and external tag (E-tag) may also be requested via the PFQDE.*HIDE_VLAN* and PFQDE.*STRIP_TAG* flags, respectively.

4.6.10.1.4 Per-Pool Settings

As soon as a pool of queues is associated to a VM, software should set the following parameters:

- Associate the unicast Ethernet MAC Address of the VM by enabling the pool in the MPSAR registers.
- If all the Ethernet MAC Addresses are used, the Unicast Hash Table (PFUTA) can be used. Pools servicing VMs whose address is in the hash table should be declared as so by setting the PFVML2FLT.*ROPE* bit. Packets received according to this method didn't pass perfect filtering and are indicated as such.
- Enable the pool in all RAH/RAL registers representing the multicast Ethernet MAC Addresses this VM belongs to.
- If all the Ethernet MAC Addresses are used, the Multicast Hash Table (MTA) can be used. Pools servicing VMs using multicast addresses in the hash table should be declared as so by setting the PFVML2FLT.*ROMPE* bit. Packets received according to this method did not pass perfect filtering and are indicated as such.
- Define whether this VM should get all multicast/broadcast packets in the same VLAN via PFVML2FLT.*MPE* and PFVML2FLT.*BAM*, and whether it should accept un-tagged packets via PFVML2FLT.*AUPE*.
- Enable the pool in each of the PFVLVF and PFVLVFB registers this VM belongs to.
- A VM might be set to receive it's own traffic in case the source and the destination are in the same pool via the PFVMTXSW.*LLE* bit.
- Whether VLAN header and CRC should be stripped from the packet. Note that even if the CRC is kept, it might not match the actual content of the forwarded packet, because of other offloading applications such as VLAN strip.
  - A striped VLAN may be also hidden from the VM.
- Set which header split is required via the PSRTYPE register.

- In RSS mode, define if the pool uses RSS via the proper MRQC.*MRQE* mode.
  - Enable the Pool in the PFVFRE register to allow Rx Filtering
  - To Enable Multiple Tx queues, Set the MTQC as described in Section 7.2.1.2.1
  - Enable the Pool in the PFVFTE register to allow Tx Filtering
- Enable Rx and Tx queues as described in Section 4.6.7 and Section 4.6.8.
- For each Rx queue a drop/no drop flag can be set in SRRCTL.*DROP_EN* and via the PFQDE register, controlling the behavior in cases no receive buffers are available in the queue to receive packets. The usual behavior is to allow a drop to avoid head of line blocking. Setting the PFQDE (per queue) is done by using the *QUEUE_INDEX* field in the PFQDE register.

4.6.10.2 IOV Initialization

4.6.10.2.1 PF Driver Initialization

The PF driver is responsible for the link setup and handling of all the filtering and offload capabilities for all the VFs as described in Section 4.6.10.1.1 and the security features as described in Section 4.6.10.1.3. It should also set the bandwidth allocation per transmit queue for each VF as described in Section 4.6.10.

Note: Link setup might include the authentication process (802.1X or other) and setup of the DCB parameters.

After all the common parameters are set, the PF driver should set all the VFMailbox[n].*RSTD* bits by setting the CTRL_EXT.*PFRSTD* bit.

PF enables VF traffic via the PFVFTE and PFVFRE registers after all VF parameters are set as defined in Section 4.6.10.1.4.

Note: If the operating system changes the NumVF setting in the PCIe SR-IOV Num VFs register after the device was active, it is required to initiate a PF software reset following this change.

4.6.10.2.1.1 VF Specific Reset Coordination

After the PF driver receives an indication of a VF FLR via the PFVFLREC register, it should enable the receive and transmit for the VF only once the device is programmed with the right parameters as defined in Section 4.6.10.1.4. The receive filtering is enabled using the PFVFRE register and the transmit filtering is enabled via the PFVFTE register.

Note: The filtering and offloads setup might be based on a central IT settings or on requests from the VF drivers.

4.6.10.2.2 VF Driver Initialization

At initialization, after the PF indicated that the global initialization was done via the VFMailbox.*RSTD* bit, the VF driver should communicate with the PF, either via the mailbox or via other software mechanisms to assure that the right parameters of the VF are programmed as described in Section 4.6.10.1.4. The PF driver might then send an acknowledge message with the actual setup done according to the VF request and the IT policy.

The VF driver should then setup the interrupts and the queues as described in Section 4.6.6, Section 4.6.7, and Section 4.6.8.

4.6.10.2.3 Full Reset Coordination

A mechanism is provided to synchronize reset procedures between the PF and the VFs. It is provided specifically for PF software reset but can be used in other reset cases as described later in this section.

The procedure is as follows:

One of the following reset cases takes place:

- LAN Power Good
- PCIe reset (PE_RST_N and in-band)
- D3hot --> D0
- FLR
- Software reset by the PF

The X550 sets the *RSTI* bits in all VFMailbox registers. Once the reset completes, each VF might read its VFMailbox register to identify a reset in progress.

Once the PF completes configuring the device, it clears the CTRL_EXT.PFRSTD bit. As a result, the X550 clears the *RSTI* bits in all VFMailbox registers and sets the *Reset Done* (*RSTD*) bits are set in all VFMailbox registers.

Until a RSTD condition is detected, the VFs should only access the VFMailbox register and should not attempt to activate the interrupt mechanism or the transmit and receive process.

4.6.11 DCB Configuration

After power-up or device reset, DCB and any type of FC are disabled by default, and a unique TC and packet buffer (such as PB0) is used. In this mode, the host might exchange information via DCX protocol to determine the number of TCs to be configured. Before setting the device to multiple TCs, it should be reset (software reset).

The registers concerned with setting the number of TCs are: RXPBSIZE[0-7], TXPBSIZE[0-7], TXPBTHRESH, MRQC, MTQC, and RTRUP2TC registers as well as the following bits RTRPCS.*RAC*, RTTDCS.*TDPAC*, RTTDCS.*VMPAC* and RTTPCS.*TPPAC*.

They cannot be modified on the fly, but only after device reset. Packet buffers with a non-null size must be allocated from PB0 and up.

Rate parameters and bandwidth allocation to VMs can be modified on the fly without disturbing traffic flows.

4.6.11.1 CPU Latency Considerations

When the CPU detects an idle period of some length, it enters a low-power sleep state. When traffic arrives from the network, it takes time for the CPU to wake and respond (such as to snoop). During that period, Rx packets are not posted to system memory.

If the entry time-to-sleep state is too short, the CPU might be getting in and out of its sleep state in between packets, therefore impacting latency and throughput. 100 µs is defined as a safe margin for entry time to avoid such effects.

Each time the CPU is in low power, received packets need to be stored (or dropped) in the X550 for the duration of the exit time. Given 64 KB Rx packet buffers per TC in the X550, (i.e. assuming typically 4 TCs per port) PFC does not spread (or the packet is not dropped) provided the CPU exits its low power state within 50 µs.

4.6.11.2 Link Speed Change Procedure

Each time the link status or speed is changed, hardware automatically updates the rates that were loaded by software relatively to the new link speed. This means that if a rate limiter was set by software to 500 Mb/s for a 10 GbE link speed, it is changed by hardware to 50 Mb/s if the link speed has changed to 1 GbE.

Since rates must be considered as absolute rate limitations (expressed in Mb/s, regardless of the link speed), in such cases software is responsible to either clear all the rate-limiters and/or re-load each rate with the correct value relatively to the new link speed. In the previous example, the new rate value to be loaded by software must be multiplied by 10 to maintain the rate limitation to 500 Mb/s.

4.6.11.3 Initial Configuration Flow

Only the following configuration modes are allowed.

4.6.11.3.1 General Case: DCB-On, VT-On

1. Configure packet buffers, queues, and traffic mapping:
   - 8 TCs mode — Packet buffer size and threshold, typically RXPBSIZE[0-7].*SIZE* = 0x30, 0x28, or 0x20, depending on the size of the flow director table. Refer to rules in Section 3.7.4.3.5 for a non-symmetrical sizing.

     TXPBSIZE[0-7].*SIZE* = 0x14

     but non-symmetrical sizing is also allowed (see rules in the TXPBSIZE register)

     TXPBTHRESH.*THRESH[0-7]* = TXPBSIZE[0-7].*SIZE* — Max expected Tx packet length in this TC.
   - 4 TCs mode — Packet buffer size and threshold, typically RXPBSIZE[0-3].*SIZE* = 0x60, 0x50, or 0x40, depending on the size of the flow director table. Refer to rules in Section 3.7.4.3.5 for a non-symmetrical sizing.

     RXPBSIZE[[4-7].*SIZE* = 0x0
     TXPBSIZE[0-3].*SIZE* = 0x28
     TXPBSIZE[4-7].*SIZE* = 0x0

     but non-symmetrical sizing among TCs[0-3] is also allowed (see rules in TXPBSIZE register)

     TXPBTHRESH.*THRESH[0-3]* = TXPBSIZE[0-3].*SIZE* — Maximum expected Tx packet length in this TC.

     TXPBTHRESH.*THRESH[4-7]* = 0x0
   - Multiple Receive and Transmit Queue Control (MRQC and MTQC)
     — Set MRQC.*MRQE* to 1xxxb, with the 3 LS bits set according to the number of VFs, TCs, and RSS mode as listed in the MRQC register section.
     — Set both *DCB_ENA* and *VT_ENA* bits in the MTQC register.

— Set MTQC.*NUM_TC_OR_Q* according to the number of TCs/VFs enabled as described in the MTQC register section.

- Set the PFVTCTL.*VT_ENA* bit (as the MRQC.*VT_ENA bit*) Queue Drop Enable (PFQDE): In SR-IOV the *PFQDE* bit should be set to 1b in the *PFQDE* register for all queues. In VMDq mode, the *PFQDE* bit should be set to 0b for all queues.
- Split Receive Control (SRRCTL[0-127]): *DROP_EN* = 1b — drop policy for all the queues, to avoid crosstalk between VMs
- Rx User Priority (UP)-to-TC (RTRUP2TC)
- Tx UP-to-TC (RTTUP2TC)
- DMA Tx TCP Max Allow Size Requests (DTXMXSZRQ) — set *MAX_BYTE_NUM_REQ* = 0x010 = 4 KB.
- Operate the SEC-TX buffer in DCB mode by setting SECTXMINIFG.*MRKRINSERT* to 0x640 or to 0x1A0 depending if 9.5 KB Jumbo frames are supported or not, respectively.

2. Enable PFC and disable legacy flow control:
   - Enables transmit priority flow control via: FCCFG.*TFCE* = 10b, and for the TC(s) where no drop behavior is required by FCRTH.*FCEN* = 1b and appropriate setting in corresponding FCRTL.*RTL* and FCRTH.*RTH*.

     Note: FCRTL.*RTL* must be set to a non zero value when FCRTH.*FCEN* = 1b.
   - Enables receive priority flow control via MFLCN.*RPFCM* = 1b and MFLCN.*RPFCE* set for the UP(s) where no drop behavior is required.
   - Disable receive legacy flow control via MFLCN.*RFCE* = 0b.
   - Refer to Section 8.2.2.3 for other registers concerned with flow control
3. Configure arbiters, per TC[0-1]:
   - Tx Descriptor Plane T1 Config (RTTDT1C) per queue, via setting RTTDQSEL first
   - Tx Descriptor Plane T2 Config (RTTDT2C[0-7])
   - Tx Packet Plane T2 Config (RTTPT2C[0-7])
   - Rx Packet Plane T4 Config (RTRPT4C[0-7])
4. Enable TC and VM arbitration layers:
   - Tx Descriptor Plane Control and Status (RTTDCS), bits: *TDPAC* = 1b, *VMPAC* = 1, *TDRM* = 1b, *BDPM* = 0b, *BPBFSM* = 0b.
   - Tx Packet Plane Control and Status (RTTPCS): *TPPAC* = 1b, *TPRM* = 1b, *ARBD* = 0x004, *BYPASS_DCB_ARB* = 0
   - Rx Packet Plane Control and Status (RTRPCS): *RAC* = 1b, *RRM* = 1b

4.6.11.3.2 DCB-On, VT-Off

Set the configuration bits as specified in Section 4.6.11.3.1 with the following exceptions:

- Configure packet buffers, queues, and traffic mapping:
  - MRQC and MTQC
    - Set *MRQE* to 0xxxb, with the 3 LS bits set according to the number of TCs, and RSS mode
    - Set the *DCB_ENA* bit and clear the *VT_ENA* bit in the MTQC register.
    - Set MTQC.*NUM_TC_OR_Q* according to the number of TCs enabled
  - Clear the PFVTCTL.*VT_ENA* bit (as the MRQC.*VT_ENA* bit)
- Allow no-drop policy in Rx:
  - PFQDE: The *PFQDE* bit should be set to 0b in the PFQDE register for all queues enabling per queue policy by the SRRCTL[n] setting.
  - Split Receive Control (SRRCTL[0-127]): The *DROP_EN* bit should be set (per receive queue) according to the required drop / no-drop policy of the TC of the queue.
- Disable VM arbitration layer:
  - Clear the RTTDT1C register, per each queue, via setting RTTDQSEL first
  - RTTDCS.*VMPAC* = 0b

4.6.11.3.3 DCB-Off, VT-On

Set the configuration bits as specified in Section 4.6.11.3.1 with the following exceptions:

- Disable multiple packet buffers and allocate all queues to PB0:
  - RXPBSIZE[0].*SIZE* = 0x180, RXPBSIZE[1-7].*SIZE* = 0x0
  - TXPBSIZE[0].*SIZE* = 0xA0, TXPBSIZE[1-7].*SIZE* = 0x0
  - TXPBTHRESH.THRESH[0] = 0xA0 — Max expected Tx packet length in this TC TXPBTHRESH.THRESH[1-7] = 0x0
  - MRQC and MTQC
    - Set *MRQE* to 1xxxb, with the 3 LS bits set according to the number of VFs and RSS mode
    - Clear *DCB_ENA* bit and set the *VT_ENA* bit in the MTQC register.
    - Set MTQC.*NUM_TC_OR_Q* according to the number of VFs enabled
  - Set the PFVTCTL.*VT_ENA* bit (as the MRQC.*VT_ENA* bit) Rx UP-to-TC (RTRUP2TC), UPnMAP=0, n=0,...,7
  - Tx UP-to-TC (RTTUP2TC), UPnMAP=0, n=0,...,7
  - DMA Tx TCP Max Allow Size Requests (DTXMXSZRQ) — set *MAX_BYTE_NUM_REQ* = 0xFFF = 1 MB
  - Operate the SEC-TX buffer in non-DCB mode by setting SECTXMINIFG.*MRKRINSERT* to 0x10
- Disable PFC and enabled legacy flow control:
  - Disable receive priority flow control via: MFLCN.*RPFCM* = 0b and MFLCN.*RPFCE[7:0]* = 0x00
  - Enable transmit legacy flow control via: FCCFG.*TFCE* = 01b

— Enable receive legacy flow control via: MFLCN.*RFCE* = 1b

- Configure VM arbiters only, reset others:
  — Tx Descriptor Plane T1 Config (RTTDT1C) per pool, via setting RTTDQSEL first for the pool index. Clear RTTDT1C for other queues.
  — Clear RTTDT2C[0-7] registers
  — Clear RTTPT2C[0-7] registers
  — Clear RTRPT4C[0-7] registers
- Disable TC arbitrations while enabling the PB Free Space Monitor:
  — Tx Descriptor Plane Control and Status (RTTDCS), bits: *TDPAC* = 0b, *VMPAC* = 1b, *TDRM* = 0b, *BDPM* = 1b, *BPBFSM* = 0b
  — Tx Packet Plane Control and Status (RTTPCS): *TPPAC* = 0b, *TPRM* = 0b, *ARBD* = 0x24, *BYPASS_DCB_ARB* = 1.
  — Rx Packet Plane Control and Status (RTRPCS): *RAC* = 0b, *RRM* = 0b

4.6.11.3.4 DCB-Off, VT-Off

Set the configuration bits as specified in Section 4.6.11.3.1 with the following exceptions:

- Disable multiple packet buffers and allocate all queues and traffic to PB0:
  — RXPBSIZE[0].*SIZE* = 0x180, RXPBSIZE[1-7].*SIZE* = 0x0
  — TXPBSIZE[0].*SIZE* = 0xA0, TXPBSIZE[1-7].*SIZE* = 0x0
  — TXPBTHRESH.*THRESH[0]* = 0xA0 — Max expected Tx packet length in this TC TXPBTHRESH.*THRESH[1-7]* = 0x0
  — MRQC and MTQC
    - Set *MRQE* to 0xxxb, with the 3 LS bits set according to the RSS mode
    - Clear both *DCB_ENA* and *VT_ENA* bits in the MTQC register.
    - Set MTQC.*NUM_TC_OR_Q* to 00b.
  — Clear the PFVTCTL.*VT_ENA* bit (as the MRQC.*VT_ENA* bit) Rx UP-to-TC (RTRUP2TC), UPnMAP=0, n=0,...,7
  — Tx UP-to-TC (RTTUP2TC), UPnMAP=0, n=0,...,7
  — DMA Tx TCP Max Allow Size Requests (DTXMXSZRQ) — set *MAX_BYTE_NUM_REQ* = 0xFFF = 1 MB
  — Operate the SEC-TX buffer in non-DCB mode by setting SECTXMINIFG.*MRKRINSERT* to 0x10
- Allow no-drop policy in Rx:
  — PFQDE: The *PFQDE* bit should be set to 0b in the PFQDE register for all queues enabling per queue policy by the SRRCTL[n] setting.
  — Split Receive Control (SRRCTL[0-127]): The *DROP_EN* bit should be set (per receive queue) according to the required drop / no-drop policy of the TC of the queue.

- Disable PFC and enable legacy flow control:
  - Disable receive priority flow control via: MFLCN.*RPFCM* = 0b and MFLCN.*RPFCE[7:0]* = 0x00
  - Enable receive legacy flow control via: MFLCN.*RFCE* = 1b
  - Enable transmit legacy flow control via: FCCFG.*TFCE* = 01b
- Reset all arbiters:
  - Clear the RTTDT1C register, per each queue, via setting RTTDQSEL first
  - Clear RTTDT2C[0-7] registers
  - Clear RTTPT2C[0-7] registers
  - Clear RTRPT4C[0-7] registers
- Disable TC and VM arbitration layers:
  - Tx Descriptor Plane Control and Status (RTTDCS), bits: *TDPAC* = 0b, *VMPAC* = 0b, *TDRM* = 0b, *BDPM* = 1b, *BPBFSM* = 1b
  - Tx Packet Plane Control and Status (RTTPCS): *TPPAC* = 0b, *TPRM* = 0b, *ARBD* = 0x224
  - Rx Packet Plane Control and Status (RTRPCS): *RAC* = 0b, *RRM* = 0b

4.6.11.4 Configuration Rules

4.6.11.4.1 TC Parameters

Traffic Class (TC):

Per 802.1p priority #7 is the highest priority.

A specific TC can be configured to receive or transmit a specific amount of the total bandwidth available per port.

Bandwidth allocation is defined as a fraction of the total available bandwidth, which can be less than the full Ethernet link bandwidth (if it is bounded by the PCIe bandwidth or by flow control).

Low latency TC should be configured to use the highest priority TC possible (TC 6, 7); the lowest latency is achieved using TC7.

Bandwidth Groups (BWGs):

BWGs are used to represent different traffic types. A traffic type (such as storage, IPC LAN, or manageability) can have more than one TC. For example, one for control traffic and one for the raw data. By grouping these two TCs to a BWG, the end user can allocate bandwidth to the storage traffic so that unused bandwidth by the control could be used by the data and vise versa. This BWG concept supports the converged fabric as each traffic type, that's used to run on a different fabric, can be configured as a BWG and gets its resources as if it was on a different fabric.

1. To configure DCB not to share bandwidth between TCs, each TC should be configured as a separate BWG.
2. There are no limits on the TCs that can be bundled together as a BWG. All TCs can be configured as a single BWG.
3. BWG numbers should be sequential starting from zero until the total number of BWG minus one.
4. BWG numbers do not imply priority; priority is only set according to TCs.

Refill Credits:

Refill credits regulate the bandwidth allocated to BWGs and TCs. The ratio between the credits of the BWG's represents the relative bandwidth percentage allocated to each BWG. The ratio between the credits of the TCs represents the relative bandwidth percentage allocated to each TC within a BWG.

Credits are configured and calculated using 64-byte granularity.

1. In any case, the number of refill credits assigned per TC should be as small as possible but still larger than the maximum frame size used and larger than 1.5 KB. Using a lower refill value causes more refill cycles before a packet can be sent. These extra cycles unnecessarily increase the latency.
2. Refill credits ratio between TCs should be equal to the desired ratio of bandwidth allocation between the different TCs. Applying rule #1, means bandwidth shares are sorted from the smaller to the bigger, and just one maximum sized frame is allocated to the smallest.
3. The ratio between the refill credits of any two TCs should not be greater than 100.
4. Exception to rule #2 — TCs that require low latency should be configured so that they are under subscribed. For example, the credit refill value should provide these TCs somewhat more bandwidth than what they actually need. Low latency TCs should always have credits so they can be next in line for WSP arbitration.

   This exception causes the low latency TC to always have maximum credits (as it starts with maximum credits and on average cycle uses less than the refill credits).

The end point that is sending/receiving packets of 127 bytes eventually get double the bandwidth it was configured to, as all credits calculated by rounding the values down to the next 64 byte-aligned value.

Max Credit Limit:

The maximum credit limit value establishes a limit for the number of credits that a TC or BWG can own at any given time. This value prevents stacking up stale credits that can be added up over a relatively long period of time and then used by TCs all at once, altering fairness and latency.

Max credit limits are configured and calculated using 64-byte granularity.

1. Maximum credit limit should be bigger than the refill credits allocated to the TC.
2. Maximum limit should be set to be as low as possible while still meeting other rules to minimize the latency impact on low latency TCs.
3. If a low latency TC generates a burst that is larger than its maximum credit limit, this TC might experience higher latency since the TC needs to wait for allocation of additional credits because it finished all its credits for this cycle. Therefore, maximum credit limit for a low latency TC must be set bigger than the maximum burst length of traffic expected on that TC — for all VMs at once. If TC7 and TC6 are for low latency traffic, it leads to:

   Max(TC7,6) >= MaxBurst(TC7,6) served with low latency
4. An arbitration cycle can be extended when one or more TCs accumulate credits more than their refill values (up to their maximum credit limit). For such a case a low latency TC should be provided with enough credits to cover for the extended cycle duration. Since the low latency TC operates at maximum credits (see rule #3) its maximum credit limit should meet the following formula.

   {Max(TCx)/SUMi=0..7[Max(TCi)]} >= {BW(TCx)/Full BW}

   The formula applies to both the descriptor arbiter and data arbiter.

5. To ensure bandwidth for a low priority TC (when those are allocated with most of the bandwidth) the maximum credit value of the low priority TC in the data arbiter needs to be high enough to ensure sync between the two arbiters. In the following equation the bandwidth numbers are from the descriptor arbiter while the maximum values are of the data arbiter.

   {Max(TCx)/SUMi=x+1..7[Max(TCi)]} >= {BW(TCx)/Full_PCIE_BW}

   Note that the previous equation is worst case and covers the assumption that all higher TCs have the full maximum to transmit.

Tip: A simplified maximum credits allocation scheme would be to find the minimum number N >= 2 such that rules #3 and #5 are respected, and allocate:

Max(TCi) = N x Refill(TCi), for i=0..7

By maintaining the same ratios between the maximum credits and the bandwidth shares, the bandwidth allocation scheme is made more immune to disturbing events such as receiving priority pause frames with short timer values.

GSP and LSP:

Bit TC.LSP (TC Link Strict Priority): This bit specifies that the configured TC can transmit without any restriction of credits. This effectively means that the TC can take up the entire link bandwidth, unless preempted by higher priority traffic. The Tx queues associated with a LSP TC must be set as strict low latency in the TXLLQ[n] registers.

Bit TC.GSP (TC Strict Priority within Group): This bit defines whether strict priority is enabled or disabled for this TC within its BWG. If bit TC.GSP is set to 1b, the TC is scheduled for transmission using strict priority. It does not check for availability of credits in the TC. It does; however, check whether the BWG of this TC has credits. For example, the amount of traffic generated from this TC is still limited by the bandwidth allocated for the BWG.

1. TCs with the *LSP* bit set should be the first to be considered by the scheduler. This implies that the *LSP* bit should be configured to the highest priority TCs. For example, starting from priority 7 and down; the other TCs should be used for groups with bandwidth allocation. It is recommended to use *LSP* only for one TC (TC7) as the first LSP TC takes its bandwidth and there are no guarantees to the lower priority LSPs.
2. The *GSP* bit can be set to more than one TC in a BWG, always from the highest priority TC within that BWG downward. For the LAN scenario, all TCs could be configured to be GSP as their bandwidth needs are not known.
3. For a low latency TC where the *GSP* bit is set, non-null refill credits must be set for at least one maximum-sized frame. This ensures that even after its been quiet for a while, some BWG credits are left available to the GSP TC, for serving it with minimum latency (without waiting for replenishing). Bigger refill credits values ensure longer bursts of GSP traffic served with minimum latency.

4.6.11.4.2 VM Parameters

Refill Credits:

Refill credits regulate the fraction of the TC's bandwidth that is allocated to a VM. The ratio between the credits of the VMs represent the relative TC bandwidth percentage allocated to each VM.

Credits are configured and calculated using 64-byte granularity.

1. In any case, the number of refill credits assigned per VM should be as small as possible but still larger than the maximum frame size used and larger than 1.5 KB. Using a lower refill value causes more refill cycles before a packet can be sent. These extra cycles increase the latency unnecessarily.
2. The refill credits ratio between VMs should be equal to the desired ratio of bandwidth allocation between the different TCs. Applying rule #1, means bandwidth shares are sorted from the smaller to the bigger, and just one maximum sized frame is allocated to the smallest.
3. The ratio between the refill credits of any two VMs within the TC should not be greater than 10.

VMs that send/receive packets of 127 bytes eventually get double the bandwidth it was configured to as all credits are calculated by rounding the values down to the next 64 byte-aligned value.

Refill and Maximum Credits (MaxCredits) Setting Example:

This example assumes a system with only four TCs and three VMs present, along with the following bandwidth allocation scheme. Also, note that full PCIe bandwidth is evaluated to 15 G.

Table 4-9. Bandwidth Share Example

TCs and VMs		Bandwidth Share%	Note
TC0	Total	40	9.5 KB jumbo frames allowed.
	VM0	60	
	VM1	30	
	VM2	10	
TC1	Total	20	No jumbo frames.
	VM0	34	
	VM1	33	
	VM2	33	
TC2	Total	30	Low latency TC. No jumbo frames. Bandwidth share already increased. MaxBurstTC2=120 KB
	VM0	80	
	VM1	10	
	VM2	10	
TC3	Total	10	Low latency LSP TC. No jumbo frames. MaxBurstTC3=36 KB
	VM0	20	
	VM1	60	
	VM2	20	

The ratios between TC refills are driven by TC0, which was set as 152 for supporting 9.5 KB jumbo frames.

The ratio between MaxCredits and Refill were taken as 17 for all the TCs, as driven by the TC2 relation between MaxCredits and MaxBurstTC2.

Table 4-10. Refill and MaxCredits Settings

TCs and VMs		Refill (64-Byte Units)	MaxCredits (64-Byte Units)
TC0	Total	152	2584
	VM0	912	
	VM1	456	
	VM2	152	
TC1	Total	76	1292
	VM0	25	
	VM1	24	
	VM2	24	
TC2	Total	114	1938
	VM0	192	
	VM1	24	
	VM2	24	
TC3	Total	38	646
	VM0	24	
	VM1	72	
	VM2	24	

4.6.11.4.3 Rate Limiters

It might be useful to setup rate limiters on Tx queues (rate-limiting VF traffic). Setting a rate-limiter on Tx queue N to a Target Rate requires the following settings:

- Use of advanced Tx descriptors
- RTTQCNRM.*MMW_SIZE* set to the amount of back-to-back compensation to be granted to a quiet item.
- RTTDQSEL.*TXQ_IDX* set to N.
- Compute the Rate Factor = Link Speed / Target Rate, where Link Speed is either 10 Gb/s or 1 Gb/s. The fractional binary number obtained is broken down into its 2^n components as follows:

  $b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0.b_{-1}b_{-2}b_{-3}b_{-4}b_{-5}b_{-6}b_{-7}b_{-8}b_{-9}b_{-10}b_{-11}b_{-12}b_{-13}b_{-14}$

  where b_n is the binary coefficient of the 2^n component.
- Set RTTQCNRC register with
  - Set RTTQCNRC[13:0] = RTTQCNRC.*RF_DEC[13:0]* = $[b_{-1}b_{-2}b_{-3}b_{-4}b_{-5}b_{-6}b_{-7}b_{-8}b_{-9}b_{-10}b_{-11}b_{-12}b_{-13}b_{-14}]$
  - *RTTQCNRC[23:14]* = RTTQCNRC.*RF_INT[9:0]* = $[b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0]$
  - RTTQCNRC[31] = RTTQCNRC.*RS_ENA* = 1b

Numerical Example:

- Target Rate = 4 Gb/s
- Link Speed = 10 Gb/s

  Rate Factor = 10 / 4 = 2.5 = $1x2^{1} + 1x2^{-1}$ = 0000000010.10000000000000b

  — RTTQCNRC[13:0] = [10000000000000]

  — RTTQCNRC[23:14] = [0000000010]

  — RTTQCNRC = [1000 0000 0000 0000 1010 0000 0000 0000] = 0x8000A000

4.6.12 Security Initialization

After power up or device reset, security offload is disabled by default (IPsec), and the content of the SA tables must be cleared by software.

Security offload cannot be enabled when either the security fuses are blown (set to zero), or the ENCRYPTION_EN pin is set to 0b. In this case, IPsec is disabled and the following security-related fields are not writable:

• SECTXCTRL.*SECTX_DIS* is set to 1b and read as 1b

• SECRXCTRL.*SECRX_DIS* is set to b1 and read as 1b

• IPSTXIDX.*IPS_TX_EN* is cleared to 0b and read as 0b

• IPSRXIDX.*IPS_RX_EN* is cleared to 0b and read as 0b

Security offload can be used when the security offload internal fuse is enabled and the ENCRYPTION_EN pin set to 1b. In this case, the security offload can be enabled/disabled via the flows that are described in the sections that follow.

4.6.12.1 Security Enable Flow

To enable one of the security modes do the following:

1. Stop the data paths by setting the SECTXCTRL.*TX_DIS* and SECRXCTRL.*RX_DIS* bits.
2. Wait for the data paths to be emptied by hardware. Poll the SECTXSTAT.*SECTX_RDY* and SECRXSTAT.*SECRX_RDY* bits until they are both asserted by hardware.
3. Clear the SECTXCTRL.*SECTX_DIS* and SECRXCTRL.*SECRX_DIS* bits to enable the Tx and Rx crypto engines.
   - When enabling IPsec offload, set the SECTXMINIFG.*MINSECIFG* bits to 0x3, extending back-to-back gap to the security block required for its functionality.
   - When enabling IPsec, set the SECTXCTRL.*STORE_FORWARD* bit, since a store and forward IPsec buffer is required for the processing of AH packets (*ICV* field insertion is done at the beginning of the frame). Otherwise, clear this bit.
   - When enabling IPsec, write the SEC buffer almost full threshold register SECTXBUFFAF.*FULLTHRESH* with the value of 0x15.
4. Enable SA lookup:
   - For IPsec, set the IPSTXIDX.*IPS_TX_EN* and the IPSRXIDX.*IPS_RX_EN* bits.
5. Restart the data paths by clearing the SECTXCTRL.*TX_DIS* and SECRXCTRL.*RX_DIS* bits.

4.6.12.2 Security Disable Flow

To disable one of the security modes do the following:

1. Stop the data paths by setting the SECTXCTRL.*TX_DIS* and SECRXCTRL.*RX_DIS* bits.
2. Wait for the data paths to be emptied by hardware. Poll the SECTXSTAT.*SECTX_RDY* and SECRXSTAT.*SECRX_RDY* bits until they are both asserted by hardware.
3. Disable SA lookup:
   - For IPsec, clear the IPSTXIDX.*IPS_TX_EN* and the IPSRXIDX.*IPS_RX_EN* bits.
4. Set the SECTXCTRL.*SECTX_DIS* and SECRXCTRL.*SECRX_DIS* bits to disable the Tx and Rx crypto engines.
   - When disabling IPsec, clear the SECTXCTRL.*STORE_FORWARD* bit to avoid using the IPsec buffer and thus reducing Tx internal latency.
   - When disabling IPsec, write the SEC buffer almost full threshold register SECTXBUFFAF.*FULLTHRESH* with the value of 0x250.
5. Restart the data paths by clearing the SECTXCTRL.*TX_DIS* and SECRXCTRL.*RX_DIS* bits.

Note: Disabling the crypto engine reduces the X550's power consumption.

4.6.13 Alternate MAC Address Support

In some systems, the MAC Address used by a port needs to be replaced with a temporary MAC Address in a way that is transparent to the software layer. One possible usage is in blade systems, to allow a standby blade to use the MAC Address of another blade that failed, so that the network image of the entire blade system does not change.

To allow this mode, a management console might change the MAC Address in the NVM image. It is important in this case to be able to keep the original MAC Address of the device as programmed at the factory.

To support this mode, the X550 provides the Alternate Ethernet MAC Address structure in the NVM to store the original MAC Addresses. This structure is described in Section 6.2.2.56.

In some systems, it might be advantageous to restore the original MAC Address at power on reset, to avoid conflicts where two network controllers would have the same MAC Address.

The X550 supports replacement of the MAC Address with an alternate MAC Address via the NC-SI interface using the BIOS CLP interface as described in the BIOS CLP document.

4.6.13.1 LAN MAC Address Restore

The X550 restores the LAN MAC Addresses stored in the Alternate Ethernet MAC Address structure to the regular LAN MAC Address location (see Section 6.2.5.3) if the following conditions are met:

1. The *restore MAC Address* bit in the *Common Firmware Parameters* NVM word is set.
2. The value in word 0x37 is not 0xFFFF.
3. The MAC Address set in the regular MAC Address location is different than the address stored in the Alternate Ethernet MAC Address structure.

4. The addresses stored in the Alternate Ethernet MAC Address structure are valid (not all zeros or all ones).

If the value at word 0x37 is valid, but the MAC Addresses in the Alternate MAC structure are not valid (0xFFFFFFFF or all zeros) and *restore MAC Address* NVM bit is set, the regular MAC Address is backed up in the Alternate MAC structure.

4.6.13.2 SAN MAC Address Restore

The X550 restores the SAN MAC Addresses (word 0x28 - see Section 6.2.2.41) at power on reset by copying the values from the Alternate SAN MAC Address structure (word 0x27; Section 6.2.2.40) to the SAN MAC Addresses if the following conditions are met:

1. The *Restore MAC Address* bit in the *Common Firmware Parameters* NVM word is set.
2. The pointer to the SAN MAC Address structure is valid (value in word 0x28 in not 0xFFFF - see Section 6.2.2.41).
3. The addresses stored in the Alternate SAN MAC Address structure (word 0x27 - see Section 6.2.11) are valid (not all zeros or all ones).

If the value at word 0x27 is valid, but the MAC Addresses in the Alternate SAN MAC structure are not valid (0xFFFFFFFF or all zeros), the regular SAN MAC Address (pointed by word 0x28) is backed up in the Alternate SAN MAC structure (pointed by word 0x27).

4.6.13.3 Restore Reporting

If after the power up cycle, the alternate SAN and LAN address equals to the SAN and LAN Factory MAC Addresses (either they were restored by the internal firmware or they were originally equal), the FWSM.*FACTORY_MAC_ADDRESS_RESTORED* bit is set. This bit is common to all ports.

4.7 Access to Shared Resources

Part of the resources in the X550 are shared between several software entities — namely the drivers of the two ports and internal firmware. To avoid contentions, a driver that needs to access one of these resources should use the *Gaining Control of Shared Resource by Software* flow described in Section 11.8.4.1 to acquire ownership of this resource, and use the *Releasing a Shared Resource by Software* flow described in Section 11.8.4.2 to relinquish ownership of this resource.

The shared resources are:

- NVM
- PHY 0 and PHY 1 registers
- MAC CSRs

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

Chapter 5 Power Management and Delivery

This section defines how power management is implemented in the X550.

5.1 Power Targets and Power Delivery

See Section 12.3.9 for the current consumption and Section 12.3.2 for the power supply specification.

5.2 Power Management

5.2.1 Introduction to X550 Power States

The X550 supports the D0 and D3 power states defined in the PCI Power Management and PCIe specifications. D0 is divided into two sub-states: D0u (D0 un-initialized), and D0a (D0 active). In addition, the X550 supports a Dr state that is entered when PE_RST_N pin is asserted (including the D3cold state).

Figure 5-1 shows the power states and transitions between them.

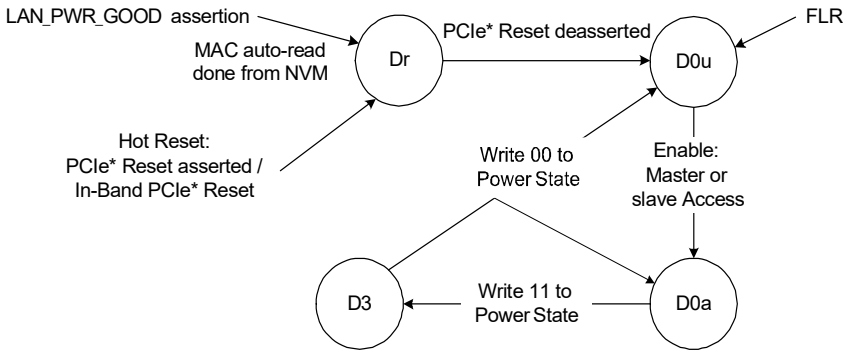


Figure 5-1. Power Management State Diagram

5.2.2 Auxiliary Power Usage

The X550 uses the *AUX_PWR* indication that auxiliary power is available to the controller, and therefore advertises D3cold wake-up support in the PMC.*PME_Support* field and sets the *Aux_Power_Detected* bit in the PCIe capability structure *Device Status* Register. The AUX_PWR pin is strapped during Power On Reset (POR). The amount of power required for the function, which includes the entire Network Interface Card (NIC), is advertised in the Power Management Data register, which is loaded from the NVM.

The only effect of setting *AUX_PWR* to 1b is advertising D3cold wake-up support and changing the reset function of *PME_En* and *PME_Status*. *AUX_PWR* is a strapping option in the X550. If D3cold is supported, the *PME_En* and *PME_Status* bits of the Power Management Control/Status Register (PMCSR), as well as their shadow bits in the Wake-Up Control Register (WUC) are reset only by the power up reset (detection of power rising). The X550 tracks the *PME_En* bit of the PMCSR and the Auxiliary (AUX) power *PM Enable* bit of the PCIe Device Control register to determine the power it can consume (and therefore its power state) in the D3cold state (internal Dr state). Note that the actual amount of power differs between form factors.

According to the following settings the X550 might consume higher auxiliary power than is allowed by PCIe specifications:

- If the *AUX Power PM Enable* bit of the PCIe Device Control register is set, the X550 might consume higher power for any purpose (even if *PME_En* is not set).
- If the *AUX Power PM Enable* bit of the PCIe Device Control register is cleared, higher power consumption is determined by the PCI-PM legacy *PME_En* bit of the PMCSR.

5.2.3 PCIe Link Power Management

The PCIe link state follows the power management state of the device. Since the X550 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 X550 supports the following PCIe power management link states:

- L0 state is used in D0u and D0a states.
- The L1 state is used in D0a and D0u states each time link conditions apply, as well as in the D3 state.
- The L2 state is used in the Dr state following a transition from a D3 state if PCI-PM PME is enabled.
- The L3 state is used in the Dr state following power up, on transition from D0a and also if PME is not enabled in other Dr transitions.

The X550 support for active state link power management is reported via the PCIe Active State Link PM Support register loaded from NVM.

The following NVM fields control L1 behavior:

- Act_Stat_PM_Sup — Indicates support for ASPM L1 in the PCIe configuration space (loaded into the *Active State Link PM Support* field).
- L1_Act_Ext_Latency — Defines L1 active exit latency.
- L1_Act_Acc_Latency — Defines L1 active acceptable exit latency.

5.2.4 Power States

5.2.4.1 D0uninitialized State

The D0u state is a low-power state used after PE_RST_N is de-asserted following power-up (cold or warm), on hot reset (in-band reset through PCIe physical layer message) or on D3 exit.

When entering D0u, the X550 disables wake-ups.

5.2.4.1.1 Entry to a D0u State

D0u is reached from either the Dr state (on de-assertion of Internal PE_RST_N) or the D3hot state (by configuration software writing a value of 00b to the *Power State* field of the PCI PM registers).

De-asserting internal PE_RST_N means that the entire state of the device is cleared, other than sticky bits. State is loaded from the NVM, followed by establishment of the PCIe link. Once this is done, configuration software can access the device.

On a transition from D3 to D0u state, the X550 requires that software perform a full re-initialization of the function not including its PCI configuration space which is kept while in D3.

5.2.4.2 D0active State

Once memory space is enabled, the X550 enters an active state. It can transmit and receive packets if properly configured by the software device driver. The PHY is enabled or re-enabled by the software device driver to operate/auto-negotiate to full line speed/power if not already operating at full capability. Any APM wake-up previously active remains active. The software device driver can deactivate ACPI wake-up by writing to the Wake-Up Control (WUC) register and APM wake-up by writing to the General Receive Control (GRC) register, or activate other wake-up filters by writing to the Wake-Up Filter Control (WUFC) register.

5.2.4.2.1 Entry to D0a State

D0a is entered from the D0u state by writing a 1b to the *Memory Access Enable* or the *I/O Access Enable* bit of the PCI Command register. The DMA, MAC, and PHY of the appropriate LAN function are enabled.

5.2.4.3 D3 State (PCI-PM D3hot)

The X550 transitions to D3 when the system writes a 11b to the *Power State* field of the PMCSR. Any wake-up filter settings that were enabled before entering this reset state are maintained. Upon transition to D3 state, the X550 clears the *Memory Access Enable* and *I/O Access Enable* bits of the PCI Command register, which disables memory access decode. In D3 the X550 only responds to PCI configuration accesses and does not generate master cycles.

Configuration and message requests are the only PCIe TLPs accepted by a function in the D3hot state. All other received requests must be handled as unsupported requests, and all received completions can optionally be handled as unexpected completions. If an error caused by a received TLP (like an unsupported request) is detected while in D3hot, and reporting is enabled, the link must be returned to L0 if it is not already in L0 and an error message must be sent. See section 5.3.1.4.1 in the PCIe Base Specification.

A D3 state is followed by either a D0u state (in preparation for a D0a state) or by a transition to Dr state (PCI-PM D3cold state). To transition back to D0u, the system writes a 00b to the *Power State* field of the PMCSR. Transition to Dr state is through PE_RST_N assertion.

5.2.4.3.1 Entry to D3 State

Transition to D3 state is through a configuration write to the *Power State* field of the PCI-PM registers.

Prior to transition from D0 to the D3 state, the software device driver disables scheduling of further tasks to the X550; it masks all interrupts, it does not write to the Transmit Descriptor Tail (TDT) register or to the Receive Descriptor Tail (RDT) register and operates the master disable algorithm as defined in Section 5.2.4.3.2. If wake-up capability is needed, the software device driver should set up the appropriate wake-up registers and the system should write a 1b to the *PME_En* bit of the PMCSR prior to the transition to D3.

If all PCI functions are programmed into D3 state, the X550 brings its PCIe link into the L1 link state. As part of the transition into L1 state, the X550 suspends scheduling of new TLPs and waits for the completion of all previous TLPs it has sent. The X550 clears the *Memory Access Enable* and *I/O Access Enable* bits of the PCI Command register, which disables memory access decode. Any receive packets that have not been transferred into system memory is kept in the device (and discarded later on D3 exit). Any transmit packets that have not be sent can still be transmitted (assuming the Ethernet link is up).

In preparation to a possible transition to D3cold state, and to reduce power consumption, whenever entering D3 state, the software device driver might disable one of the LAN ports (LAN disable) and/or the PHY(s) auto-negotiate network link(s) to a lower speed (if supported by the network interface). See Section 5.3.5.2 for a description of network interface behavior in this case.

5.2.4.3.2 Master Disable

System software might disable master accesses on the PCIe link by either clearing the *PCI Bus Master* bit or by bringing the function into a D3 state. From that time on, the X550 must not issue master accesses for this function. Due to the full-duplex nature of PCIe, and the pipelined design in the X550, it might happen that multiple requests from several functions are pending when the master disable request arrives. The protocol described in this section insures that a function does not issue master requests to the PCIe link after its *Master Enable* bit is cleared (or after entry to D3 state).

Two configuration bits are provided for the handshake between the device function and its software device driver:

- *PCIe Master Disable* bit in the Device Control Register (CTRL) register — When the *PCIe Master Disable* bit is set, the X550 blocks new master requests by this function. The X550 then proceeds to issue any pending requests by this function. This bit is cleared on master reset (LAN_PWR_GOOD all the way to software reset) to allow master accesses.
- *PCIe Master Enable Status* bits in the Device Status register — Cleared by the X550 when the *PCIe Master Disable* bit is set and no master requests are pending by the relevant function. Set otherwise. Indicates that no master requests are issued by this function as long as the *PCIe Master Disable* bit is set. The following activities must end before the X550 clears the *PCIe Master Enable Status* bit:
- Master requests by the transmit and receive engines
- All pending completions to the X550 are received.

Notes: The software device driver disables any reception to the Rx queues as described in Section 4.6.7.1.1. Then, the software device driver sets the *PCIe Master Disable* bit when notified of a pending master disable (or D3 entry). The X550 then blocks new requests and proceeds to issue any pending requests by this function. The software device driver then reads the change made in the *PCIe Master Disable* bit and then polls the *PCIe Master Enable Status* bit. Once the bit is cleared, it is guaranteed that no requests are pending from this function.

The software device driver might time out if the *PCIe Master Enable Status* bit is not cleared within a given time. Examples for cases that the X550 might not clear the *PCIe Master Enable Status* bit for a long time are cases of flow control, link down, or DMA completions not making it back to the DMA block. In these cases, the software device driver should check that the *Transaction Pending* bit (bit 5) in the Device Status register in the PCI config space is cleared before proceeding. In such cases, the driver needs to clear internal buffers by:

— Setting the HLREG0.*LPBK* bit.

— Clearing RXCTRL.*RXEN* bit.

— Setting the setting GCR_EXT.*Buffers_Clear_Func* bit for 20 µs.

— Initiate two consecutive software resets with a delay larger than 1 µs between them.

Intel's device driver software times-out at ~800 µs. The *PCIe Master Disable* bit must be cleared to enable master request to the PCIe link. This bit should be cleared through reset.

5.2.4.4 Dr State

Transition to Dr state is initiated on several occasions:

- **On system power up** — Dr state begins with the assertion of the internal power detection circuit (LAN_PWR_GOOD) and ends with de-assertion of PE_RST_N.
- **On transition from a D0a state** — During operation the system might assert PE_RST_N at any time. In an ACPI system, a system transition to the G2/S5 state causes a transition from D0a to Dr state.
- **On transition from a D3 state** — The system transitions the device into the Dr state by asserting PCIe PE_RST_N.

Any wake-up filter settings that were enabled before entering this reset state are maintained.

The system might maintain PE_RST_N asserted for an arbitrary time. The de-assertion (rising edge) of PE_RST_N causes a transition to D0u state.

While in Dr state, the X550 might maintain functionality (for WoL or manageability) or might enter a Dr Disable state (if no WoL and no manageability) for minimal device power. The Dr Disable mode is described in the sections that follow.

5.2.4.4.1 Dr Disable Mode

The X550 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 such as all PCI functions) is in Dr state.
- APM WoL is inactive for both LAN functions.
- Pass-through manageability is disabled.

- ACPI PME is disabled for all PCI functions.

Entry into Dr disable is usually done on assertion of PCIe PE_RST_N. It might also be possible to enter Dr disable mode by reading the NVM while already in Dr state. The usage model for this later case is on system power up, assuming that manageability and wake-up are not required. Once the X550 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 X550 then enters this mode (assuming that PCIe PE_RST_N is still asserted).

Exit from Dr disable is through de-assertion of PCIe PE_RST_N.

If Dr disable mode is entered from D3 state, the platform might remove X550 power. If the platform removes X550 power, it must remove all power rails from the X550 if it needs to use this capability. Exit from this state is through power-up cycle to the X550.

5.2.4.4.2 Entry to Dr State

Dr-entry on platform power-up is as follows:

- Asserting the internal power detection circuit (LAN_PWR_GOOD). Device power is kept to a minimum by keeping the PHYs in low power.
- The NVM is then read and determines device configuration.
- If the *APM Enable* bit in *NVM Control Word 3* is set, APM wake-up is enabled (for each port independently).
- Each of the LAN ports can be enabled if required for WoL or manageability. See Section 5.3.5.2 for exact condition to enable a port. In such a case, the PHY might auto-negotiate to a lower speed on Dr entry (see Section 5.3.5).
- The PCIe link is not enabled in Dr state following system power up (since PE_RST_N is asserted).

Entry to Dr state from D0a state is by asserting the PE_RST_N signal. An ACPI transition to the G2/S5 state is reflected in a device transition from D0a to Dr state. The transition can be orderly (such as user selected a shut down operating system option), in which case the software device driver might have a chance to intervene. Or, it might be an emergency transition (like power button override), in which case, the software device driver is not notified.

To reduce power consumption, if any of manageability, APM wake or PCI-PM PME[1] is enabled, the PHY auto-negotiates to a lower link speed on D0a to Dr transition (see Section 5.3.5).

Transition from D3 state to Dr state is done by asserting the PE_RST_N signal. Prior to that, the system initiates a transition of the PCIe link from L1 state to either the L2 or L3 state (assuming all functions were already in D3 state). The link enters L2 state if PCI-PM PME is enabled.

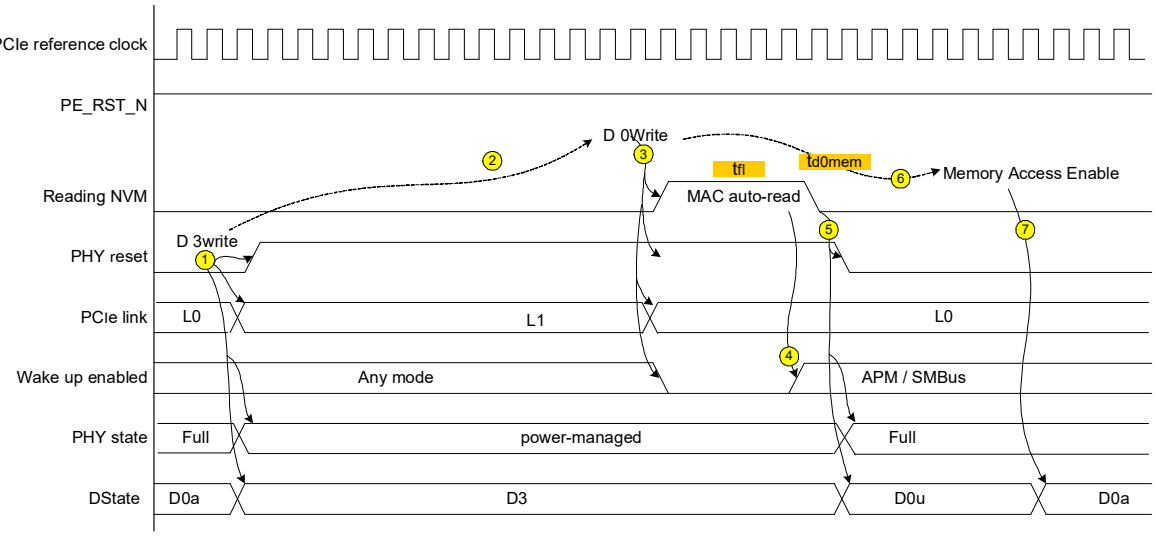
1. ACPI 2.0 specifies that OSPM does not disable wake events before setting the *SLP_EN* bit when entering the S5 sleeping state. This provides support for remote management initiatives by enabling Remote Power On (RPO) capability. This is a change from ACPI 1.0 behavior.

5.2.5 Timing of Power-State Transitions

The following sections give detailed timing for the state transitions. In the diagrams, the dotted connecting lines represent the X550 requirements, while the solid connecting lines represent the X550 guarantees.

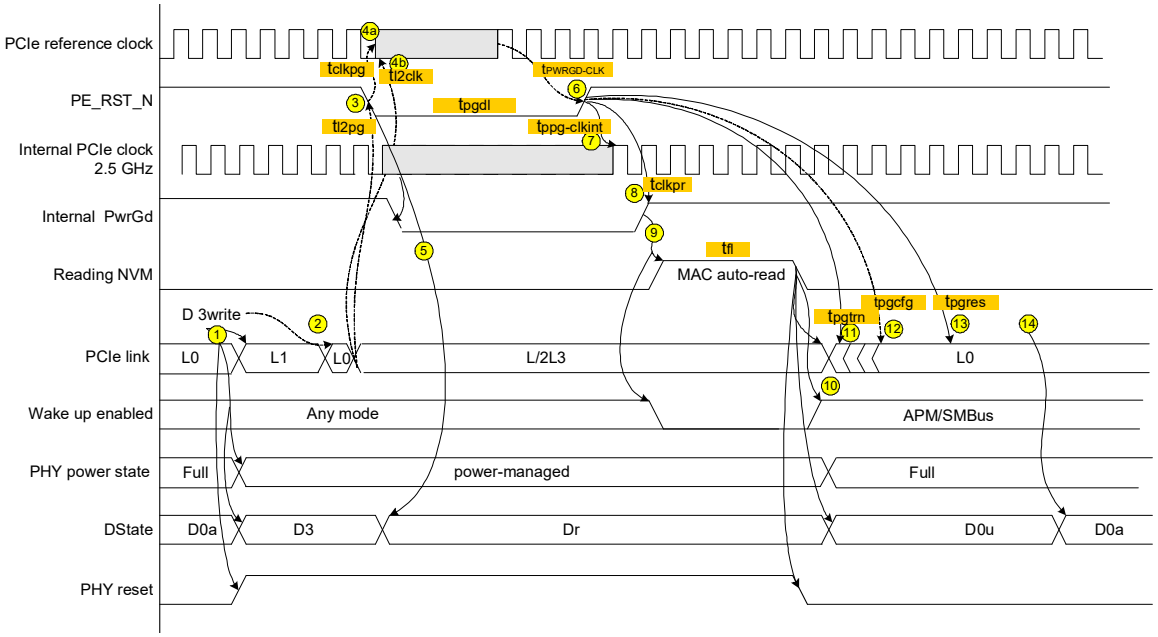
Note that the timing diagrams are not to scale. The clock's edges are shown to indicate running clocks only and are not used to indicate the actual number of cycles for any operation.

5.2.5.1 Transition from D0a to D3 and Back without PE_RST_N



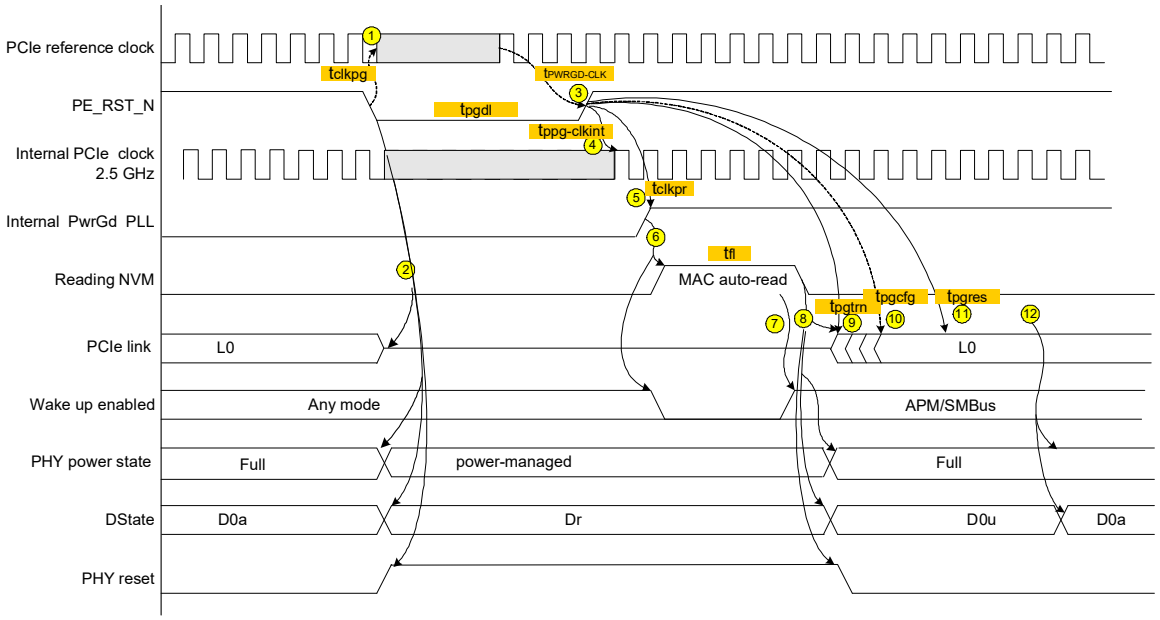
No.	Notes
1	Writing 11b to the *Power State* field of the PMCSR transitions the X550 to D3. PHY re-negotiates to a lower link speed once *VETO* bit is cleared if manageability and/or WoL is enabled, else it is powered down. When in power-down mode, PHY is reset and its CSRs are reset to their default values according to provisional register segment loaded from NVM at last LAN_PWR_GOOD event.
2	The system can keep the X550 in D3 state for an arbitrary amount of time.
3	To exit D3 state the system writes 00b to the *Power State* field of the PMCSR.
4	APM wake-up or manageability might be enabled based on what is read in the NVM.
5	After reading the MAC section of the NVM, the LAN ports are enabled and the X550 transitions to D0u state. PHY re-negotiates to full link speed.
6	The system can delay an arbitrary time before enabling memory access.
7	Writing a 1b to the *Memory Access Enable* bit or to the *I/O Access Enable* bit in the PCI Command register transitions the X550 from D0u to D0 state.

5.2.5.2 Transition from D0a to D3 and Back with PE_RST_N



No.	Notes
1	Writing 11b to the *Power State* field of the PMCSR transitions the X550 to D3. PCIe link transitions to L1 state. PHY re-negotiates to a lower link speed once *VETO* bit is cleared if manageability and/or WoL is enabled, else it is powered down. When in power-down mode, PHY is reset and its CSRs are reset to their default values according to provisional register segment loaded from NVM at last LAN_PWR_GOOD event.
2	The system can delay an arbitrary amount of time between setting D3 mode and transitioning the link to an L2 or L3 state.
3	Following link transition, PE_RST_N is asserted.
4	The system must assert PE_RST_N before stopping the PCIe reference clock. It must also wait t_{l2clk} after link transition to L2/L3 before stopping the reference clock.
5	On assertion of PE_RST_N, the X550 transitions to Dr state. Once entering Dr state, if manageability and/or WoL is enabled, PHY re-negotiates a lower link speed in any case, even if the *VETO* bit is asserted, else it is maintained powered down.
6	The system starts the PCIe reference clock $t_{PWRGD-CLK}$ before de-assertion PE_RST_N.
7	The Internal PCIe clock is valid and stable $t_{ppg-clkint}$ from PE_RST_N de-assertion.
8	The PCIe Internal PWRGD signal is asserted t_{clkpr} after the external PE_RST_N signal.
9	Assertion of Internal PCIe PWRGD causes the MAC section in the NVM to be re-read and disables wake-up. PHY is reset only if manageability unit and WoL are disabled. In such a case, PHY CSRs are reset to their default values according to provisional register segment loaded from NVM on last power-up event.
10	APM wake-up mode might be enabled based on what is read from the NVM. PHY re-negotiates to full link speed.
11	Link training starts after t_{pgtrn} from PE_RST_N de-assertion.
12	A first PCIe configuration access might arrive after t_{pgcfg} from PE_RST_N de-assertion.
13	A first PCI configuration response can be sent after t_{pgres} from PE_RST_N de-assertion.
14	Writing a 1b to the *Memory Access Enable* bit in the PCI Command register transitions the device from D0u to D0 state.

5.2.5.3 Transition from D0a to Dr and Back without Transition to D3



No.	Notes
1	The system must assert PE_RST_N before stopping the PCIe reference clock. It must also wait tl2clk after link transition to L2/L3 before stopping the reference clock.
2	On assertion of PE_RST_N, the X550 transitions to Dr state and the PCIe link transition to electrical idle. Once entering Dr state, if manageability and/or WoL is enabled, PHY re-negotiates a lower link speed in any case, even if the *VETO* bit is asserted, else it is powered down. When in power-down mode, PHY is reset and its CSRs are reset to their default values according to provisional register segment loaded from NVM at last LAN_PWR_GOOD event
3	The system starts the PCIe reference clock $t_{PWRGD-CLK}$ before de-assertion PE_RST_N.
4	The Internal PCIe clock is valid and stable $t_{ppg-clkint}$ from PE_RST_N de-assertion.
5	The PCIe Internal PWRGD signal is asserted t_{clkpr} after the external PE_RST_N signal.
6	Asserting Internal PCIe PWRGD causes the MAC section in the NVM to be re-read and disables wake-up. PHY is reset only if manageability unit and WoL are disabled. In such a case, PHY CSRs are reset to their default values according to provisional register segment loaded from NVM on last power-up event.
7	APM wake-up mode might be enabled based on what is read from the NVM.
8	After reading the NVM, PHY re-negotiates to full link speed.
9	Link training starts after t_{pgtrn} from PE_RST_N de-assertion.
10	A first PCIe configuration access might arrive after t_{pgcfg} from PE_RST_N de-assertion.
11	A first PCI configuration response can be sent after t_{pgres} from PE_RST_N de-assertion.
12	Writing a 1b to the *Memory Access Enable* bit in the PCI Command register transitions the device from D0u to D0 state.

5.2.5.4 Timing Requirements

The X550 requires the following start-up and power-state transitions.

Table 5-1. Start-up and Power-State Transitions

Parameter	Description	Min	Max	Notes
t_{xog}	Xosc stable from power stable.		10 ms	
$t_{PWRGD-CLK}$	PCIe clock valid to PCIe power good.	100 µs	-	According to PCIe specification.
t_{PVPGL}	Power rails stable to PCIe PE_RST_N inactive.	100 ms	-	According to PCIe specification.
t_{pgcfg}	External PE_RST_N signal to first configuration cycle.	100 ms		According to PCIe specification.
t_{d0mem}	Device programmed from D3h to D0 state to next device access.	10 ms		According to PCI power management specification.
t_{l2pg}	L2 link transition to PE_RST_N assertion.	0 ns		According to PCIe specification.
t_{l2clk}	L2 link transition to removal of PCIe reference clock.	100 ns		According to PCIe specification.
t_{clkpg}	PE_RST_N assertion to removal of PCIe reference clock.	0 ns		According to PCIe specification.
t_{pgdl}	PE_RST_N assertion time.	100 µs		According to PCIe specification.

5.2.5.5 Timing Guarantees

The X550 guarantees the following start-up and power state transition related timing parameters.

Table 5-2. Start-up and Power-State Transition Timing Parameters

Parameter	Description	Min	Max	Notes
t_{xog}	Xosc stable from power stable.		10 ms	
t_{ppg}	Internal power good delay from valid power rail.	35 ms	35 ms	
t_{fl}	NVM read duration.		20 ms	
$t_{ppg-clkint}$	PCIe PE_RST_N to internal PLL lock.	-	50 µs	
t_{clkpr}	Internal PCIe PWGD from external PCIe PE_RST_N.		50 µs	
t_{pgtrn}	PCIe PE_RST_N to start of link training.		20 ms	According to PCIe specification.
t_{pgres}	External PE_RST_N to response to first configuration cycle.		1 second	According to PCIe specification.

5.3 Network Interfaces Power Management

The X550 transitions any of the network interfaces into a low-power state in the following cases:

- The respective LAN function is in LAN disable mode using LANx_DIS_N pin.
- The X550 is in D3 or Dr state, APM WoL is disabled for the port, ACPI wake is disabled for the port and pass-through manageability is disabled for the port.

Use of the LAN ports for pass-through manageability follows the following behavior:

- If manageability is disabled (*Manageability Mode* field in the *Common Firmware Parameters* NVM word is set to None - Section 6.2.16), LAN ports are not allocated for manageability.
- If manageability is enabled:
  - Power-up — Following an NVM read, a single port is enabled for manageability, running at the lowest speed supported by the interface. If APM WoL is enabled on a single port, the same port is used for manageability. Otherwise, manageability protocols (such as teaming) determine which port is used.
  - D0 state — Both LAN ports are enabled for manageability.
  - D3 and Dr states — A single port is enabled for manageability, running at the lowest speed supported by the interface. If WoL is enabled on a single port, the same port is used for manageability. Otherwise, manageability protocols such as teaming) determine which port is used.

Enabling a port as a result of the previous behaviors cause an internal reset of the port including its associated PHY.

When a network interface is in low-power state, the X550 MAC asserts internal signals to notify the PHY that it must either power down as well or re-negotiate to a lower link speed.

5.3.1 PHY Power-Down State

Each X550 port enters a power-down state when none of its clients is enabled and therefore has no need to maintain a link. This can happen in one of the following cases:

1. **D3/Dr state** — Each PHY enters a low-power state if all the following conditions are met:
   a. The LAN function associated with this PHY is in a non-D0 state
   b. APM WoL is inactive
   c. Manageability does not use this port
   d. ACPI PME is disabled for this port
2. **LAN disable pin** — Each PHY is disabled if the associated LAN disable input pin indicates that the port should be disabled.
3. ***LAN PCI Disable* bit in NVM** — A single LAN port can also be disabled through NVM settings. If the *LAN PCI Disable* bit is set in *NVM Control Word 2*, and if the port is not used for WoL or for MC traffic, the *LAN Disable Select* bit selects the MAC and PHY port that enters power down even in D0 state. Note that if the port is used for WoL or by the MC, setting the *LAN PCI Disable* bit in *NVM Control Word 2* does not bring the MAC and PHY into power down, but only the DMA block.

When powered down by one of these means, a significant portion of the MAC and the PHY (including its microprocessor, MDIO, and interrupt logic) are powered down. Only the PHY PLLs are functioning, and the Analog Front-End (AFE) is turned off (gets placed in high-impedance mode).

When the X550 is completely powered down (Dr state), the PHYs reach a deeper power saving mode.

When the PHY exits power down, it re-initializes all analog functions, and retrieves the default configuration settings that were loaded from NVM the last time NVM was read.

5.3.2 PHY Power-Down via the PHY Register

The PHY can also be powered down by setting a *Low Power* bit in a PHY register (Global Standard Control 1: Address 1E.0, bit B). This bit powers down a significant portion of the PHY but clocks provided to the MAC and to the register section of the PHY remain active. Only the MDIO, interrupts, and microprocessor are functioning, and the AFE is turned off (gets placed in high-impedance mode). This enables the PHY management interface to remain active during register power down. Setting this bit also sets all of the *Low Power* bits in the other MMDs.

When the PHY exits software power down (*Low Power* bit cleared), it re-initializes all analog functions, but retains its previous configuration settings.

5.3.3 Smart Power-Down (SPD)

Smart Power-Down (SPD) is a feature that allows the PHY to enter a deep power saving state if it detects that the link is down for a long period of time. In this mode, the PHY does not transmit Link pulses. To detect a connection of a partner, the PHY periodically wakes up and sends link pulses for a short period of time to allow detection by the link partner. SPD is applicable to all power management states.

SPD combines a power-saving mechanism with the fact that the link might disappear and resume. Smart power-down is enabled by *Smart Power-Down Enable* PHY register field, which can be provisioned in the PHY NVM section, and is entered when the PHY detects link loss.

SPD is supported only when working in auto-negotiation mode, and not in parallel detect mode.

While in the smart power-down state, the PHY powers down circuits and clocks that are not required for detection of link activity. The PHY is still able to detect link pulses (including parallel detect) and wakeup to engage in link negotiation. The PHY does not send auto-negotiation words (FLP) while in SPD state unless configured to Deadlock-avoidance as defined in Section 5.3.3.1. Register accesses to the PHY is still possible.

Note: Since some of the PHY blocks might be in power-down during SPD, Access to the registers of these blocks is not supported during SPD.

PHY indicates SPD-Status via the *Smart Power-Down Status* bit. This register is accessible to the MAC in all the PHY power-modes.

When the PHY is in smart power-down and detects link activity, it re-negotiates link speed based on the power state as in the normal case.

Note: The link-disconnect state applies to all power management states (Dr, D0u, D0a, D3). The link might change status, that is go up or go down, while in any of these states

See Section 5.3.3.2 for details of the SPD control bit.

5.3.3.1 Deadlock Avoidance Mechanism (FLP)

While in link disconnect, the X550 monitors the link for Fast Link Pulses to identify when a link is reconnected. The X550 also periodically transmits pulses to resolve the case of two X550 (or devices with X550-like behavior) connected to each other across the link. Otherwise, two such devices might be locked in smart power-down mode, not capable of identifying that a link was re-connected. The link pulses are transmitted on average every 100 ms on alternate channels (A/B and C/D), and add negligible power to total X550 power at link disconnect mode.

Pulses might not conform to IEEE specification regarding link pulse template.

If the link partners are disconnected and then reconnected, it is possible that the two controllers transmit their pulses at the same time. Since the X550 might mask its receiver during pulse transmission, such synchronization causes pulses to be missed by both partners. A randomization factor is therefore applied to the timing of transmitted pulses, affecting the period between pulses. The randomization factor is specific per device and should reduce the probability of a lock to at least 10-4.

Note: If the two partners happen to transmit within the same slot, and if the randomization factor happens to be similar, it takes longer for the partners to get out of sync with each other.

Deadlock-Avoidance smart power-down is enabled by the *Deadlock Avoidance Enable* bit in the PHY registers. The default value is enabled. The *Enable* bit applies only while in smart power-down mode.

Note: The emission of FLP by the two ports of the X550 are not synchronized to avoid issues in tests where a link is established between the two ports.

5.3.3.2 SPD Control

The following bits in the PHY registers are used to control/monitor the SPD feature:

- Controls
  - *Deadlock Avoidance Enable* — Enables deadlock avoidance by transmitting FLP while in smart power down mode — 1E.C475.3 (Section 10.6.18).
  - *Smart Power-Down Enable* — 1E.C475.2 (Section 10.6.18).
  - *Smart Power-Down Entered Mask* (Enable interrupt generation) — The mask bit is to enable an interrupt to the MAC in case SPD was entered — 1E.D401.E (Section 10.6.47).
- Statuses
  - *Smart Power-Down Status* (active, inactive) — 1E.C475.D (Section 10.6.18).
  - *Smart Power-Down Entered* — When this bit is set, it indicates that the smart power down state was entered — 1E.CC01.E (Section 10.6.44).

Note: The *Smart Power-Down Entered* bit is self cleared when read, but it is set again by the next try of the Deadlock avoidance mechanism to establish a link.

5.3.3.3 Timing Definitions

When port is configure to SPD it should meet the following timing constraints:

Table 5-3. SPD Timing Parameters

Parameter	Description	Min	Typ	Max	Remark
T_{spd-in}	Time elapsed from the Link-Disconnect event until Port actually enter into SPD mode.	2 sec		10 sec	
$T_{spd-out}$	Time elapsed from the Link-Reconnect event (first link pulse arrives at package pins) until Port actually exit from SPD mode and start the auto-negotiation mode.	70 ms		100 ms	
$T_{b2b-flp}$	Time between two consecutive FLPs (Fast-Link-Pulses) transmitted by TVL during Deadlock-Avoidance mode.	90 ms	100 ms	110 ms	The $T_{b2b-flp}$ Min/Max numbers include the randomization factor during the Deadlock-Avoidance mode.

5.3.4 Disable 10GBASE-T and/or 1000BASE-T Speeds

To reduce power consumption of a port even when it is active, there is an option to disable 10 Gb/s and/or 1000 Mb/s advertisement during auto-negotiation.

This can be useful for cases where a copper link is used as a dormant link for redundancy purposes only.

These options are enabled by the following bits in PHY registers:

- Auto-Negotiation 10GBASE-T Control Register: Address 7.20, bit C - Disable 10 Gb/s in any state.
- Auto-Negotiation Vendor Provisioning 1: Address 7.C400, bit F - Disable 1000 Mb/s in any state.

Note: These bits disable the advertisement of a speed in any power state, whether the port is active (D0u or D0a) or in low power (Dr or D3).

5.3.5 Low Power Link Up (LPLU)

The PHY is aware of power management states. If the PHY is not in a power down state, PHY behavior regarding several features are different depending on the device or port power state. The internal PHY power state is controlled internally from the MAC.

Normal PHY speed negotiation drives to establish a link at the highest possible speed. The PHY supports an additional mode of operation, where the PHY drives to establish a link at a low speed. The LPLU process enables a link to come up at the lowest possible speed in cases where power is more important than performance. Different behavior is defined for the D0 state and the other non-D0 states.

Table 5-4 lists link speed as function of power management state, link speed control, and GbE speed enabling:

Table 5-4. Link Speed vs. Power State

Power State	LPLU Bit in NVM	Disable Bits in NVM		Disable Bits in Register		PHY Speed Negotiation
		Disable 10 GbE in LPLU Mode	Disable 1 GbE in LPLU Mode	Disable 10 GbE in Any State	Disable 1 GbE in Any State	
D0	X	X	X	0b	0b	PHY advertises 10 GbE, 1 GbE, 100 Mb/s
				1b	0b	PHY advertises 1 GbE and 100 Mb/s only.
				0b	1b	PHY advertises 10 GbE and 100 Mb/s only.
				1b	1b	PHY advertises 100 Mb/s only.
Non-D0	0b	X	X	0b	0b	PHY advertises 10 GbE, 1 GbE, 100 Mb/s
				1b	0b	PHY advertises 1 GbE and 100 Mb/s only.
				0b	1b	PHY advertises 10 GbE and 100 Mb/s only.
				1b	1b	PHY advertises 100 Mb/s only.
	1b	0b	0b	0b	0b	PHY goes through LPLU procedure, starting from 100 Mb/s, up to 1 GbE and 10 GbE.
				1b	0b	PHY goes through LPLU procedure, starting from 100 Mb/s and followed by 1 GbE only.
				0b	1b	PHY goes through LPLU procedure, starting from 100 Mb/s and followed by 10 GbE only.
				1b	1b	PHY advertise 100 Mb/s only.
		1b	0b	X	0b	PHY goes through LPLU procedure, starting from 100 Mb/s and followed by 1 GbE.
				X	1b	PHY advertises 100 Mb/s only.
		1b	1b	X	X	PHY advertises 100 Mb/s only.
		0b	1b	X	X	N/A (non supported setting)

X means Don't care.
LPLU bit in NVM - Refers to *NVM Control Word 3*, bit 3, shared by both ports.
Disable 10 GbE in LPLU mode - Refers to *NVM Control Word 3*, bits 6 and 8, one bit per port.
Disable 1 GbE in LPLU mode - Refers to *NVM Control Word 3*, bits 5 and 7, one bit per port.
Disable 10 GbE any state - Refers to PHY register 7.20[C].
Disable 1 GbE any state - Refers to PHY register 7.C400[F].

Notes: The software device driver is responsible to re-start auto-negotiation when it changes the setting of the *Disable 10 GbE* or *1 GbE* bits in the PHY register.

For proper LPLU functionality, *Upshift-Enabled* bit must be set in the PHY at register 7.C411.0 (default is 1b, enabled).

If manageability and WoL are both disabled, the directives of Table 5-4, when in non-D0 state, are not relevant as the PHY port is disabled.

5.3.5.1 Behavior in Non-D0 State

If the *LPLU* bit is set in the NVM, the PHY negotiates to a low speed while in non-D0 states (Dr or D3). This applies only when the link is required by one of the following: SMBus or NC-SI manageability, APM wake, or PME. Otherwise, the PHY is disabled during the non-D0 state.

Link negotiation begins with the PHY trying to negotiate at the lowest speed it is allowed to advertise, as listed in Table 5-4. If link establishment fails, the PHY tries to negotiate at additional speeds, such as all speeds allowed up to the lowest speed supported by the partner. For example, the PHY advertises 100 Mb/s only and the partner supports 1000 Mb/s only. After the first try fails, PHY enables 100/1000 Mb/s and tries again. The PHY continues to try and establish a link until it succeeds or until it is instructed otherwise.

Note: Automatic MDI/MDI-X resolution is done during the first auto-negotiation stage.

5.3.5.2 Link Speed Change vs. Power Mode

Normal speed negotiation drives to establish a link at the Highest Common Denominator (HCD) link speed. The X550 supports an additional mode of operation, where the PHY establishes a link at the Lowest Common Denominator (LCD) link speed. The LPLU process allows a link to come up at any possible speed in cases where power is more important than performance. Different behavior is defined for the D0 state and non-D0 states as a function of the *D10GMP*, *D1GMP* bits in the NVM and of the MMNGC.*MNG_VETO* register bit.

The X550 initiates auto-negotiation without a direct software device driver command in the following cases:

- When the state of MAIN_PWR_OK pin changes.
- When the *MNG_VETO* bit value changes.
- On a transition from D0 state to a non-D0 state, or from a non-D0 state to D0 state.

During a manageability session, any change in a power management state must not cause the Ethernet link to drop because the manageability session will be lost. Therefore in such a case the Ethernet link speed should be kept unchanged. For example, the transition to D3hot state is not propagated to the PHY as long as a manageability session exists.

Note: However, if main power is removed, the PHY is allowed to react to the change in power state (such as the PHY might respond in link speed change). The motivation for this exception is to reduce power when operating on auxiliary power by reducing link speed.

The capability of masking from the PHY the changes that occur in a power management state is enabled by default on LAN_PWR_GOOD reset. The *Keep_PHY_Link_Up_En* bit in *NVM Control Word 3* - Offset 0x38 word must be cleared to disable it. Once enabled, the feature is enabled until the next LAN_PWR_GOOD (such as the X550 does not revert to the hardware default value on PE_RST_N, PCIe reset, or any other reset but LAN_PWR_GOOD).

Existence of a manageability session is identified by the *MNG_VETO* bit set in the MMNGC register.

The *MNG_VETO* bit is set by the MC through the Management Control command (see Section 11.5.11.1.5 for SMBus command and Section 11.6.3.12 for NC-SI command) on the sideband interface. It is cleared by the external MC (also through a command on the sideband interface) when the manageability session ends.

The *MNG_VETO* bit becomes meaningless when de-asserting the MAIN_PWR_OK input pin. MAIN_PWR_OK must be 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 5-2 shows the X550 behavior when entering low power mode:

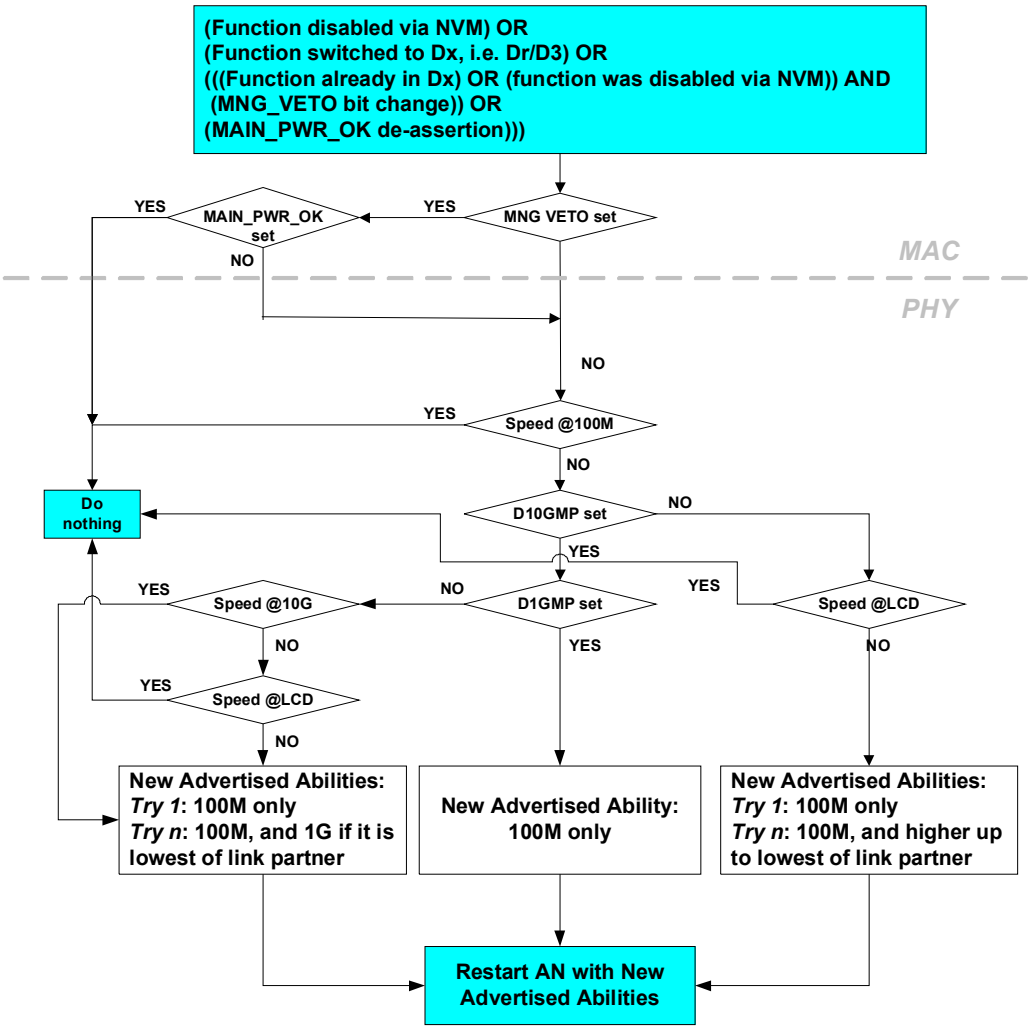


Figure 5-2. Link Speed Change when Entering Power-Down Mode

Figure 5-3 shows the X550 behavior when going to power-up mode.

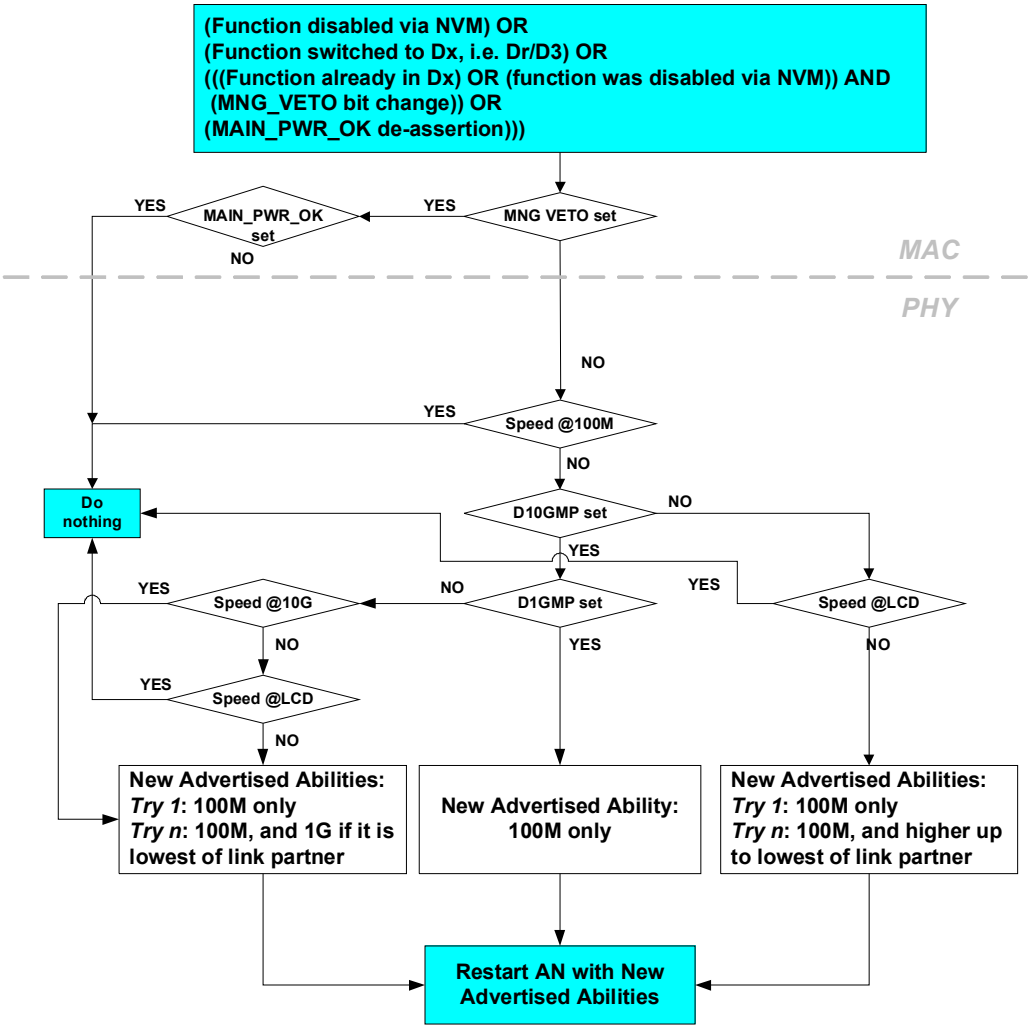


Figure 5-3. Link Speed Change when Entering Power-Up Mode

Notes: To simplify things, the impact of the *LPLU* and *Keep_PHY_Link_Up_En* bits in the *NVM Control Word 3* is not shown in the flow charts because these bits are assumed to be set. If the *Keep_PHY_Link_Up_En* bit is cleared, the *VETO* bit set condition always returns NO.

The same remark for the impact of the *Disable Speed* bits described in Section 5.3.4, which are assumed to be cleared.

The PHY re-negotiates full line speed once exiting power down state, without waiting for the *Memory Access Enable* bit of the PCIe Command register to be written.

5.3.6 Energy Efficient Ethernet (EEE)

the X550 enters EEE Low Power Idle (LPI) mode on transmit or receive independently each time the X550 detects no data is scheduled for transmission, or the link partner indicates no data is pending for reception.

EEE LPI mode defined in IEEE802.3az enables power saving by switching off part of the X550 functionality when no data needs to be transmitted or/and received. Decision on whether the X550 transmit path should enter LPI mode or exit LPI mode is done according to a need to transmit. Information on whether a link partner has entered LPI mode is detected by the X550 and utilized for power saving in the receive circuitry.

When no data needs to be transmitted, a request to enter transmit LPI is issued on the internal xxMII Tx interface causing the PHY to transmit sleep symbols for a predefined period of time followed by a quite period. During LPI, the PHY periodically transmits refresh symbols that are used by the link partner to update adaptive filters and timing circuits to maintain link integrity. This quiet-refresh cycle continues until transmitting 'normal inter-frame' encoding on the internal xxMII Tx interface. The PHY communicates to the link partner the move to link active state by sending wake symbols for a predefined period of time. The PHY then enters a normal operating state where data or idle symbols are transmitted.

In the receive direction, entering LPI mode is triggered by receiving sleep symbols from the link partner. This signals that the link partner is about to enter LPI mode. After sending the sleep symbols, the link partner ceases transmission. When the link partner initiates the move to LPI, the PHY indicates "assert low power idle" on the internal xxMII Rx interface and the X550 receiver disables functionality to reduce power consumption.

Figure 5-4 and Table 5-5 illustrate the general principles of EEE LPI operation on the Ethernet link.

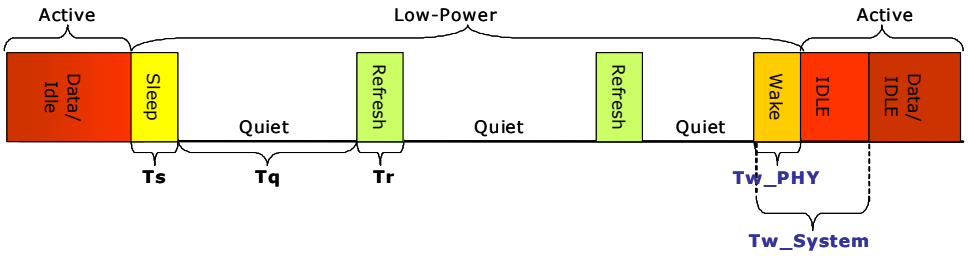


Figure 5-4. EEE Operation

Table 5-5. EEE Parameters

Parameter	Description
Sleep Time (Ts)	Duration PHY sends sleep symbols before going quiet.
Quiet Duration (Tq)	Duration PHY remains quiet before it must wake for refresh period.
Refresh Duration (Tr)	Duration PHY sends refresh symbols for timing recovery and coefficient synchronization.
PHY Wake Time (Tw_PHY)	Minimum duration PHY takes to resume to active state after decision to wake.
Receive System Wake Time (Tw_System_rx)	Wait period where no data is expected to be received to give the local receiving system time to wake-up.
Transmit System Wake Time (Tw_System_tx)	Wait period where no data is transmitted to give the remote receiving system time to wake-up.

5.3.6.1 Conditions to Enter EEE Tx LPI

In the transmit direction entry into EEE LPI mode of operation is triggered when one of the following conditions exist:

1. No transmission is pending, management does not need to transmit and internal transmit buffer is empty and the EEER.*TX_LPI_EN* bit is set to 1b. If the EEER.*TX_LPI_EN* bit is set to 1b and a XOFF flow control packet is received from the link partner or PFC XOFF are received for all the traffic classes supporting PFC, the X550 moves the link into the Tx LPI state for the pause duration even if a transmission is pending.
2. When EEER.*FORCE_TLPI* is set (even if EEER.*TX_LPI_EN* is cleared).
   - If EEER.*FORCE_TLPI* is set in mid-packet, the X550 completes the packet transmission and moves Tx to LPI.

When one of the previous conditions to enter Tx LPI state are detected "assert LPI" is transmitted on the internal xxMII interface and the X550 PHY transmits sleep symbols on the network interface to communicate to the link partner entry into Tx LPI link state. After sleep symbols transmission, the PHY immediately enters the low power quiet mode. In this state the PHY periodically transitions between quiet link state, where link is idle, to sending refresh symbols until a request to transition link back to normal (active) mode is transmitted on the internal xxMII Tx interface (See Figure 5-4).

Note: Initial EEER.*TX_LPI_EN* configuration is loaded from NVM.

Note: 10GBASE-T Ethernet LPI allows each link direction to enter sleep, refresh or wake states asymmetric from the other direction.

Note: EEE LPI status of a X550 port can be found in the EEE_STAT register.

5.3.6.2 Transition from Tx LPI to Active Link State

The X550 exits Tx LPI link state and transition link into active link state when none of the conditions defined in Section 5.3.6.1 exist. To transition into active link state, the X550 transmits:

1. Normal 'inter-frame' encoding on the internal xxMII Tx interface for a pre-defined link rate dependent period time of Tw_sys_tx-min (defined by the EEE_SU register and IEEE802.3az clause 78.5). As a result, the PHY transmits wake symbols for a Tw_phy duration followed by idle symbols.
2. If the Tw_System_tx duration defined in the EEER.*TW_SYSTEM* field is longer than Tw_sys_tx-min, the X550 continues transmitting the 'inter-frame' encoding on the internal xxMII interface until the time defined in the EEER.*TW_SYSTEM* field has expired, before transmitting the actual data. During this period the PHY continues transmitting idle symbols.

Note: When moving out of Tx LPI to transmit a 802.3x flow control frame, the X550 waits for the Tw_sys_tx-min duration before transmitting the flow control frame. It should be noted that even in this scenario actual data is transmitted only after the Tw_System_tx time defined in the EEER.*TW_SYSTEM* field has expired.

5.3.6.3 EEE Auto-Negotiation

Auto-negotiation provides the capability to negotiate EEE capabilities with the link partner using the next page mechanism defined in IEEE802.3 Annex 73A. IEEE802.3 auto-negotiation is performed at power up, on command from software, upon detection of a PHY error or following link re-connection.

During the link establishment process, both link partners indicate their EEE capabilities using IEEE802.3 auto-negotiation. If EEE is supported by both link partners for the negotiated PHY type, the EEE function can be used independently in either direction.

5.3.6.4 EEE Link Level (LLDP) Capabilities Discovery

The X550 supports LLDP negotiation via software using the EEE IEEE802.1AB Link Layer Discovery Protocol (LLDP) Type, Length, Value (TLV) fields defined in IEEE802.3az clause 78 and clause 79. LLDP negotiation enables negotiation of increased system wake time (Transmit T_w and Receive T_w) to enable improving system energy efficiency.

5.3.6.4.1 LLDP Negotiation Actions

Following negotiation of a new system wake time via EEE LLDP negotiation, the following fields and registers should be updated:

The EEER.*TW_SYSTEM* field with the negotiated Transmit T_w time value, to increase the duration where idle symbols are transmitted following a move out of EEE Tx LPI state before actual data can be transmitted.

- A value placed in EEER.*TW_SYSTEM* field does not effect transmission of flow control packets. Depending on the technology flow control packet, transmission is delayed following a move out of EEE Tx LPI state only by the minimum Tw_sys_tx time as defined in IEEE802.3az clause 78.5. In the X550 the minimum Tw_sys_tx time value is defined in the *EEE_SU* register together with time defined in IEEE802.3az clause 78.5. Value varies as a function of link rate and technology.

5.3.6.5 EEE Statistics

The X550 supports reporting a number of EEE LPI Tx and Rx events via the RLPIC and TLPIC registers.

5.4 Wake-Up

5.4.1 Advanced Power Management Wake-Up

Advanced power management wake-up, or APM wake-up, was previously known as Wake on LAN (WoL). It is a feature that has existed in the 10/100 Mb/s NICs for several generations. The basic premise is to receive a packet with an explicit data pattern, and then to assert a signal to wake-up the system. In the earlier generations, this was accomplished by using a special signal that ran across a cable to a defined connector on the motherboard. The NIC asserts the signal for approximately 50 ms to signal a wake-up. The X550 uses (if configured to) an in-band PM_PME message for this.

At power up, the X550 reads the *APM Enable* bit from the NVM into the *APM Enable* (*APME*) bits of the GRC register. This bit control the enabling of *APM Wakeup*.

When *APM Wakeup* is enabled and when not in D0 state, the X550 checks all incoming packets for Magic Packets. See Section A.1.3 for a definition of Magic Packets.

Once the X550 receives a matching magic packet, it:

- Sets the *PME_Status* bit in the *PMCSR* and issues a PM_PME message (in some cases, this might require to assert the PE_WAKE_N signal first to resume power and clock to the PCIe interface).

  Note: At wake-up from first power on, the PM_PME message is not sent.

- Stores the first 128 bytes of the packet in the Wake-Up Packet Memory (WUPM).
- Sets the *Magic Packet Received* bit (*MAG*) in the Wake-Up Status (WUS) register.
- Sets the packet length in the Wake-Up Packet Length Register (WUPL).

The X550 maintains the first magic packet received in the Wake-Up Packet Memory (WUPM) until the software device driver writes a 0b to the Magic Packet Received *MAG* bit in the Wake-Up Status Register (WUS), hence the software device driver has to read the packet from WUPM prior to clearing the WUS indication.

APM Wakeup event is issued only when in Dx power states and it is disabled only if a subsequent NVM read results in the *APM Wake-Up* bit being cleared or if the software explicitly writes a 0b to the *APM Wake-Up* (*APM*) bit of the GRC register.

5.4.2 ACPI Power Management Wake-Up

The X550 supports ACPI power management based wake-up. It can generate system wake-up events from three sources, regardless to the power state:

- Receiving a Magic Packet.
- Receiving a network wake-up packet.
- Detecting a link change of state.
- Firmware reset event.

Activating ACPI power management wake-up requires the following steps:

- The operating system (at configuration time) writes a 1b to the *PME_En* bit of the PMCSR (PMCSR.8).

- The software device driver clears all pending wake-up status in the WUS register by writing 1b to all the status bits.
- The software device driver programs the WUFC register to indicate the packets it needs to wake-up and supplies the necessary data to the IPv4/v6 Address Table (IP4AT, IP6AT), Flexible Host Filter Table (FHFT). It can also set the *Link Status Change Wake-Up Enable* (*LNKC*) bit in the WUFC register to cause wake-up when the link changes state.
- Once the X550 wakes the system the software device driver needs to clear the WUS and WUFC registers until the next time the system goes to a low power state with wake-up.

Normally, after enabling ACPI wake-up, the operating system writes (11b) to the lower two bits of the PMCSR to put the X550 into low-power mode, and once entering D0, OS disables ACPI wake-up.

Once wake-up is enabled, the X550 monitors incoming packets, first filtering them according to its standard address filtering method, then filtering them with all of the enabled wake-up filters. If a packet passes both the standard address filtering and at least one of the enabled wake-up filters, the X550:

- Sets the *PME_Status* bit in the PMCSR.
- If the *PME_En* bit in the PMCSR is set and device is in Dr state or PCI_GLBL_CNF.*WAKE_PIN_EN* is set in all states, asserts PE_WAKE_N.
- In non Dr state or when exiting Dr, a PM_PME message is also issued.
- Stores the first 128 bytes of the packet in the Wake-Up Packet Memory (WUPM) register. Sets one or more of the *Received* bits in the WUS register. Note that the X550 sets more than one bit if a packet matches more than one filter. Sets the packet length in the Wake-Up Packet Length Register (WUPL).

If enabled, a link state change wake-up causes similar results, setting *PME_Status*, asserting PE_WAKE_N and setting the *LNKC* bit in the WUS register when the link goes up or down.

Note: As the LPLU mechanism may cause a link change when entering Sx, setting the link state change wake-up while link is up may cause an immediate wake-up when entering Sx. If this is needed, the software device driver should move the link to the lowest available link speed before enabling the wake-up event and entering Sx.

If enabled, a firmware reset causes similar results, setting *PME_Status*, asserting PE_WAKE_N and setting the *FW_RST_WK* bit in the *WUS* register when the firmware is reset.

PE_WAKE_N remains asserted until the operating system either writes a 1b to the *PME_Status* bit of the PMCSR register or writes a 0b to the *PME_En* bit.

5.4.3 Wake-Up Packets

The X550 supports various wake-up packets using two types of filters:

- Pre-defined filters
- Flexible filters

Each of these filters are enabled if the corresponding bit in the WUFC register is set to 1b.

5.4.3.1 Pre-Defined Filters

The following packets are supported by the X550's pre-defined filters:

- Directed packet (including exact, multicast indexed, and broadcast)
- Magic Packet
- ARP/IPv4 request packet
- Directed IPv4 packet
- Directed IPv6 packet

Each of these filters are enabled if the corresponding bit in the WUFC register is set to 1b.

The description of each filter includes a table showing which bytes at which offsets are compared to determine if the packet passes the filter. Both VLAN frames and LLC/SNAP can increase the given offsets if they are present.

5.4.3.1.1 Directed Packets

Unicast — The X550 generates a wake-up event after receiving any packet whose destination address matches one of the 128 valid programmed receive addresses if the *Directed Exact Wake-Up Enable* bit is set in the WUFC register (WUFC.*EX*).

For multicast packets, the upper bits of the incoming packet's destination address index a bit vector, the Multicast Table Array (MTA) that indicates whether to accept the packet. If the *Directed Multicast Wake-Up Enable* bit set in the WUFC register (WUFC.MC) and the indexed bit in the vector is one, the X550 generates a wake-up event. The exact bits used in the comparison are programmed by software in the *Multicast Offset* field of the Multicast Control register (MCSTCTRL.MO).

If the *Broadcast Wake-Up Enable* bit in the WUFC register (WUFC.BC) is set, the X550 generates a wake-up event when it receives a broadcast packet.

5.4.3.1.2 Magic Packet

See Section A.1.3 for a definition of Magic Packets.

A Magic Packet's destination address must match the address filtering enabled in the configuration registers with the exception that broadcast packets are considered to match even if the *Broadcast Accept* bit of the Receive Control register (FCTRL.*BAM*) is 0b. If *APM Wakeup* is enabled in the NVM, the X550 starts up with the Receive Address Register 0 (RAH0, RAL0) loaded from the NVM. This enables the X550 to accept packets with the matching IEEE address before the software device driver comes up.

Note: Setting the FCTRL.*MPE* does not allow reception of magic packets with a multicast address. To receive such packets, the multicast address should be stored in an RHA/RAL register or enabled in the MTA table.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	MAC header – processed by main address filter
6	6	Source Address		Skip	
12	S= (0/4)	Possible VLAN Tag		Skip	
12/16	D=(0/8)	Possible Len/LLC/SNAP Header		Skip	
12+S+D	2	Type		Skip	
Any	6	Synchronizing Stream	FF*6+	Compare	
any+6	96	16 Copies of Node Address	A*16	Compare	Compared to Receive Address Register 0 (RAH0, RAL0)

5.4.3.1.3 ARP/IPv4 Request Packet

The X550 supports receiving ARP request packets for wake-up if the *ARP* bit is set in the WUFC register. Four IPv4 addresses are supported, which are programmed in the IPv4 Address Table (IP4AT). A successfully matched packet must pass L2 address filtering, a Protocol Type of 0x0806, an ARP OpCode of 0x01, and one of the four programmed IPv4 addresses. The X550 also handles ARP request packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	MAC header – processed by main address filter
6	6	Source Address		Skip	
12	S=(0/4)	Possible VLAN Tag		Compare	
12/16	D=(0/8)	Possible Len/LLC/SNAP Header		Skip	
12+S+D	2	Type	0x0806	Compare	ARP
14+S+D	2	Hardware Type	0x0001	Compare	
16+S+D	2	Protocol Type	0x0800	Compare	
18+S+D	1	Hardware Size	0x06	Compare	
19+S+D	1	Protocol Address Length	0x04	Compare	
20+S+D	2	Operation	0x0001	Compare	
22+S+D	6	Sender Hardware Address	-	Ignore	
28+S+D	4	Sender IP Address	-	Ignore	
32+S+D	6	Target Hardware Address	-	Ignore	
38+S+D	4	Target IP Address	IP4AT	Compare	Can match any of 4 values in IP4AT

5.4.3.1.4 Directed IPv4 Packet

The X550 supports receiving directed IPv4 packets for wake-up if the *IPV4* bit is set in the WUFC register. Four IPv4 addresses are supported that are programmed in the IPv4 Address Table (IP4AT). A successfully matched packet must pass L2 address filtering, a Protocol Type of 0x0800, and one of the four programmed IPv4 addresses. The X550 also handles directed IPv4 packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

5.4.3.1.5 Directed IPv6 Packet

The X550 supports receiving directed IPv6 packets for wake-up if the *IPV6* bit is set in the WUFC register. One IPv6 address is supported and it is programmed in the IPv6 Address Table (IP6AT). A successfully matched packet must pass L2 address filtering, a Protocol Type of 0x0800, and the programmed IPv6 address. The X550 also handles directed IPv6 packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

5.4.3.2 Flexible Filter

The X550 supports a total of **eight** host flexible filters. Each filter can be configured to recognize any arbitrary pattern within the first 128 bytes of the packet. To configure the flexible filter, software programs the required values into the Flexible Host Filter Table (FHFT_FILTER). These contain separate values for each filter. Software must also enable the filter in the WUFC register, and enable the overall wake-up functionality must be enabled by setting *PME_En* in the PMCS register or the WUC register.

Structure of the Flexible Host Filter Table:

31 0	31 8	7 0	31 0	31 0
Reserved	Reserved	Mask [7:0]	DW 1	DW 0
Reserved	Reserved	Mask [15:8]	DW 3	DW 2
Reserved	Reserved	Mask [23:16]	DW 5	DW 4
Reserved	Reserved	Mask [31:24]	DW 7	DW 6

. . .

31 7	6 0	31 8	7 0	31 0	31 0
Reserved	Reserved	Reserved	Mask [127:120]	DW 29	DW 28
Reserved	Length	Reserved	Mask [127:120]	DW 31	DW 30

Each of the filters is allocated addresses as follows:

- Filter 0 — 0x09000 — 0x090FF
- Filter 1 — 0x09100 — 0x091FF
- Filter 2 — 0x09200 — 0x092FF
- Filter 3 — 0x09300 — 0x093FF
- Filter 4 — 0x09600 — 0x096FF
- Filter 5 — 0x09700 — 0x097FF
- Filter 6 — 0x09800 — 0x098FF
- Filter 7 — 0x09900 — 0x099FF

The following table lists the addresses used for filter 0.

Field	DWord	Address	Bit(s)	Initial Value
Filter 0 DW0	0	0x09000	31:0	X
Filter 0 DW1	1	0x09004	31:0	X
Filter 0 Mask[7:0]	2	0x09008	7:0	X
Reserved	3	0x0900C		X
Filter 0 DW2	4	0x09010	31:0	X
…				
Filter 0 DW30	60	0x090F0	31:0	X
Filter 0 DW31	61	0x090F4	31:0	X
Filter 0 Mask[127:120]	62	0x090F8	7:0	X
Length	63	0x090FC	6:0	X

Accessing the FHFT_FILTER registers during filter operation might result in a packet being mis-classified if the write operation collides with packet reception. It is therefore advised that the flex filters are disabled prior to changing their setup.

Once enabled, the flexible filters scan incoming packets for a match. If the filter encounters any byte in the packet where the mask bit is one and the byte does not match the byte programmed in the Flexible Host Filter Table (FHFT_FILTER), the filter fails that packet. If the filter reaches the required length without failing the packet, it passes the packet and generates a wake-up event. It ignores any mask bits set to one beyond the required length.

Packets that passed a wake-up flexible filter should cause a wake-up event only if it is directed to the X550 (passed L2 and VLAN filtering).

Note: The flexible filters are temporarily disabled when read from or written to by the host. Any packet received during a read or write operation is dropped. Filter operation resumes once the read or write access is done.

5.4.3.3 Wake-Up Packet Storage

The X550 saves the first 128 bytes of the wake-up packet in its internal buffer, which can be read through the Wake-Up Packet Memory (WUPM) after the system wakes up.

5.4.4 Wake-Up and Virtualization

When operating in a virtualized environment, all wake-up capabilities are managed by a single entity (such as the VMM or an IOVM). In an IOV architecture, the physical software device driver controls wake-up and none of the Virtual Machines (VMs) has direct access to the wake-up registers. The wake-up registers are not replicated.

5.5 DMA Coalescing

The X550 supports DMA coalescing that enables synchronizing port activity and optimizes power consumption of memory and CPU usage. DMA coalescing is a device-based feature and when the conditions to enter DMA coalescing operating mode exists (as defined in Section 5.5.2.1), the X550:

- Stops initiation of any activity on the PCIe link. This includes all the DMA transactions and MCTP VDMs (if DMACR.*EN_MNG_IND* is set).
- Buffers data received from the Ethernet link in internal Rx buffer until the conditions defined in Section 5.5.2.2 to exit DMA coalescing exist. If DMACR.*EN_MNG_IND* is set, MCTP pass-through traffic is buffered also.

5.5.1 DMA Coalescing Activation

Activating DMA coalescing functionality is done by the software device driver. The software device driver can enable or disable the device DMA coalescing functionality by clearing the DMACR.*DMAC_EN* bit.

To activate DMA coalescing functionality the following fields need to be programmed:

1. DMCTH.*DMACRXT* fields to set the per Rx packet buffer receive threshold that causes the X550 to move out of DMA coalescing operating mode. When the amount of data in the receive packet buffer exceeds the receive threshold, the X550 moves out of DMA coalescing operating mode. A programmed receive watermark should take into account the actual link speed, Latency Tolerance Reported (LTR) and L1/L0s to L0 latency to avoid receive buffer overflow when DMA coalescing is enabled. See LTR description in Section 5.6.
2. DMACR.*DMACWT* field to define a maximum time-out value for:
   a. A receive packet to be stored in the internal receive buffer before the X550 moves a packet to host memory.
      - Each time the X550 enters DMA coalescing, an internal DMA coalescing watchdog timer is re-armed with the value placed in DMACR.*DMACWT*. When in DMA coalescing, the internal watchdog timer starts to count when one of the following conditions occurs:
        — An incoming Rx packet is received.
        — An MCTP transaction is pending.
        — An interrupt is pending.

   b. Once an interval defined in the DMACR.*DMACWT* field has passed, the X550 exits DMA coalescing, internal buffers are flushed, RSC flows are closed and interrupts are flushed.
3. DMCTLX.*TTLX* timer field to define the time between detection of DMA idle condition and entry into DMA coalescing state. To limit entry into DMA coalescing state when packet rate is high.
4. DMACR.*DMAC_EN* bit should be set to 1b to enable activating DMA coalescing operating mode.
5. DMACR.*LX_COALESCING_INDICATION* bit defines whether to move in/out of DMA coalescing when the PCIe block moves in/out of L1/L0s, when set to 1b, DMA coalescing conditions are met only when PCIe is in L1/L0s, when set to 0b DMA coalescing can also start in L0. In addition, when the bit is set to 0b, DMA coalescing stops when any TLP transactions are executed on the PCIe.
6. DMACR.*EN_MNG_IND* bit should be set to 1b to enable a management indication impact on DMA coalescing mode. This bit also enables a DMA coalescing impact on MCTP over PCIe traffic.

7. DMCMNGTH.*DMCMNGTHR* field to set the threshold for the management data buffer that causes a move out of DMA coalescing operating mode. Any control data (packet generated by internal firmware) that is pending on MCTP over PCIe interrupts the coalescing. The manageability indications are ignored if DMACR.*EN_MNG_IND* is cleared.

Note: Any change to a DMA coalescing configuration should be done while the feature is disabled by setting DMAR.*DMAC_EN* to 0b.

5.5.2 DMA Coalescing Operating Mode

Enabling DMA coalescing operation by setting the DMACR.*DMAC_EN* bit to 1b, enables aligning bus master traffic and interrupts from all ports. Power saving is achieved since synchronizing PCIe accesses between ports increases the occurrence of idle intervals on the PCIe bus and also increases the duration of these idle intervals. The Power Management Unit (PMU) on a platform can use these idle intervals to reduce system power.

5.5.2.1 Conditions to Enter DMA Coalescing

The X550 enters DMA coalescing when all of the following conditions exist:

1. DMA coalescing is enabled (DMACR.*DMAC_EN* = 1b).
2. Internal receive buffers are empty.
3. There are no pending DMA operations.
4. There are no MCTP pending operations (if DMACR.*EN_MNG_IND* is set).
5. None of the conditions defined in Section 5.5.2.2 to move out of DMA coalescing exist.

5.5.2.2 Conditions to Exit DMA Coalescing

When the X550 is in DMA coalescing operating mode, DMA coalescing mode is exited when one of the following events occurs:

1. Amount of data in the internal receive buffers passed the DMCTH.*DMACRXT* threshold.
2. An interrupt associated to a high priority vector defined by EITR.*HIGH_PRIORITY* was detected.
3. A received packet associated to a high priority traffic class defined by DMACR.*HIGH_PRIORITY* was detected.
4. DMA coalescing watchdog timer expires as a result of the following occurrences not being serviced for the duration defined in the *DMACWT* timer field:
   a. An incoming Rx packet is received.
   b. An MCTP transaction is pending (if DMACR.*EN_MNG_IND* is set).
   c. An interrupt is pending.
   d. An RSC flow was closed due to an ITR expiration.
5. DMA coalescing is disabled (DMACR.*DMAC_EN* = 0b).
6. PCIe link moves to an active L0 link power state (if the DMACR.*LX_COALESCING_INDICATION* bit is set to 1b) or any TLP transaction is detected on the PCIe interface (if the DMACR.*LX_COALESCING_INDICATION* bit is set to 0b).

Notes: Management indications are enabled through *DMACR.EN_MNG_IND* and the amount of data buffered in the management buffer exceeds DMCMNGTH.*DMCMNGTHR*.

Even when conditions for DMA coalescing do not exist, the X550 continues to be in low power PCIe link state (L0s or L1) if there is no requirement for PCIe access.

5.5.3 DMA Coalescing Recommended Settings

These are the recommended DMA Coalescing settings for the X550:

1. DMA Coalescing Receive Threshold per traffic class (DMCTH.*DMACRXT[TC]*) = MAX{(RPB_Size[TC] - 70us of Rx data), MFS}

Note: The receive threshold must be set so that its distance from the Effective Rx Packet Buffer size (Section 7.1.3.6.5) allows buffering all Rx traffic that could be received at full blown during 70us, which corresponds to the worst case observed exit time from L1/L0s to L0. It means that the threshold setting depends on the link speed. A minimum of MFS (Maximum Frame Size) must be set for the DMA Coalescing threshold.

2. DMA Coalescing Watchdog Timer (DMACR.*DMACWT*) must be set to get a worst case packet delay of ~ 20ms.
   a. If TTLX >= Max ITR delay over all Qs: DMACWT = 0x262 ~ 20ms. Flush entire RPB to host before re-entering into DMA coalescing state. The longest time a packet can wait in packet buffer is DMACWT.
   b. If TTLX < Max ITR delay over all Qs: DMACWT = (20ms - Max ITR delay) / 2. The device re-enters the DMA coalescing state as soon as the first packet which is waiting in packet buffer is served. Then, when the ITR timer for the second packet expires, the device starts the DMACWT timer (again). It comes out that in this case, the longest time a packet can wait in packet buffer is ITR delay + 2 x DMACWT.
3. Time to LX request (DMCTLX.*TTLX*) = 0x20 ~ 40us, so that it is smaller than worst case L1/L0s max exit time (70us), taking advantage of the fact coalescing occurs in L0 too.
4. DMACR.Lx_Coalescing_Indication = 0b, i.e. coalescing occurs when in L0.
5. DMA Coalescing Management Threshold (DMCMNGTH.*DMCMNGTHR*) = 0x100 (Equivalent to 4 KB)

5.6 LTR

The X550 generates PCIe LTR messages to report service latency requirements for memory reads and writes to the root complex for system power management.

The X550 reports either latency tolerance or no latency tolerance requirements as a function of link, LAN port and function status. The reported latency tolerance value is set to optimize platform power consumption without incurring packet loss due to receive buffer overflow.

5.6.1 LTR Algorithm

The X550 sends LTR messages according to the following algorithm when the capability is enabled in the LTR capability structure of Function 0 located in the PCIe configuration space:

1. When links on all ports are disconnected, or all LAN ports are disabled, the software device driver instructs the X550 to send an LTR PCIe message with LTR requirement bits cleared, to indicate that no latency tolerance requirements exist.
2. If the X550 reported following PCIe link-up latency tolerance requirements with any requirement bit set in the PCIe LTR message and all enabled functions were placed in D3 low power state via the *PMCSR* register, the X550 sends a new LTR message with all the requirement bits clear.
3. If the X550 reported following PCIe link-up latency tolerance requirements with any requirement bit set, and the LTR *Mechanism Enable* bit in the PCIe configuration space is cleared, the software device driver instructs the X550 to send an LTR PCIe message with all the requirement bits clear.
4. The X550 holds the LTR value to send defined in LTR Control (*LTRC*) controlled and configured by the software device driver. The X550 sends a new LTR message with the minimum latency tolerance requirement value upon an explicit request from the software device driver (issued by setting LTRC.*LTR_SEND* bit). The following event can cause a the X550 software device driver to change its latency tolerance requirement:
   a. Link speed changed
   b. Link was disconnected

The X550 conglomerates latency requirements from the different functions and sends a single LTR message in the following manner:

- The acceptable latency values for the message sent upstream by the X550 must reflect the lowest latency tolerance values associated with any function.
  — It is permitted that the snooped and non-snooped values reported in the conglomerated message are associated with different functions.
  — If none of the functions have a latency requirement for a certain type of traffic (snoop/non-snoop), the message sent by the X550 does not have the *Requirement* bit corresponding to that type of traffic set.
  — The Scale parameter should be set to zero by the software device driver if minimal latency is required for a specific type of traffic (Latency = 0 and Requirement is set).

5.6.2 LTR Initialization Flow

- BIOS/OS configures the maximum snoop/non-snoop platform latency in the PCIe (function 0) configuration LTR capability registers.
- As part of software device driver initialization, the software device driver:
  - Reads the LTR capability to get the worst case latency tolerance requirement for the platform that the X550 is connected to.
  - Auto negotiates extended LPI exit latencies.
  - The software device driver configures receive buffer watermarks for DMAC as aggressively as possible for the active target latency tolerance value (pre defined value).
  - The software device driver configures the content of the LTR snoop/non-snoop requirements to LTRC.
- The software device driver initiates sending a LTR message from the X550 by setting LTRC.Send.
- When a software device driver unloads, it should clear its latency tolerance requirements in the LTRC register and set the LTRC.*LTR_SEND* bit, this could cause a new LTR message as defined in Section 5.6.1.

5.7 Thermal Management

5.7.1 General

The X550's thermal management solution includes an on-die diode and an on-die thermal sensor used to:

- Monitor the die temperature (junction temperature) by an external device. Refer to Section 12.5 for more details.
- Indicate junction temperature as well as being programmed to indicate that temperature thresholds (such as trip points) are reached.
- The thermal sensor indications can be monitored by:
  - The MC (see Section 5.7.2)
  - Autonomously by the X550 in NVM-based mode (see Section 5.7.3).
  - By host software driver in NVM-based mode (see Section 5.7.3.1).

Thermal sensor registers are generally accessible only from MDIO of PHY 0.

Note: Host software might read these registers but must not change their programming. PHY registers remain accessible even after the PHY has been shut down by a thermal sensor event.

5.7.2 MC-Based Mode

The thermal sensor behavior relative to the thresholds/actions configured by MC is referred as the TS MC-based mode. It can be used by the MC to control an active heat sink or the fan operation if present on the board.

This mode can work together with the NVM mode, where the NVM mode is used by the software device driver, and this mode is used by the MC. Each mode uses different thresholds in the PHY.

The MC-based mode is configured via the Set Thermal Sensor Commands. Refer to Section 11.5.11.1.9 (Legacy SMBus mode) and Section 11.6.3.19 (NC-SI mode).

Thermal sensor event notifications can be issued:

- Via an AEN over NC-SI (see Section 11.6.3.20.1)
- Via an SMBus alert (see Section 11.5.6).
- Via SDP1 output pin of LAN port 1 indication.

5.7.3 NVM-Based Mode

The thermal sensor behavior relative to the threshold configured via NVM is referred as the TS NVM-based mode. It is mainly useful for non-managed environments or when the MC does not support the thermal sensor commands. However, this mode can also be enabled in parallel to the TS MC-based mode, as a back up mode relying on a catastrophic threshold, which must be set beyond the thresholds configured by the MC to avoid any interference between the two defined modes (NVM based and MC based).

It is configured as follows:

- TS NVM-based mode is enabled via setting the *TS NVM-based Mode Enable* bit to 1b in the *Common Firmware Parameters* word (Section 6.2.16).
- Program the PHY NVM section as follows:
  - The thermal threshold should be set to 107 °C, by the PHY provisioning register 1E.C421 (Section 10.6.11) and 1E.C423 (Section 10.6.13) both set to 0x6B00.
  - Enable the high temperature failure alarm to assert the PHY Global Interrupt, via PHY provisioning register 1E.D400 bits E:B set to 1000b.
- Optionally, if on-board thermal alerts are required, set the SDP1 pin of LAN port 0 as an output pin that asserts on the thermal sensor event, via the *SDP1_IODIR* bit (bit 9) set to 1b, the *SDP1_NATIVE* bit (bit 17) set to 0b, and the *SDP1_FUNCTION* bit (bit 25) set to 1b in the *SDP Control* word (Offset 0x20).
- Recovering from a thermal sensor PHY power down event in this mode requires a device power cycle.
- The host can check whether the TS NVM-based mode is enabled or not by reading bit 0 in FWSM register.

5.7.3.1 Thermal Sensor (TS) Monitoring by Host Software

When operating in NVM mode, thermal events can be monitored by the host device driver. The thermal sensor is configured by the NVM and should not be changed by host software. The junction temperature can be read from the Global Thermal Status 1: Address 1E.C820 register (Section 10.6.29).

Note: A guard band of ±3 °C should be taken over thermal sensor temperature indication.

TS event notifications can be issued via:

- An interrupt to the host (via EICR.*TS* bit 23).
- SDP1 output pin of LAN port 0 indication.

Note: TS events are enabled by default and might be spurious. Host software should react TS events only if TS NVM-based mode is enabled (Section 5.7.3).

Note: To get an indication through SDP1, the NVM setting of SDP1 should be configured as described in Section 5.7.3.

5.7.4 Thermal Sensor Control

The temperature sensor can be manually controlled by the host device driver through the MDIO, by setting the *Temperature Sense Override* bit to 1b.

Once the temperature sensor has performed the read, it asserts the *Temperature Ready* bit to 0b (see Global Thermal Status 2: Address 1E.C821 — Section 10.6.30). The temperature data can then be read from the Temperature [F:0] bits (see Common Thermal Status 1: Address 1E.2C820 — Section 10.6.29).

Note: Take in account that the *Temperature Sense Ready* bit toggles a number of times according to the number of samples configured in *Temperature Sense Sample Configuration* field (1E.2825) and to the wait time between samples configured in *Temperature Sense Wait Configuration* field (1E.2824).

5.7.5 Thermal Sensor Characteristics

Table 5-6. Thermal Sensor Characteristics

Parameter	Min	Max	Comment
Range	0 °C	125 °C	Temperature is reported as 16-bit unsigned decimal value.
Resolution	1/256 °C per LSB		For the least significant bit.
Accuracy	+/- 3 °C		
Conversion time	11 ms		

Chapter 6 Non-Volatile Memory Map

6.1 NVM Organization

The X550 NVM contains the following six high-level modules:

- **Legacy EEPROM Modules** — These modules are mapped over one of the first two 16 KB sections of the Flash device, and cannot be extended beyond them. It is composed of all the NVM modules used by MAC hardware or by manageability firmware, not including the manageability and PHY firmware code images. The sections included in this block are:
  - **NVM Pointers and Generic Words** — This section (described in Section 6.1) start is at the beginning of the valid section. It contains basic device information, pointers to other NVM modules, and several software configurations.
  - **PCI Analog Module** — This module is pointed by NVM word 0x2, and contains settings for the PCI PHY block. It is loaded only at Power On Reset.
  - **LCB Module** — This module is pointed by NVM word 0x3, and contains settings for the PCI Link Layer block. It is loaded at each PCI reset.
  - **PCIe Modules** — These modules (described in Section 6.2.6 and Section 6.2.7) contain the parameters required to configure the PCIe configuration space. The pointers to these modules are located at NVM words 0x6 (Generic), 0x7 (Port 0), and 0x8 (Port 1).
  - **LAN Core Modules** — These two modules (described in Section 6.2.5) contain parameters required to configure the LAN port. These include the MAC Address, LED, and SDP configuration. The pointers to these modules are located at NVM words 0x9 (Port 0) and 0xA (Port 1). In addition, each LAN port has a module that enables loading specific CSR values after reset. The pointers to these modules are located at NVM words 0xD (Port 0) and 0xE (Port 1). These modules are loaded following a software or hardware reset.
  - **Firmware Parameters Module** — Pointed by Firmware Module Pointer located at NVM word 0x0F. The Firmware Extension Module starts with a list of firmware sub-modules pointers, as listed in Section 6.2.13.
  - **Boot Configuration** — This module is pointed by NVM word 0x17, and contains the configuration parameters used by the PXE, iSCSI and FCoE boot code.
  - **VPD** — This module is pointed by NVM word 0x2F, and contains the VPD module exposed via the VPD PCIe capability as described in Section 6.2.2.48.
- **Firmware Image** — This module contains the main code of the firmware loaded to the internal RAM. This image must fit within 512 KB and start at a 4 KB boundary. This module is authenticated upon update. See Section 6.1.4 for details.
- **PCIe Expansion/Option ROM** — This module includes the PXE Driver (61 KB), iSCSI Boot Image (116 KB), FCoE Boot Image (80 KB), UEFI Network Driver (37 KB for x64, 67 KB for IA64), and can also include a CLP module (60 KB). It must fit within 512 KB and start at a 4 KB boundary. It is pointed by the *PCIe Expansion/Option ROM Pointer* located at the NVM word 0x05.
- **PHY Image** — This module includes the PHY micro controller Boot Vector section, two PHY registers provisional segments (one per port), and a PHY micro controller code/data segment. The PHY registers provisional segments are used to change the defaults of PHY registers. It must fit within 512 KB and start at a 4 KB boundary. The *PHY Image Pointer* is located at NVM word 0x04.

- **FCoE Scratch Pad** — A scratch pad used for FCoE software usage. The location of this area is described in the module pointed by word 0x39. The minimal allocated size is 8 KB, but can be larger if there is additional free area available in the NVM. The scratch pad is open to software read or write. Software device driver is responsible for the maintenance of this area.
- **Free Space Provisioning Segment** — The NVM structure includes a space utilized to update the PXE code, Firmware Image, and PHY Image modules via a double image policy. This space is referred as the Free Space Provisioning module or segment. It must be large enough to contain the largest of these three high-level modules, which means at least 512 KB in a 4 MB flash and 500 KB in a 2 MB flash. It is pointed by the *Free Space Provisioning Segment Pointer* located at NVM word 0x40. See Section 3.4.8.1 for the usage model of this Flash area.

Note: Flash device size can be read from NVM Control Word 1 (see Section 6.2.2.1).

Figure 6-1 shows a general NVM structure and not a required order.

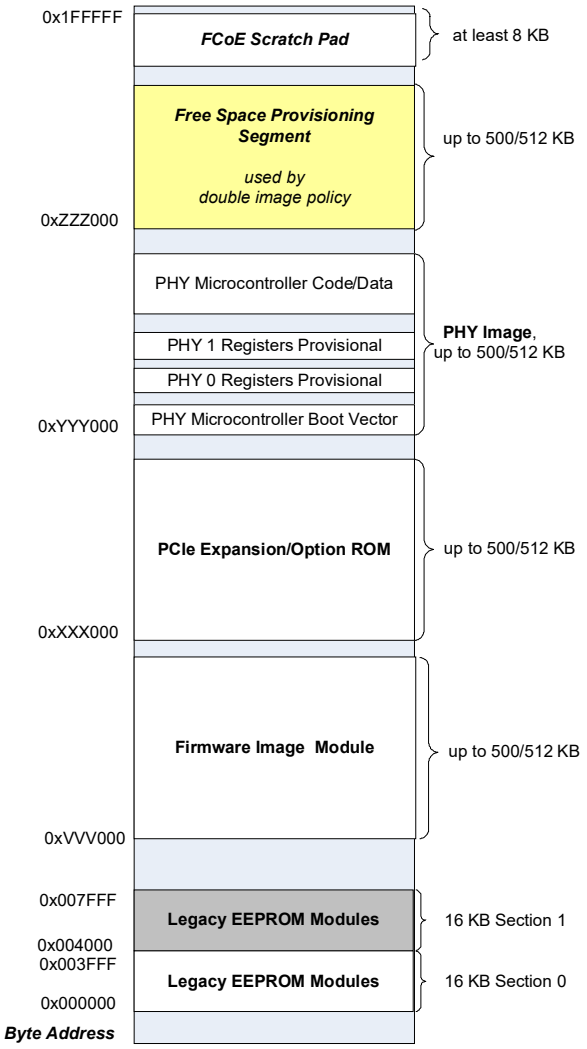


Figure 6-1. NVM Structure

6.1.1 Protected Areas

The following areas are protected from host writes:

- The Firmware Code Area
- The PHY code Area
- The PCI configuration areas (PHY, Link)
- The pointers to the different modules.
- The mini loader
- NVM Control Words 1/2/4

The list of words and modules that are read-only to the software is listed in Table 6-1.

The Firmware, PHY code areas, and Option ROM can be updated using the flow described in Section 3.4.8.1.

Read/Write words can be updated using the flow described in Section 3.4.7.6 and Section 3.4.2.1.

6.1.2 NVM Header

Note: **Intel configures the reserved NVM fields, and they are not intended to be changed beyond the default image provided by Intel.**

Table 6-1 lists the fixed part of the Legacy EEPROM Modules used by the X550. This table lists common modules for the NVM including: hardware pointers, software and firmware. Blocks pointed in this section are detailed in the following sections. All addresses in this table are absolute in word units.

Pointers may be in word units or in 4 KB units. If bit 15 of the pointer is set, bits [14:0] point to a 4 KB sector, and the word address is {pointer[14:0], 000,0000,0000b}. If bit 15 is cleared, the word address is pointer[14:0].

Table 6-1. Legacy EEPROM Modules Description

Word Address	Used By	Field Name	LAN 0/1	RO to Host
0x00	HW	NVM Control Word 1	Shared Logic	RO Word
0x01	HW	NVM Control Word 2	Shared Logic	RO Word
0x02	HW	PCIe Analog Module Pointer	Shared Logic	RO Pointer[1] RO Module
0x03	HW	PCIe Link Configuration Module Pointer	Shared Logic	RO Pointer RO Module[1]
0x04	HW	PHY Module Pointer	Shared Logic	RO Pointer[2] RO Module
0x05	HW	PCIe Expansion/Option ROM Pointer	SW + Shared Logic	RO Module[2]
0x06	HW	PCIe General Configuration Module Pointer	Shared Logic	
0x07	HW	PCIe Configuration Space 0 Module Pointer	Function 0	
0x08	HW	PCIe Configuration Space 1 Module Pointer	Function 1	
0x09	HW	LAN Core 0 Module Pointer	Port 0	
0x0A	HW	LAN Core 1 Module Pointer	Port 1	

Table 6-1. Legacy EEPROM Modules Description [continued]

Word Address	Used By	Field Name	LAN 0/1	RO to Host
0x0B	HW	MAC 0 Module Pointer	Port 0	RO Pointer RO Module[1]
0x0C	HW	MAC 1 Module Pointer	Port 1	RO Pointer RO Module[1]
0x0D	HW	CSR 0 Auto Configuration Module Pointer	Port 0	RO Pointer RO Module[1]
0x0E	HW	CSR 1 Auto Configuration Module Pointer	Port 1	RO Pointer RO Module[1]
0x0F	FW	Firmware Module Pointer	FW	
0x10 – 0x14	SW	SW Compatibility Module	SW	
0x15 – 0x16	SW	PBA Bytes 1...4	SW	
0x17	SW + FW	Boot Configuration Start Address	SW+FW	
0x18	SW + FW	NVM Dev Starter Version	SW+FW	
0x19	SW + FW	PHY Firmware Version	SW+FW	
0x1A – 0x26	SW	Software Reserved	SW	
0x27	SW	Alternate SAN MAC Address Pointer	SW	
0x28	SW	Active SAN MAC Address Pointer	SW	
0x29	SW	Map Version	SW	
0x2A	SW	Image Revision	SW	
0x2B	SW	Software Reserved	SW	
0x2C	SW	Platform/NIC/LOM Specific Capabilities	SW	
0x2D – 0x2E	SW	eTrack_ID number	SW	
0x2F	OEM	VPD Pointer	Shared Logic	RO Pointer[3] RO Section
0x30 – 0x36	PXE	PXE Configuration Words	SW	
0x37	SW + FW	Alternate Ethernet MAC Addresses Pointer	SW+FW	
0x38	HW	NVM Control Word 3	Shared Logic	
0x39	SW	FCoE Scratch Pad Structure Pointer	SW	
0x3A	FW	Firmware Code Pointer	FW	RO Pointer[2] RO Module
0x3B – 0x3E	HW	Hardware Reserved	Reserved	
0x3F	SW+FW	Software Checksum, Words 0x00 — 0x3F	Shared Logic	
0x40	FW	Free Space Provisioning Segment Pointer	FW	RO Pointer
0x41	FW	Free Provisioning Area Size, expressed in 4KB sectors. Default is 0x80 for 4MB flash and 0x71 for 2 MB flash.	FW	RO Word
0x42	FW	Mini Loader Pointer	FW	RO Pointer RO Module[1]
0x43	FW	PHY Configuration	FW	RO Pointer RW Module

Table 6-1. Legacy EEPROM Modules Description [continued]

Word Address	Used By	Field Name	LAN 0/1	RO to Host
0x44 – 0x4F		Reserved		
0x50	FW	RO Updates Version - This field is copied by firmware from the RO *Updates Version* field present in the trailer of the firmware secured module. Refer to Section 6.1.4.2.	FW	RO Word

1. Updated via RO words mechanism.
2. Updated via the secured module update flow (Section 3.4.8.1).
3. VPD area is updated via the VPD PCIe capability or via Shadow RAM Update host I/F command if *VPD Write Enable* bit in *NVM Control Word 1* is set.

All pointers refer to word addresses, except pointers to PCIe Expansion/Option ROM (0x5), PHY Image (0x4), Firmware Code (0x3A) and Free Space Provisioning Segment (0x41) which are expressed in 4 KB sector units.

6.1.3 Hardware Sections

This module contains address control words and hardware pointers indicated as hardware in Table 6-1. The process of loading this module (or any of it sub-modules) into the device is referred as MAC auto-load process. These modules must be mapped in the first valid 16 KB section of the Flash.

6.1.3.1 Hardware Section — Auto-Load Sequence

The following table lists sections of auto-read following device reset events or specific commands from registers. Auto-read is performed from the internal Shadow RAM (or from internal memory for PHY module) and not from the NVM device, except following LAN_PWR_GOOD or Reset Pin events.

Table 6-2. NVM Section Auto-Read

	LAN_PWR_GOOD or Reset Pin	PCIe Reset or PCIe Inband Reset	D3 to D0 Transit (per port)	FLR (per port)	SW Reset (per port)	Link Reset (per port)	FW Reset	Force TCO	PHY Image Re-Load Command[1]
PCIe Analog Configuration	X[2]								
PCI Link Configuration (LCB)		X							
PCIe General Configuration		X							
PCIe Function 0/1 Config Space (for each function)		X							
LAN Core and CSRs (for each LAN port)	X	X	X	X	X	X		X	
PHY Module (for each LAN port)	X								X
Manageability Firmware Module	X						X		

1. via MDIO (per port) PHY register bit 1E.C442.0
2. This is the unique module that requires power-up to be reloaded.

6.1.4 Firmware Image Module

The firmware image module contains the main code of the internal firmware loaded to the internal RAM. This module includes a header and a trailer. The header is used when the firmware is updated to authenticate it. The optional trailer includes a list of Read only words to be modified by the firmware or a new version of the first 16 KB of the NVM.

6.1.4.1 Header of Firmware Image Module

Since authenticated modules are by-definition not modifiable *in the fields*, no "holes" are present in such modules.

Table 6-3 shows the header of the Firmware Image module. Fields colored in light blue are protected by the authentication signature.

Table 6-3. Header of Firmware and Option ROM Images

Byte Offset	Number of Words	Field or Segment Name	Description and Comments
0x00 – 0x7F	64	Authentication Header	
0x80 – 0x17F	128	RSA Public Key	This field is skipped for the sake of SHA256 Hash computing.
0x180 – 0x183	2	RSA Exponent	This field is skipped for the sake of SHA256 Hash computing.
0x184 – 0x283	128	Encrypted SHA256 Hash	This field is skipped for the sake of SHA256 Hash computing.
0x284 – 0x285	1	X550 Blank NVM Device ID	A unique ND-provided device ID that identifies the X550 controller among other ND controllers. It must be set to 0x1562 in the X550.
0x286 – 0x289	2	Max Module Area	It is the maximum flash area expressed in words that can be used by the module, starting from authentication header (included). It is set to 256 K words (i.e. 512 KB) for all modules.
0x28A – 0x28D	2	Current Module Area	It is the flash area expressed in words that is currently used by the module, starting from authentication header (included).
0x28E – 0x28F	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. Bits[14:8] = Module format version. Set to 0x02 is this currently defined format is used. Bits[7:0] = CRC8 value computed over the parent module only, starting from authentication header (included) and including all other module's fields and segments – excluding all descendant modules and *RSA Public Key/RSA Exponent* fields. CRC8 field is skipped for the sake of CRC8 computing (when bit 15 is set to 0b).
0x290 – 0x291	1	Code Revision	Bits[15:8] = Major revision number. Bits[7:0] = Minor revision number.
0x292 – 0x293	1	Reserved Spare Word	Must be zeroed.
0x294 – 0x297	2	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. It must be set to N. Modules read by firmware are NOT size limited to 128 KB. It excludes the last 16 KB sector of the firmware image which is reserved for the RO Commands section. This field is not relevant for option ROM header and should be zeroed.

Table 6-3. Header of Firmware and Option ROM Images [continued]

Byte Offset	Number of Words	Field or Segment Name	Description and Comments
...	N	parent word 1	Format of the contents of firmware modules is specific to each module. This field is not relevant for option ROM header and should be zeroed.
		parent word 2	
		...	
		parent word N	

6.1.4.2 Trailer of the Firmware Image Module

Table 6-4 shows the format of the last 16 KB sector of the firmware image. Fields colored in light blue are protected by the authentication signature.

Table 6-4. 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. A null field here indicates that the next words up to the 16 KB sector's end contain the new Shadow RAM contents, starting from word 0x000 up to word 0x0x1FF7 included.
1	X550 Blank NVM Device ID	A unique ND-provided device ID that identifies the X550 controller among other ND controllers. It must be set to 0x1562 in the X550.
1	Minimum Firmware Code Revision	Minimum Firmware Code Revision number required for being able to parse the RO Commands section. It must be less than or equal to the Code Revision number read from the Firmware Image Header listed in Table 6-3.
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 6.1.4.2.1.
	RO Commands word 2	
	...	
	RO Commands word N	

6.1.4.2.1 Format of the RO Commands

The RO Commands words can contains the following three structures:

- Shadow RAM Word Write Command (2 words)
- CSR Write Command (4 words)
- Shadow RAM contents, from word 0x0 to 0x1FF7 included

The first two structures starts with a type field.

Table 6-5 describes the different commands types:

Table 6-5. RO Commands Types

Type	Description
xxx1b	Word auto-load.
0100b	CSR auto-load with port number.
Other	Invalid type, parsing is stopped here.

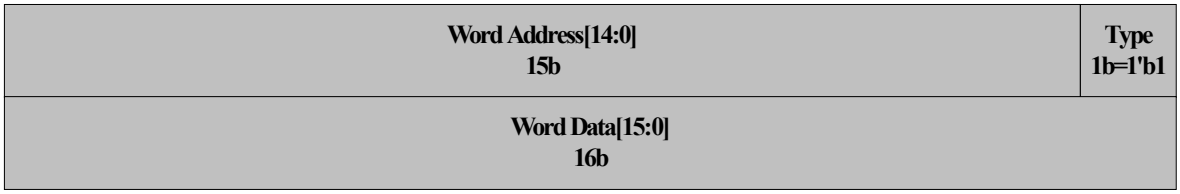


Figure 6-2. Shadow RAM Word Write Command

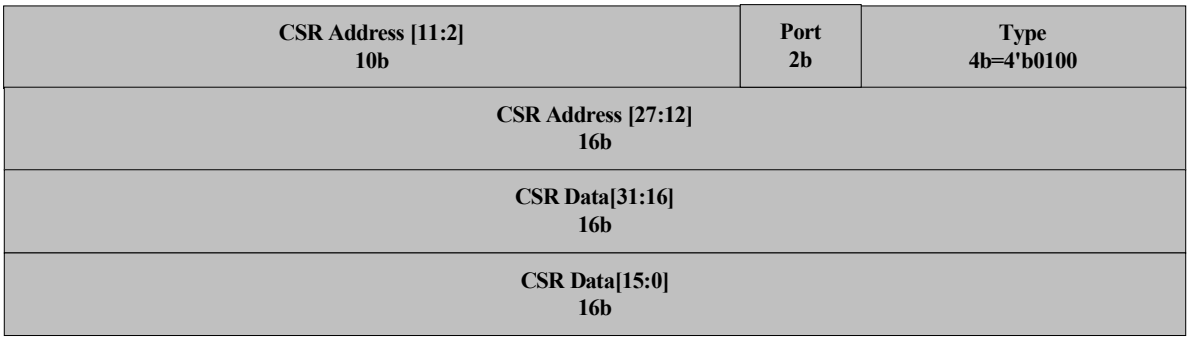


Figure 6-3. CSR Write Command (with Port Number)

If the *Port* value is invalid, the command is ignored.

6.1.5 PCIe Expansion/Option ROM

This module might include the PXE driver, iSCSI boot and/or FCoE boot image, UEFI network driver, and a Command Line Protocol (CLP) module. It is made of a single combo module (no pointers to sub-sections) and must fit into 512 KB.

It is not required for LOM systems where it may be stored on the BIOS Flash device.

The module is pointed by the PCIe expansion/option ROM pointer at NVM word 0x05, expressed in 4 KB sector units. Whenever modifying this pointer in the NVM, it is required to issue a PCIe reset before any new access is performed to the Expansion ROM. Otherwise the X550 would continue to use the old pointer each time it internally maps accesses to the expansion ROM. Refer to Section 3.4.3.1.

The first 330 words listed in Table 6-3 (up to the *Reserved spare word*) are mapped at the end of the area allocated to the module (a trailer), though for the sake of the authentication, these words are mapped at the module's header, as listed in the table.

6.1.6 PHY Module

Refer to Section 3.7.3.3 for the way this module is used by the X550.

The module is pointed by the PHY module pointer at NVM word address 0x4 (or word address 0x2004), expressed in 4 KB sector units. Before performing an auto-load, the PHY reads this pointer directly from the NVM.

The PHY module is composed of the following three sections as shown in Figure 6-4:

- **Header** — Provides configuration information for the SPI interface and pointers to the Instruction RAM and Data RAM locations. (See Table 6-6 for detailed description of the header content).
- **Instruction RAM** — This section of the image is loaded into the instructional SRAM (ISRAM).
- **Data RAM** — This section of the image is loaded into the data SRAM (DSRAM).

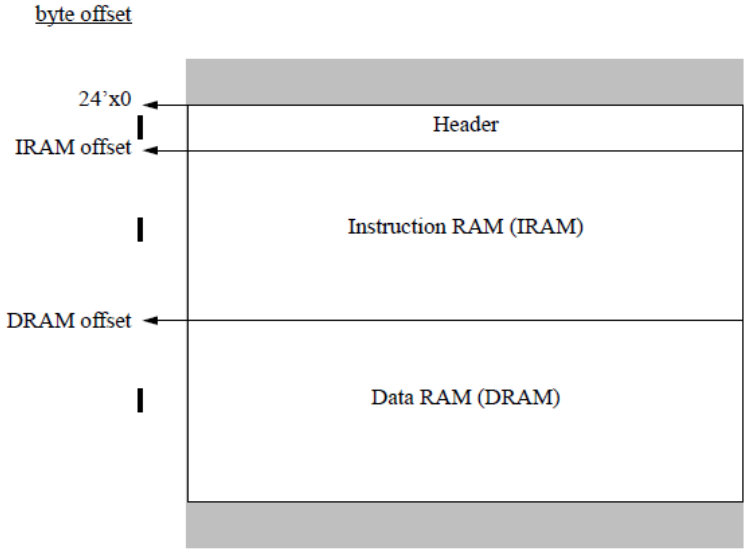


Figure 6-4. PHY Module Map

Table 6-6. PHY Module Header

Byte Offset	Description
0x00 – 0x7F	Authentication Header.
0x80 – 0x17F	RSA Public Key — This field is skipped for the sake of SHA256 Hash computing.
0x180 – 0x183	RSA Exponent — This field is skipped for the sake of SHA256 Hash computing.
0x184 – 0x283	Encrypted SHA256 Hash — This field is skipped for the sake of SHA256 Hash computing.
0x284 – 0x285	X550 Blank NVM Device ID — A unique ND-provided device ID that identifies the X550 controller among other ND controllers. It must be set to 0x1562 in the X550.
0x286 – 0x289	Max Module Area — It is the maximum flash area expressed in words that can be used by the module, starting from authentication header (included). It is set to 256 K words (i.e. 512 KB) for all modules.
0x28A – 0x28D	Current Module Area — It is the flash area expressed in words that is currently used by the module, starting from authentication header (included).

Table 6-6. PHY Module Header [continued]

Byte Offset	Description
0x28E – 0x28F	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. Bits[14:8] = Module format version. Set to 0x02 is this currently defined format is used. Bits[7:0] = CRC8 value computed over the parent module only, starting from authentication header (included) and including all other module's fields and segments - excluding all descendant modules and *RSA Public Key/RSA Exponent* fields. CRC8 field is skipped for the sake of CRC8 computing (when bit 15 is set to 0b).
0x290 – 0x291	Code Revision: Bits[15:8] = Major revision number. Bits[7:0] = Minor revision number.
0x292 – 0x2FF	Reserved Spare Words — Must be set to zero.
0x300	SPI Clock Divider[7:0]
0x301	Daisy Chain Clock Divider[7:0]
0x302	Wait Timer[7:0]
0x303	Wait Timer[15:8]
0x304	MCP IRAM Head Pointer[7:0]
0x305	MCP IRAM Head Pointer[15:8]
0x306	MCP IRAM Head Pointer[23:16]
0x307	MCP IRAM Byte Length[7:0]
0x308	MCP IRAM Byte Length[15:8]
0x309	MCP IRAM Byte Length[23:16]
0x30A	MCP DRAM Head Pointer[7:0]
0x30B	MCP DRAM Head Pointer[15:8]
0x30C	MCP DRAM Head Pointer[23:16]
0x30D	MCP DRAM Byte Length[7:0]
0x30E	MCP DRAM Byte Length[15:8]
0x30F	MCP DRAM Byte Length[23:16]
0x310	{Timeout Configuration[3:0], Reserved[2:0], Enabled CRC}
0x311	Reserved
0x312	Reserved
0x313	Reserved
0x314	Reserved
0x315	Reserved
0x316	Expected CRC16[7:0][1]
0x317	Expected CRC16[15:8]

1. Calculated from 0x300 to the end of the PHY code.

The PHY module is organized as follows, where offsets are expressed in bytes from the beginning of the module:

- 0x00000000 – 0x00000317: Header (see Table 6-6).
- 0x00000318 – 0x000004BF: Reset Vector (processor jumps here on reset).

  This small assembly language routine sets up basic processor functions and loads the IRAM and DRAM memories. When memories are initialized, they jump to the start up code in the IRAM.
- 0x000004C0 – 0x000005BF: Register Provisioning Table.

  One of the first things that's done at reset is to override the PIF registers with registers from these tables. A utility is available that manages this table called provision.

  Defaults to PHY interrupts mask registers must always be provisioned here because they impact the PHY's operation with the MC before the operating system is up.
- 0x000005C0 – 0x000005FF: Version String (62 bytes), Firmware major/minor revision numbers (2 bytes).
- 0x00000600 – Current Module Area: IRAM/DRAM load tables and contents.

  The boot code uses the load table and data in this region to initialize IRAM and DRAM.

6.1.7 Register Provisional Table

The PHY registers provisioning table consists of a variable number of change lists, the minimum number of change list is zero, and the maximum number is limited only by the size of the table, 256 bytes shared by both ports.

There is a master provisioning list that applies to both PHYs in addition to separate PHY specific changes lists (to reduce the table size).

Each change list consists of a 4-byte header and at least one change specification. The header consists of a one byte PHY identifier, a one byte MMD device address, and a two bytes count of change specifications to follow. Master provisioning is identified by a 0xFF PHY identifier.

Each change specification consists of a two byte register address within the MMD and a two byte value. Only fields marked with a PD Type in Chapter 10 are affected.

The following content is expected in the provisioning area:

Register	Value
1E.D400	0x40
1E.D402	0x501
1E.FF00	0x1
1E.FF01	0x5

6.2 NVM Content

6.2.1 NVM General Summary Table

Table 6-7. NVM General Summary Table

NVM Section	Section Number
Init Module Section	6.2.2
MAC Module Section	6.2.3
CSR Auto Config Section	6.2.4
LAN Core Module Section	6.2.5
PCIe General Configuration Module Section	6.2.6
PCIe Configuration Space Section	6.2.7
Alternate Ethernet MAC Address	6.2.8
FCoE Scratch Pad Header Section	6.2.9
Active SAN MAC Address Section	6.2.10
Alternate SAN MAC Address Section	6.2.11
Boot Configuration Block Section	6.2.12
Firmware Module Header Section	6.2.13
Firmware Header Reserved Word Section	6.2.14
Test Configuration Module Section	6.2.15
Common Firmware Parameters Module Section	6.2.16
Sideband Configuration Structure Section	6.2.17
Pass-Through Control Words Section	6.2.18
Flexible TCO Filter Configuration Structure Section	6.2.19
LESM Configurations (not in SGVL) Section	6.2.20
PXE VLAN Configuration Section	6.2.21
VPD Module Section	6.2.22
PBA Number Module Section	6.2.23
Mini Loader Module Section	6.2.24
PHY Config Section	6.2.25
PCIe Link (LCB) Configuration Section	6.2.26
PCIe Analog Configuration Module Section	6.2.27
2'nd Init Module Section	6.2.28
FCoE Scratch Pad Section	6.2.29
Firmware Module Section	6.2.30
PXE/OROM Module Section	6.2.31
AQ PHY Module Section	6.2.32
Free Provisioning Module Section	6.2.33

6.2.2 Init Module Section

Table 6-8. Init Module Section Summary Table

Word Address	Used By	Word Name	Section Number
0x0000	HW	NVM Control Word 1	6.2.2.1
0x0001	HW	NVM Control Word 2	6.2.2.2
0x0002	HW	PCIe Analog Configuration Module Pointer	6.2.2.3
0x0003	HW	PCIe Link (LCB) Configuration Pointer	6.2.2.4
0x0004	HW	PHY Module Pointer	6.2.2.5
0x0005	HW	PCIe Expansion/Option ROM Pointer	6.2.2.6
0x0006	HW	PCIe General Configuration Module Pointer	6.2.2.7
0x0007	HW	PCIe Configuration Space 0 Module Pointer	6.2.2.8
0x0008	HW	PCIe Configuration Space 1 Module Pointer	6.2.2.9
0x0009	HW	LAN Core 0 Module Pointer	6.2.2.10
0x000A	HW	LAN Core 1 Module Pointer	6.2.2.11
0x000B	HW	MAC 0 Module Pointer	6.2.2.12
0x000C	HW	MAC 1 Module Pointer	6.2.2.13
0x000D	HW	CSR 0 Auto Configuration Module Pointer	6.2.2.14
0x000E	HW	CSR 1 Auto Configuration Module Pointer	6.2.2.15
0x000F	FW	Firmware Module Pointer	6.2.2.16
0x0010	SW	SW Compatibility Word 1	6.2.2.17
0x0011	SW	SW Compatibility Word 2	6.2.2.18
0x0012	SW	SW Compatibility Word 3	6.2.2.19
0x0013	SW	SW Compatibility Word 4	6.2.2.20
0x0014	SW	SW Compatibility Word 5	6.2.2.21
0x0015	SW	PBA Word 1	6.2.2.22
0x0016	SW	PBA Word 2	6.2.2.23
0x0017	SW	Boot Configuration Start Address	6.2.2.24
0x0018	SW	Software Reserved Word 1 - Dev Starter Version	6.2.2.25
0x0019	SW	Software Reserved Word 2 - PHY Image Revision	6.2.2.26
0x001A	SW	Software Reserved Word 3	6.2.2.27
0x001B	SW	Software Reserved Word 4	6.2.2.28
0x001C	SW	Software Reserved Word 5	6.2.2.29
0x001D	SW	Software Reserved Word 6	6.2.2.30
0x001E	SW	Software Reserved Word 7	6.2.2.31
0x001F	SW	Software Reserved Word 8	6.2.2.32
0x0020	SW	Software Reserved Word 9 - PXE VLAN Config Pointer	6.2.2.33
0x0021	SW	Software Reserved Word 10	6.2.2.34
0x0022	SW	Software Reserved Word 11	6.2.2.35

Table 6-8. Init Module Section Summary Table [continued]

Word Address	Used By	Word Name	Section Number
0x0023	SW	Software Reserved Word 12	6.2.2.36
0x0024	SW	Software Reserved Word 13	6.2.2.37
0x0025	SW	Software Reserved Word 14 - Original EETrack ID 1	6.2.2.38
0x0026	SW	Software Reserved Word 15 - Original EETrack ID 2	6.2.2.39
0x0027	SW	Software Reserved Word 16 - Alternate SAN MAC Address Pointer	6.2.2.40
0x0028	SW	Software Reserved Word 17 - Active SAN MAC Address Pointer	6.2.2.41
0x0029	SW	Software Reserved Word 18 - MAP Version	6.2.2.42
0x002A	SW	Software Reserved Word 19 - IMAGE Version	6.2.2.43
0x002B	SW	Software Reserved Word 20	6.2.2.44
0x002C	SW	Software Reserved Word 21 - FCoE Offload	6.2.2.45
0x002D	SW	Software Reserved Word 22 - EETRACK ID 1	6.2.2.46
0x002E	SW	Software Reserved Word 23 - EETRACK ID 2	6.2.2.47
0x002F	SW	VPD Module Pointer	6.2.2.48
0x0030	SW	PXE Setup Options PCI Function 0	6.2.2.49
0x0031	SW	PXE Configuration Customization Options PCI Function 0	6.2.2.50
0x0032	SW	PXE Version	6.2.2.51
0x0033	SW	Flash Capabilities	6.2.2.52
0x0034	SW	PXE Setup Options PCI Function 1	6.2.2.53
0x0035	SW	PXE Configuration Customization Options PCI Function 1	6.2.2.54
0x0036	SW	iSCSI Option ROM Version	6.2.2.55
0x0037	SW	Alternate Ethernet MAC Addresses Pointer	6.2.2.56
0x0038	SW	NVM Control Word 3	6.2.2.57
0x0039	HW	FCoE Scratch Pad Pointer	6.2.2.58
0x003A	FW	Firmware Code Pointer	6.2.2.59
0x003B	HW	Hardware	6.2.2.60
0x003C	HW	Hardware	6.2.2.61
0x003D	HW	Hardware	6.2.2.62
0x003E	HW	Hardware	6.2.2.63
0x003F	SW	Software Checksum, Words 0x00 - 0x3F	6.2.2.64
0x0040	HW	Free Provisioning Area Pointer	6.2.2.65
0x0041	HW	Free Provisioning Area Size	6.2.2.66
0x0042	HW	Mini Loader Pointer	6.2.2.67
0x0043	HW	PHY Config Pointer	6.2.2.68
0x0044	HW	Reserved	6.2.2.69
0x0045	N/A	Reserved	6.2.2.70
0x0046	HW	Reserved	6.2.2.71
0x0047	HW	Reserved	6.2.2.72

Table 6-8. Init Module Section Summary Table [continued]

Word Address	Used By	Word Name	Section Number
0x0048	HW	Reserved	6.2.2.73
0x0049	HW	Reserved	6.2.2.74
0x004A	HW	Reserved	6.2.2.75
0x004B	HW	Reserved	6.2.2.76
0x004C	HW	Reserved	6.2.2.77
0x004D	HW	Reserved	6.2.2.78
0x004E	HW	Reserved	6.2.2.79
0x004F	HW	Reserved	6.2.2.80
0x0050	HW	RO Updates Version	6.2.2.81

6.2.2.1 NVM Control Word 1 (0x0000)

Bit(s)	Field Name	Default	Description
15	Reserved	0b	Reserved.
14	NC-SI Ext HW ARB Enable	0b	**NC-SI Hardware Arbitration Enable** 0b = NCSI_ARB_IN and NCSI_ARB_OUT pads are not used. NCSI_ARB_IN is pulled up internally to provide stable input. 1b = NCSI_ARB_IN and NCSI_ARB_OUT pads are used.
13	Reserved	0b	Reserved.
12	NVM Host Debug Mode	1b	**NVM Host Debug Mode** 0b = Normal operating mode (default). 1b = Firmware reset has an effect on software write access to NVM via EEPROM-mode although it should not have any effect on it.
11	Reserved	0b	Reserved.
10:8	NVM Size	0x5	**NVM Size** These bits indicate the Flash's actual size. Mapped to FLA.*FL_SISE* (see field definition in the FLA register section). 0x4 = 1 MB 0x5 = 2 MB 0x6 = 4 MB All other values are reserved.
7:6	Signature	01b	**Signature** The *Signature* field indicates 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. 00b = NVM not present 0. 01b = NVM present. 10b = NVM not present 2. 11b = NVM not present 3.
5:0	Reserved	0x0	Reserved.

6.2.2.2 NVM Control Word 2 (0x0001)

Bit(s)	Field Name	Default	Description
15	PCI LAN Function Sel -- Swap	0b	**PCI LAN Function Sel -- Swap** When both LAN ports are enabled and the *LAN Function Sel* equals 0b, LAN 0 is routed to PCI function 0 and LAN 1 is routed to PCI function 1. If the *LAN Function Sel* equals 1b, LAN 0 is routed to PCI function 1 and LAN 1 is routed to PCI function 0. This bit is loaded from NVM. ***Note:*** This field is preserved by Intel NVM update tool.
14	Keep PHY CB	0b	**Keep PHY CB** If set, the Keep_phy_in_dx aux bit affects the PHY state only while in D3/DR (legacy behavior). Otherwise, it affects the PHY state in D0 also.
13:12	Reserved	00b	Reserved.
11	PCI Function Off via SDPins Enable	0b	**PCIe Function Off via SDP Pins Enable.** 0b = Legacy mode (default), SDPn_1 pins does not control PCIe functions off. 1b = SDPn_1 pins are strapped (sampled by PE_RST_N) to disable the corresponding PCIe function. ***Note:*** If manageability is present, MC-to-LAN paths are not disabled. Readable as DEV_FUNC_EN[3] ***Note:*** This field is preserved by Intel NVM update tool.
10	Reserved	0b	Reserved.
9	LAN PCI Disable Select	0b	**LAN PCI Disable Select** Selects which port is disabled. ***Note:*** This field is preserved by Intel NVM update tool.
8	LAN PCI Disable	0b	**LAN PCI Disable** When set one PCI LAN port is disabled. ***Note:*** This field is preserved by Intel NVM update tool.
7:4	Reserved	0x0	Reserved.
3	Deadlock Timeout Enable	1b	**Deadlock Timeout Enable** If set, a device that was granted access to the NVM that does not toggle the interface for 2 seconds will have the grant revoked. 0b = Timeout disabled. 1b = Timeout enabled. ***Note:*** This field is preserved by Intel NVM update tool.
2	Reserved	1b	Reserved.
1	Core Clocks Gate Disable	0b	**Core Clocks Gate Disable** During nominal operation, this bit should be zero enabling core clock gating in low power state. When set, disables the gating of the core clock in low power state. 0b = Enable Enables gating of the core clock in low power state. 1b = Disable Disables gating of the core clock in low power state ***Note:*** This field is preserved by Intel NVM update tool.
0	Reserved	0b	Reserved.

6.2.2.3 PCIe Analog Configuration Module Pointer (0x0002)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	PCIe Analog Configuration Module Pointer	0xFFFF	**PCIe Analog Configuration Module Pointer** Points to PCIe Analog Configuration Module Section. For PCIe Analog Configuration Module inner structure, see Section 6.2.27.

6.2.2.4 PCIe Link (LCB) Configuration Pointer (0x0003)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	PCIe Link (LCB) Configuration Pointer	0x0	**PCIe Link (LCB) Configuration Pointer** Points to PCIe Link (LCB) Configuration Section. For PCIe Link (LCB) Configuration inner structure, see Section 6.2.26.

6.2.2.5 PHY Module Pointer (0x0004)

The address is in 4 KB sector index units.

Bit(s)	Field Name	Default	Description
15	Pointer Type	1b	**Pointer Type** 0b = Word units. 1b = 4 KB sector units.
14:0	PHY Module Pointer	0x0	**PHY Module Pointer** Points to AQ PHY Module Section. For AQ PHY Module inner structure, see Section 6.2.32.

6.2.2.6 PCIe Expansion/Option ROM Pointer (0x0005)

The address is in 4 KB sector index units.

Bit(s)	Field Name	Default	Description
15	Pointer Type	1b	**Pointer Type** 0b = Word units. 1b = 4 KB sector units.
14:0	PCIe Expansion/ Option ROM Pointer	0x7FFF	**PCIe Expansion/Option ROM Pointer** Points to PXE/OROM Module Section. For PXE/OROM Module inner structure, see Section 6.2.31.

6.2.2.7 PCIe General Configuration Module Pointer (0x0006)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	PCIe General Configuration Pointer	0xFFFF	**PCIe General Configuration Pointer** Points to PCIe General Configuration Module Section. For PCIe General Configuration Module inner structure, see Section 6.2.6.

6.2.2.8 PCIe Configuration Space 0 Module Pointer (0x0007)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	PCIe Configuration Space 0 Module Pointer	0xFFFF	**PCIe Configuration Space 0 Module Pointer** Points to PCIe Configuration Space Section. For PCIe Configuration Space inner structure, see Section 6.2.7.

6.2.2.9 PCIe Configuration Space 1 Module Pointer (0x0008)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	PCIe Configuration Space 1 Module Pointer	0xFFFF	**PCIe Configuration Space 1 Module Pointer** Points to PCIe Configuration Space Section. For PCIe Configuration Space inner structure, see Section 6.2.7.

6.2.2.10 LAN Core 0 Module Pointer (0x0009)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	LAN Core 0 Section Pointer	0xFFFF	**LAN Core 0 Section Pointer** Points to LAN Core Module Section. For LAN Core Module inner structure, see Section 6.2.5.

6.2.2.11 LAN Core 1 Module Pointer (0x000A)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	LAN Core 1 Section Pointer	0xFFFF	**LAN Core 1 Section Pointer** Points to LAN Core Module Section. For LAN Core Module inner structure, see Section 6.2.5.

6.2.2.12 MAC 0 Module Pointer (0x000B)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	MAC 0 Section Pointer	0xFFFF	**MAC 0 Section Pointer** Points to MAC Module Section. For MAC Module inner structure, see Section 6.2.3.

6.2.2.13 MAC 1 Module Pointer (0x000C)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	MAC 1 Section Pointer	0xFFFF	**MAC 1 Section Pointer** Points to MAC Module Section. For MAC Module inner structure, see Section 6.2.3.

6.2.2.14 CSR 0 Auto Configuration Module Pointer (0x000D)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	CSR 0 Auto Configuration Pointer	0xFFFF	**CSR 0 Auto Configuration Pointer** Points to CSR Auto Config Section. For CSR Auto Config inner structure, see Section 6.2.4.

6.2.2.15 CSR 1 Auto Configuration Module Pointer (0x000E)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	CSR 1 Auto Configuration Pointer	0xFFFF	**CSR 1 Auto Configuration Pointer** Points to CSR Auto Config Section. For CSR Auto Config inner structure, see Section 6.2.4.

6.2.2.16 Firmware Module Pointer (0x000F)

The address is in words.

Bit(s)	Field Name	Default	Description
15:0	Firmware Section Pointer	0x0	**Firmware Section Pointer** Points to Firmware Module Header Section. For Firmware Module Header inner structure, see Section 6.2.13.

6.2.2.17 SW Compatibility Word 1 (0x0010)

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
11	LOM	0b	**LOM** Indicates whether the NVM attached to LAN silicon contains dedicated module for option ROM. Used by option ROM update applications. 0b = NIC (Attached flash contains module for option ROM). 1b = LOM (Attached flash has no module for option ROM). ***Note:*** This field is preserved by Intel NVM update tool.
10	Server	1b	**Server** Legacy, not currently used. 0b = Client 1b = Server ***Note:*** This field is preserved by Intel NVM update tool.
9	Reserved	0b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
8	OEM/Retail	0b	**OEM/Retail** Legacy, not currently used. 0b = Retail 1b = OEM ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Reserved	0x0	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.18 SW Compatibility Word 2 (0x0011)

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.19 SW Compatibility Word 3 (0x0012)

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.20 SW Compatibility Word 4 (0x0013)

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.21 SW Compatibility Word 5 (0x0014)

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.22 PBA Word 1 (0x0015)

The nine-digit Printed Board Assembly (PBA) number used for Intel manufactured Network Interface Cards (NICs) is stored in the NVM.

Note: Through the course of hardware ECOs, the suffix field is incremented. The purpose of this information is to enable customer support (or any user) to identify the revision level of a product.

Network driver software should not rely on this field to identify the product or its capabilities.

Current PBA numbers have exceeded the length that can be stored as hex values in these two words. For these PBA numbers the high word is a flag (0xFAFA) indicating that the PBA is stored in a separate PBA block. The low word is a pointer to a PBA block.

Bit(s)	Field Name	Default	Description
15:0	PBA Word 1	0x0	**PBA Word 1** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.23 PBA Word 2 (0x0016)

The nine-digit Printed Board Assembly (PBA) number used for Intel manufactured Network Interface Cards (NICs) is stored in the NVM.

Note: Through the course of hardware ECOs, the suffix field is incremented. The purpose of this information is to enable customer support (or any user) to identify the revision level of a product.

Network driver software should not rely on this field to identify the product or its capabilities.

Current PBA numbers have exceeded the length that can be stored as hex values in these two words. For these PBA numbers the high word is a flag (0xFAFA) indicating that the PBA is stored in a separate PBA block. The low word is a pointer to a PBA block.

Bit(s)	Field Name	Default	Description
15:0	PBA Word 2	0x0	**PBA Word 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.24 Boot Configuration Start Address (0x0017)

Address of iSCSI Boot configuration module. This is a word pointer. The block length is embedded in the module.

Bit(s)	Field Name	Default	Description
15:0	iSCSI Boot Configuration Start Address	0x0	**iSCSI Boot Configuration Start Address** This module is long by 1504B and must be mapped in the first valid 4 KB sector of the Flash. Points to Boot Configuration Block Section. For Boot Configuration Block inner structure, see Section 6.2.12.

6.2.2.25 Software Reserved Word 1 - Dev Starter Version (0x0018)

Copy of first contents of MAP Version, not auto generated, to trace source of any image.

Bit(s)	Field Name	Default	Description
15:12	Major Version	0x1	**Major Version** NVM minor version.
11:8	Reserved	0x0	Reserved. NVM major version.
7:0	Minor Version	0x92	**Minor Version** Human-readable, hex using only the decimal digits to make the digital version.

6.2.2.26 Software Reserved Word 2 - PHY Image Revision (0x0019)

Bit(s)	Field Name	Default	Description
15:12	Major	0x2	**Major** PHY major version.
11:4	Minor	0x0B	**Minor** PHY minor version.
3:0	ID	0xB	**ID** PHY Image ID.

6.2.2.27 Software Reserved Word 3 (0x001A)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.28 Software Reserved Word 4 (0x001B)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.29 Software Reserved Word 5 (0x001C)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.30 Software Reserved Word 6 (0x001D)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.31 Software Reserved Word 7 (0x001E)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.32 Software Reserved Word 8 (0x001F)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.33 Software Reserved Word 9 - PXE VLAN Config Pointer (0x0020)

Bit(s)	Field Name	Default	Description
15:0	PXE VLAN Config Pointer	0xFFFF	Reserved. Points to PXE VLAN Configuration Section. For PXE VLAN Configuration inner structure, Section 6.2.21.

6.2.2.34 Software Reserved Word 10 (0x0021)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.35 Software Reserved Word 11 (0x002)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.36 Software Reserved Word 12 (0x0023)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.37 Software Reserved Word 13 (0x0024)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.38 Software Reserved Word 14 - Original EETrack ID 1 (0x0025)

Bit(s)	Field Name	Default	Description
15:0	Original EETrack ID 1	0xFFFF	Reserved.

6.2.2.39 Software Reserved Word 15 - Original EETrack ID 2 (0x0026)

Bit(s)	Field Name	Default	Description
15:0	Original EETrack ID 2	0xFFFF	Reserved.

6.2.2.40 Software Reserved Word 16 - Alternate SAN MAC Address Pointer (0x0027)

Word 0x27 points to the Alternate SAN MAC Address block used for FCoE (SPMA and FPMA) and DCB.

Bit(s)	Field Name	Default	Description
15:0	Alternate SAN MAC Address Pointer	0xFFFF	**Alternate SAN MAC Address Pointer** Pointer to the Permanent SAN MAC Address block. 0xFFFF indicates an invalid pointer. Points to Alternate SAN MAC Address Section. For Alternate SAN MAC Address inner structure, see Section 6.2.11.

6.2.2.41 Software Reserved Word 17 - Active SAN MAC Address Pointer (0x0028)

Word 0x28 points to the Permanent SAN MAC Address block used for FCoE (SPMA and FPMA) and DCB.

Bit(s)	Field Name	Default	Description
15:0	SAN MAC Address Pointer	0xFFFF	**SAN MAC Address Pointer** Pointer to the Permanent SAN MAC Address block. 0xFFFF indicates an invalid pointer. Points to Active SAN MAC Address Section. For Active SAN MAC Address inner structure, see Section 6.2.10.

6.2.2.42 Software Reserved Word 18 - MAP Version (0x0029)

Bit(s)	Field Name	Default	Description
15:0	MAP_Version	0xFFFF	**MAP Version** Auto-generated from same input as filename.

6.2.2.43 Software Reserved Word 19 - IMAGE Version (0x002A)

Bit(s)	Field Name	Default	Description
15:0	IMAGE _Version	0xFFFF	**IMAGE Version** Auto-generated from same input as filename

6.2.2.44 Software Reserved Word 20 (0x002B)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.45 Software Reserved Word 21 - FCoE Offload (0x002C)

This word is for platform/NIC/LOM specific settings.

Bit(s)	Field Name	Default	Description
15:7	Reserved	0x1FF	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
6	UEFI Port Number Display	1b	**UEFI Port Number Display** 0b = Enabled 1b = Disabled ***Note:*** This field is preserved by Intel NVM update tool.
5	HP OCD Support	1b	**HP OCD Support** 0b = Enabled 1b = Disabled ***Note:*** This field is preserved by Intel NVM update tool.
4	Lenovo Agentless VPD	1b	**Lenovo Agentless VPD** 0b = Enabled 1b = Disabled ***Note:*** This field is preserved by Intel NVM update tool.
3:2	Wake On LAN Support	11b	**Wake On LAN Support** This bit indicates to software if the device supports Wake on LAN. 00b = Reserved (not supported). 01b = WoL supported on both ports. 10b = WoL supported on port A only. 11b = WoL not supported on either port. ***Note:*** This field is preserved by Intel NVM update tool.
1	FCoE Offload	0b	**FCoE Offload** This bit indicates to software if the device supports FCoE Offload. 0b = FCoE Offload enabled. 1b = FCoE Offload disabled. ***Note:*** This field is preserved by Intel NVM update tool.
0	Reserved	1b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.46 Software Reserved Word 22 - EETRACK ID 1 (0x002D)

This word is for the first word of the eTrack_ID number written by EEPROM Manager Tool.

Bit(s)	Field Name	Default	Description
15:0	eTrack_ID Word 1	0xFFFFFFFF	**eTrack_ID Word 1** EEPROM Manager Tool writes a unique 32-bit eTrack_ID number in two sequential NVM words (0x002D-0x002E). The eTrack_ID is written when EEPROM Manager Tool creates an image on the Intel network. The eTrack_ID DB tracks NVM images back to a specific SCM build.

6.2.2.47 Software Reserved Word 23 - EETRACK ID 2 (0x002E)

This word is for the second word of the eTrack_ID number written by EEPROM Manager Tool.

Bit(s)	Field Name	Default	Description
15:0	eTrack_ID Word 2		**eTrack_ID Word 2**

6.2.2.48 VPD Module Pointer (0x002F)

Word pointer to Vital Product Data module. Block length is embedded in the module.

Bit(s)	Field Name	Default	Description
15:0	VPD Pointer	0xFFFF	**VPD Pointer** Vital Product Data Pointer. 0xFFFF Default unless VPD relative section is specified. The VPD section size is usually 64 words and is initialized to 0 or 0xFFFF. 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 4KB sector of the Flash. Points to VPD Module Section. For VPD Module inner structure, see Section 6.2.22.

6.2.2.49 PXE Setup Options PCI Function 0 (0x0030)

The main setup options for Port 0 are stored in this word. These options are those that can be changed by the user using the Control-S setup menu.

Bit(s)	Field Name	Default	Description
15:13	Reserved	000b	Reserved. Must be 000b. ***Note:*** This field is preserved by Intel NVM update tool.
12:10	FSD	000b	**Force Speed and Duplex** Bits 12-10 control forcing speed and duplex during driver operation. 000b = Auto-negotiate 001b = 10 Mb/s half duplex 010b = 100 Mb/s half duplex 011b = Not valid (treated as 000b) 100b = Not valid (treated as 000b) 101b = 10 Mb/s full duplex 110b = 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. ***Note:*** This field is preserved by Intel NVM update tool.
9	LWS	0b	**Legacy OS Wakeup Support** For 82559-based adapters only. If set to 1b, the agent enables PME in the adapter's PCI configuration space during initialization. This allows remote wake up under legacy operating systems that do not normally support it. Note that enabling this makes the adapter technically non-compliant with the ACPI specification, which is why the default is disabled. Must be set to 0b for 1 GbE and 10 GbE adapters. 0b = Disabled (default) 1b = Enabled ***Note:*** This field is preserved by Intel NVM update tool.
8	DSM	1b	**Display Setup Message** If set to 1b, the "Press Control-S" message is displayed after the title message. Default value is 1b. ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
7:6	PT	00b	**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. ***Note:*** This field is preserved by Intel NVM update tool.
5	iSCSI Disable	0b	**iSCSI Disable** 0b = Enabled 1b = Disabled ***Note:*** This field is preserved by Intel NVM update tool.
4:3	DBS	00b	**Default Boot Selection** These bits select which device is the default boot device. These bits are only used if the agent detects that the BIOS does not support boot order selection or if the MODE field of word 31h is set to MODE_LEGACY. 00b = Network boot, then local boot (default) 01b = Local boot, then network boot 10b = Network boot only 11b = Local boot only ***Note:*** This field is preserved by Intel NVM update tool.
2:0	PS	000b	**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: 000b = PXE (default value). 001b = Boot Disabled 010b = iSCSI Primary 011b = iSCSI Secondary 100b = FCoE All other values are reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.50 PXE Configuration Customization Options PCI Function 0 (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.

Bit(s)	Field Name	Default	Description
15:14	Signature	01b	**Signature** Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software. ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
13:12	Reserved	00b	Reserved. Must be 0. ***Note:*** This field is preserved by Intel NVM update tool.
11	Continuous Retry	0b	**Continuous Retry** 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. 0b = Disable 1b = Enable ***Note:*** This field is preserved by Intel NVM update tool.
10:8	Operating Mode	000b	**Operating Mode** Selects the agent's boot order setup mode. This field changes the agent's default behavior 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. ***Note:*** This field is preserved by Intel NVM update tool.
7:6	Reserved	00b	Reserved. Must be 0. ***Note:*** This field is preserved by Intel NVM update tool.
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. 0b = Enable Flash Update 1b = Disable Flash Update ***Note:*** This field is preserved by Intel NVM update tool.
4	Disable Legacy OS Wakeup Menu	0b	**Disable Legacy Wakeup Support** If this bit is set to 1b, the user is not allowed to change the Legacy OS Wakeup Support menu option. Default value is 0b. 0b = Enable Legacy Wakeup Support 1b = Disable Legacy Wakeup Support ***Note:*** This field is preserved by Intel NVM update tool.
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. 0b = Enable 1b = Disable ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
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. 0b = Enable 1b = Disable ***Note:*** This field is preserved by Intel NVM update tool.
1	Disable Title Message Display	0b	**Disable Title Message** If this bit is set to 1b, the title message displaying the version of the Boot Agent is suppressed; the Control-S message is also suppressed. This is for OEMs who do not wish the boot agent to display any messages at system boot. Default value is 0b. 0b = Enable 1b = Disable ***Note:*** This field is preserved by Intel NVM update tool.
0	Disable 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 may only be changed via an external program. Default value is 0b. 0b = Enable 1b = Disable ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.51 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.

Bit(s)	Field Name	Default	Description
15:12	Major Version	0x2	**Major Version** PXE Boot Agent Major Version. ***Note:*** This field is preserved by Intel NVM update tool.
11:8	Minor Version	0x3	**Minor Version** PXE Boot Agent Minor Version. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Build Number	0x48	**Build Number** PXE Boot Agent Build Number. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.52 Flash Capabilities (0x0033)

Word 0x33h of the NVM is used to enumerate the boot technologies that have been programmed into the Flash. This is updated by Flash configuration tools and is not updated or read by IBA.

Bit(s)	Field Name	Default	Description
15:14	Signature	01b	**Signature** Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software. ***Note:*** This field is preserved by Intel NVM update tool.
13	Allow PXE Disable	0b	**Allow PXE Disable** 0b = Disabled 1b = Enabled ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
12:6	Reserved	0x0	Reserved. Must be 0. ***Note:*** This field is preserved by Intel NVM update tool.
5	FCOE Boot	0b	**FCoE Boot** FCoE boot code is present if set to 1b. 0b = Not Present 1b = Present ***Note:*** This field is preserved by Intel NVM update tool.
4	iSCSI Boot	0b	**iSCSI Boot** iSCSI is present if set to 1b. 0b = Not Present 1b = Present ***Note:*** This field is preserved by Intel NVM update tool.
3	EFI/UNDI Driver	0b	**EFI/UNDI Driver** EFI UNDI driver is present if set to 1b. 0b = Not Present 1b = Present ***Note:*** This field is preserved by Intel NVM update tool.
2	RPL	0b	**RPL** RPL module is present if set to 1b. Reserved bit for devices. 0b = Not Present 1b = Present ***Note:*** This field is preserved by Intel NVM update tool.
1	PXE/UNDI Driver	1b	**PXE/UNDI Driver** PXE UNDI driver is present if set to 1b. 0b = Not Present 1b = Present ***Note:*** This field is preserved by Intel NVM update tool.
0	PXE Base Code	1b	**PXE Base Code** PXE Base Code is present if set to 1b. 0b = Not Present 1b = Present ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.53 PXE Setup Options PCI Function 1 (0x0034)

This word is the same as word 0x30, but for function 1 of the device.

Bit(s)	Field Name	Default	Description
15:13	Reserved	000b	Reserved. Must be 0x0. ***Note:*** This field is preserved by Intel NVM update tool.
12:10	FSD	000b	**Force Speed and Duplex** Bits 12-10 control forcing speed and duplex during driver operation. 000b = Auto-negotiate (0x0) 001b = 10 Mb/s half duplex (0x1) 010b = 100 Mb/s half duplex (0x2) 011b = Not valid (0x3, treated as 0x0). 100b = Not valid (0x4. treated as 0x0). 101b = 10 Mb/s full duplex (0x5) 110b = 100 Mb/s full duplex (0x6) 111b = 1000 Mb/s full duplex (0x7) Only applicable for copper-based adapters. Not applicable to 10 GbE. Default value is 000b. ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
9	LWS	0b	**Legacy OS Wakeup Support** For 82559-based adapters only. If set to 1b, the agent enables PME in the adapter's PCI configuration space during initialization. This allows remote wake up under legacy operating systems that do not normally support it. Note that enabling this makes the adapter technically non-compliant with the ACPI specification, which is why the default is disabled. Must be set to 0b for 1 GbE and 10 GbE adapters. 0b = Disabled (default) 1b = Enabled ***Note:*** This field is preserved by Intel NVM update tool.
8	DSM	1b	**Display Setup Message** If the bit is set to 1b, the "Press Control-S" message is displayed after the title message. Default value is 1b. ***Note:*** This field is preserved by Intel NVM update tool.
7:6	PT	00b	**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. ***Note:*** This field is preserved by Intel NVM update tool.
5	iSCSI Disable	0b	**iSCSI Disable** 0b = Enabled 1b = Disabled ***Note:*** This field is preserved by Intel NVM update tool.
4:3	DBS	00b	**Default Boot Selection** These bits select which device is the default boot device. These bits are only used if the agent detects that the BIOS does not support boot order selection or if the MODE field of word 31h is set to MODE_LEGACY. 00b = Network boot, then local boot (default) 01b = Local boot, then network boot 10b = Network boot only 11b = Local boot only ***Note:*** This field is preserved by Intel NVM update tool.
2:0	PS	000b	**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. 000b = PXE (default value). 001b = Boot Disabled 010b = iSCSI Primary 011b = iSCSI Secondary 100b = FCoE All other values are reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.54 PXE Configuration Customization Options PCI Function 1 (0x0035)

This word is the same as word 0x31, but for function 1 of the device.

Bit(s)	Field Name	Default	Description
15:14	Signature	01b	**Signature** Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software. ***Note:*** This field is preserved by Intel NVM update tool.
13:12	Reserved	00b	Reserved. Must be 0. ***Note:*** This field is preserved by Intel NVM update tool.
11	Continuous Retry	0b	**Continuous Retry** Selects Continuous Retry operation. If this bit is set, IBA will 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 will restart 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 will not be attempted due to hardware conditions such as an invalid NVM checksum or failing to establish link. Default value is 0b. 0b = Disable 1b = Enable ***Note:*** This field is preserved by Intel NVM update tool.
10:8	Operating Mode	000b	**Operating Mode** Selects the agent's boot order setup mode. This field changes the agent's 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 will attempt to detect BBS and PnP Expansion ROM support as it normally does 001b = Force Legacy Mode — The agent will not attempt to detect BBS or PnP Expansion ROM supports in the BIOS and will assume the BIOS is not compliant. The user can change the BIOS boot order in the Setup Menu. 010b = Force BBS Mode — The agent will assume the BIOS is BBS-compliant, even though it may not be detected as such by the agent's detection code. The user can NOT change the BIOS boot order in the Setup Menu. 011b = Force PnP Int18 Mode — The agent will assume the BIOS allows boot order setup for PnP Expansion ROMs and will hook interrupt 18h (to inform the BIOS that the agent is a bootable device) in addition to registering as a BBS IPL device. The user can NOT change the BIOS boot order in the Setup Menu. 100b = Force PnP Int19 Mode — The agent will assume the BIOS allows boot order setup for PnP Expansion ROMs and will hook interrupt 19h (to inform the BIOS that the agent is a bootable device) in addition to registering as a BBS IPL device. The user can NOT 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. ***Note:*** This field is preserved by Intel NVM update tool.
7:6	Reserved	00b	Reserved. Must be 0. ***Note:*** This field is preserved by Intel NVM update tool.
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. 0b = Enable Flash Update 1b = Disable Flash Update ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
4	Disable Legacy OS Wakeup Menu	0b	**Disable Legacy Wakeup Support** If this bit is set to 1b, the user is not allowed to change the Legacy OS Wakeup Support menu option. Default value is 0b. 0b = Enable Legacy Wakeup Support 1b = Disable Legacy Wakeup Support ***Note:*** This field is preserved by Intel NVM update tool.
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. 0b = Enable 1b = Disable ***Note:*** This field is preserved by Intel NVM update tool.
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. 0b = Enable 1b = Disable ***Note:*** This field is preserved by Intel NVM update tool.
1	Disable Title Message Display	0b	**Disable Title Message** If this bit is set to 1b, the title message displaying the version of the Boot Agent is suppressed; the Control-S message is also suppressed. This is for OEMs who do not wish the boot agent to display any messages at system boot. Default value is 0b. 0b = Enable 1b = Disable ***Note:*** This field is preserved by Intel NVM update tool.
0	Disable 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 may only be changed via an external program. Default value is 0b. 0b = Enable 1b = Disable ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.55 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.

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.56 Alternate Ethernet MAC Addresses Pointer (0x0037)

The alternate MAC Address does not function if the pointer value is set to 0xFFFF. Word offset for port 0 MAC Address is: value in word 0x37 + 0 ; ;- 3 words. Word offset for port 1 MAC Address is: value in word 0x37 + 3 ; ;- Next 3 words.

Bit(s)	Field Name	Default	Description
15:0	Pointer	0xFFFF	**Pointer** Points to Alternate Ethernet MAC Address Section. For Alternate Ethernet MAC Address inner structure, see Section 6.2.8.

6.2.2.57 NVM Control Word 3 (0x0038)

Bit(s)	Field Name	Default	Description
15:9	Reserved	0x03	Reserved
8	D10GMP Port 1	1b	**D10GMP Port 1** Disable 10 GbE in LPLU for LAN Port 1. When set, LAN port 1 never advertises 10 GbE speed capability when in LPLU state (D3/Dr). ***Note:*** This field is preserved by Intel NVM update tool.
7	D1GMP Port 1	0b	**D1GMP Port 1** Disable 1 GbE in LPLU for LAN Port 1. When set, LAN port 1 never advertises 1 GbE speed capability when in LPLU state (D3/Dr). If set, D10GMP bit must be set as well. ***Note:*** This field is preserved by Intel NVM update tool.
6	D10GMP Port 0	1b	**D10GMP Port 0** Disable 10 GbE in LPLU for LAN Port 0. When set, LAN port 0 never advertises 10 GbE speed capability when in LPLU state (D3/Dr). ***Note:*** This field is preserved by Intel NVM update tool.
5	D1GMP Port 0	0b	**D1GMP Port 0** Disable 1 GbE in LPLU for LAN Port 0. When set, LAN port 0 never advertises 1 GbE speed capability when in LPLU state (D3/Dr). If set, D10GMP bit must be set as well. ***Note:*** This field is preserved by Intel NVM update tool.
4	Reserved	0b	Reserved
3	Enable LPLU	1b	**Enable LPLU** Enable the Low Power Link Up feature. When set, enables a decrease in link speed of the port defined to stay awake in non-D0a states when power policy and power management states dictate it. ***Note:*** This field is preserved by Intel NVM update tool.
2	Keep_PHY_Link_Up_En	1b	**Keep_PHY_Link_Up_En** Enables No PHY Link Down when the MC indicates that the PHY should be kept on. When asserted, this bit prevents changes in power management state to be reflected to the PHYs according to the MMNGC.MNG_VETO bit value. When cleared, the MMNGC.MNG_VETO bit is meaningless. ***Note:*** This field is preserved by Intel NVM update tool.
1	APM Enable Port 1	0b	**APM Enable Port 1** Initial value of advanced power management wake up enable in the General Receive Control register (GRC.APME). Mapped to GRC.APME of port 1. ***Note:*** This field is preserved by Intel NVM update tool.
0	APM Enable Port 0	0b	**APM Enable Port 0** Initial value of advanced power management wake up enable in the General Receive Control register (GRC.APME). Mapped to GRC.APME of port 0. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.2.58 FCoE Scratch Pad Pointer (0x0039)

Bit(s)	Field Name	Default	Description
15:0	FCOE Scratch Pad Pointer	0x0	**FCOE Scratch Pad Pointer** Points to FCoE Scratch Pad Header Section. For FCoE Scratch Pad Header inner structure, see Section 6.2.9.

6.2.2.59 Firmware Code Pointer (0x003A)

The address is in 4KB sector index units.

Bit(s)	Field Name	Default	Description
15	Pointer Type	1b	**Pointer Type** 0b = Word units 1b = 4 KB sector units
14:0	Firmware Code Pointer	0x0	Points to Firmware Module Section. For Firmware Module inner structure, see Section 6.2.30.

6.2.2.60 Hardware (0x003B)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.61 Hardware (0x003C)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.2.62 Hardware (0x003D)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.63 Hardware (0x003E)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.64 Software Checksum, Words 0x00 - 0x3F (0x003F)

The checksum word (0x3F) is used to ensure that the base NVM image is a valid image. It covers contents of all the NVM modules located in the first valid 4KB sector, excepted to all Firmware modules. The value of this word should be calculated such that after adding all the concerned words, including the checksum word itself, the sum should be 0xBABA. This word is used strictly by the software. The

hardware does not calculate nor check its content but rather checks the *Signature* field in the NVM Control Word 1.

Bit(s)	Field Name	Default	Description

15:0	Checksum		**Checksum**
			The following documents the calculation:
			#define IXGBE_EEPROM_CHECKSUM 0x3F
			#define IXGBE_EEPROM_SUM 0xBABA
			#define IXGBE_PCIE_ANALOG_PTR 0x03
			#define IXGBE_PHY_PTR 0x04
			#define IXGBE_OPTION_ROM_PTR 0x05
			#define IXGBE_FW_PTR 0x0F
			/**
			* ixgbe_calc_eeprom_checksum_X540 - Calculates and returns the checksum
			* @hw: pointer to hardware structure
			**/
			u16 ixgbe_calc_eeprom_checksum_X540(struct ixgbe_hw *hw)
			{
			u16 i;
			u16 j;
			u16 checksum = 0;
			u16 length = 0;
			u16 pointer = 0;
			u16 word = 0;
			DEBUGFUNC("ixgbe_calc_eeprom_checksum_X540");
			/* Include 0x0-0x3F in the checksum */
			for (i = 0; i < IXGBE_EEPROM_CHECKSUM; i++) {
			if (hw->eeprom.ops.read(hw, i, &word) != IXGBE_SUCCESS) {
			DEBUGOUT("EEPROM read failed\n");
			break;
			}
			checksum += word;
			}
			/*
			* Include all data from pointers 0x3, 0x6-0xE. This excludes the
			* FW, PHY module, and PCIe Expansion/Option ROM pointers.
			*/
			for (i = IXGBE_PCIE_ANALOG_PTR; i < IXGBE_FW_PTR; i++) {
			if (i == IXGBE_PHY_PTR || i == IXGBE_OPTION_ROM_PTR)
			continue;
			if (hw->eeprom.ops.read(hw, i, &pointer) != IXGBE_SUCCESS) {
			DEBUGOUT("EEPROM read failed\n");
			break;
			}
			/* Skip pointer section if the pointer is invalid. */
			if (pointer == 0xFFFF || pointer == 0 ||
			pointer >= hw->eeprom.word_size)
			continue;
			if (hw->eeprom.ops.read(hw, pointer, &length) != IXGBE_SUCCESS) {
			DEBUGOUT("EEPROM read failed\n");
			break;
			}
			/* Skip pointer section if length is invalid. */
			if (length == 0xFFFF || length == 0) ||
			(pointer + length) >= hw->eeprom.word_size)
			continue;
			for (j = pointer+1; j <= pointer+length; j++) {
			if (hw->eeprom.ops.read(hw, j, &word) != IXGBE_SUCCESS) {
			DEBUGOUT("EEPROM read failed\n");
			break;
			}
			checksum += word;
			}
			}
			checksum = (u16)IXGBE_EEPROM_SUM - checksum;
			return checksum;
			}

6.2.2.65 Free Provisioning Area Pointer (0x0040)

Bit(s)	Field Name	Default	Description
15	Pointer Type	1b	
14:0	Free Provisioning Area Pointer	0x0	**Free Provisioning Area Pointer** Points to Free Provisioning Module Section. For Free Provisioning Module inner structure, see Section 6.2.33.

6.2.2.66 Free Provisioning Area Size (0x0041)

Bit(s)	Field Name	Default	Description
15:0	Free Provisioning Area Size	0x7A	**Free Provisioning Area Size**

6.2.2.67 Mini Loader Pointer (0x0042)

Bit(s)	Field Name	Default	Description
15:0	Mini Loader Section Pointer	0xFFFF	**Mini Loader Section Pointer** Points to Mini Loader Module Section. For Mini Loader Module inner structure, see Section 6.2.24.

6.2.2.68 PHY Config Pointer (0x0043)

Bit(s)	Field Name	Default	Description
15:0	PHY Config Section Pointer	0xFFFF	**PHY Config Section Pointer** Points to PHY Config Section. For PHY Config inner structure, see Section 6.2.25.

6.2.2.69 Reserved (0x0044)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.70 Reserved (0x0045)

6.2.2.71 Reserved (0x0046)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.72 Reserved (0x0047)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.73 Reserved (0x0048)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.74 Reserved (0x0049)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.75 Reserved (0x004A)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.76 Reserved (0x004B)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.77 Reserved (0x004C)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.78 Reserved (0x004D)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.79 Reserved (0x004E)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.80 Reserved (0x004F)

Bit(s)	Field Name	Default	Description
15:0	Reserved		Reserved.

6.2.2.81 RO Updates Version (0x0050)

Bit(s)	Field Name	Default	Description
15:0	RO Updates Version	0x0	Reserved.

6.2.3 MAC Module Section

Table 6-9. MAC Module General Summary Table

Word Offset	Description	Section Number
0x0000	Reserved	6.2.3.1
0x0001	MAC Configuration	6.2.3.2
0x0002	MAC Configuration Extended	6.2.3.3

6.2.3.1 Reserved (0x0000)

6.2.3.2 MAC Configuration (0x0001)

Bit(s)	Field Name	Default	Description
15	Reserved	0b	Reserved.
14	Reserved	00b	Reserved.
13:10	FIFOLT	0x3	**FIFO Low Threshold** Determines the low threshold of the MAC elastic FIFO. Mapped to MACC.*FIFOHT*.
9:6	FIFOHT	0xD	FIFO High Threshold. Determines the high threshold of the MAC elastic FIFO. Mapped to MACC.*FIFOLT*.
5	Swap Rx Control	0b	**Swap Rx Control** Swap the MAC internal Rx XGMII controls 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.*Swap Rx Contro*l.
4	Swap Rx Data	0b	**Swap Rx Data** Swap the MAC internal Rx XGMII lanes. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.*Swap Rx Data.*
3	Swizzle Rx Data	0b	**Swizzle Rx Data** Swizzle the bytes in all 4 MAC internal Rx XGMII lanes. 0b = Swizzle disabled, normal mode. 1b = Swizzle enabled. Mapped to MACC.*Swizzle Rx Data.*
2	Swap Tx Control	0b	**Swap Tx Control** Swap the MAC internal Tx XGMII controls. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.*Swap Tx Control*.
1	Swap Tx Data	0b	**Swap Tx Data** Swap the MAC internal Tx XGMII lanes. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.*Swap Tx Data*.
0	Swizzle Tx Data	0b	**Swizzle Tx Data** Swizzle the bytes in all 4 MAC internal Tx XGMII lanes. 0b = Swizzle disabled, normal mode. 1b = Swizzle enabled. Mapped to MACC.*Swizzle Tx Data*.

6.2.3.3 MAC Configuration Extended (0x0002)

Bit(s)	Field Name	Default	Description
15:3	Reserved	0x0	Reserved.
2	LDINFITR	0b	**Link Down in Fatal Interrupt** 0b = Normal operation. The link indication is the same as comes from the PCS and auto negotiation machinery. 1b = Link indication is gated with the Fatal interrupt from the PHY. When the PHY assert fatal interrupt the link is down.
1:0	BLKXGMII	00b	**Block XGMII Interface** 00b = Normal Operation — The XGMII interface fed from the PCS while link is down. 01b = Idle State — The XGMII interface fed with Idles while link is down. 10b = Local Fault State — The XGMII interface fed with Local Fault sequences while link is down. 11b = Reserved.

6.2.4 CSR Auto Config Section

Table 6-10. CSR Auto Config Section Summary Table

Word Offset	Description	Section Reverence
0x0000	Section Length	6.2.4.1
0x0001	CSR RAW1	6.2.4.2
0x0002	CSR RAW2	6.2.4.3

6.2.4.1 Section Length (0x0000)

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word. Section Length = 3*n (n = number of CSRs to configure).

Bit(s)	Field Name	Default	Description
15:0	Section Length		**Section Length** Length in: 2 Bytes unit - 1 First Section -> Word: CSR Auto Config -> Section length Last Section -> Word: CSR Auto Config -> CSR RAW2 Section length in words

6.2.4.2 CSR RAW1 (0x0001)

Raw data module length: variable

6.2.4.3 CSR RAW2 (0x0002)

Raw data module length: variable

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word. Section Length = 3*n (n = number of CSRs to configure).

6.2.5 LAN Core Module Section

Table 6-11. LAN Core Module Section Summary Table

Word Offset	Description	Section Number
0x0000	Section Length	6.2.5.1
0x0001	Ethernet MAC Address Register1	6.2.5.2
0x0002	Ethernet MAC Address Register2	6.2.5.3
0x0003	Ethernet MAC Address Register3	6.2.5.4
0x0004	LED Control Lower Word	6.2.5.5
0x0005	LED Control Upper Word	6.2.5.6
0x0006	SDP Control	6.2.5.7
0x0007	Filter Control	6.2.5.8

6.2.5.1 Section Length (0x0000)

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word.

Bit(s)	Field Name	Default	Description
15:0	Section Length		**Section Length** Length in: 2 Bytes unit - 1 First Section -> Word: LAN Core Module -> Section Length Last Section -> Word: LAN Core Module -> Filter Control Section Length in words

6.2.5.2 Ethernet MAC Address Register1 (0x0001)

The Ethernet Individual Address (IA) is a 6-byte field that must be unique for each NIC or LOM and must also be unique for each copy of the NVM image. The first three bytes are vendor specific. For example, the IA is equal to [00 AA 00] or [00 A0 C9] for Intel products. The value of this field is loaded into the Receive Address register 0 (RAL0/RAH0). For the purpose of this specification, the numbering convention is as follows: Vendor ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;1 ; ; ; ; ; ; ; ; ; ; ; ; ;2 ; ; ; ; ; ; ; ; ; ; ; ; ; ;3 ; ; ; ; ; ; ; ; ; ; ; ; ; ;4 ; ; ; ; ; ; ; ; ; ; ; ; ;5 ; ; ; ; ; ; ; ; ; ; ; ; ;6 Intel original ; ; ; ; ;00 ; ; ; ; ; ; ; ; ; ;AA ; ; ; ; ; ; ; ; ; ; ;00 ; ; ;Variable ; ; ;Variable ; ; ;Variable Intel new ; ; ; ; ; ; ; ; ; ;00 ; ; ; ; ; ; ; ; ; ; ;A0 ; ; ; ; ; ; ; ; ; ; ;C9 ; ; ;Variable ; ; ;Variable ; ; ;Variable

Bit(s)	Field Name	Default	Description
15:8	Eth_Addr_Byte2	0x12	**Ethernet MAC Address Byte 2** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Eth_Addr_Byte1	0x34	**Ethernet MAC Address Byte 1** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.5.3 Ethernet MAC Address Register2 (0x0002)

Bit(s)	Field Name	Default	Description
15:8	Eth_Addr_Byte4	0x56	**Ethernet MAC Address Byte 4** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Eth_Addr_Byte3	0x78	**Ethernet MAC Address Byte 3** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.5.4 Ethernet MAC Address Register3 (0x0003)

Bit(s)	Field Name	Default	Description
15:8	Eth_Addr_Byte6	0x0	**Ethernet MAC Address Byte 6** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Eth_Addr_Byte5	0x0	**Ethernet MAC Address Byte 5** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.5.5 LED Control Lower Word (0x0004)

The LEDCTL register defaults are loaded from two words as listed in the following tables.

Bit(s)	Field Name	Default	Description
15	LED1_BLINK	0b	**LED 1 Blink** 0b = Don't Blink 1b = Blink ***Note:*** This field is preserved by Intel NVM update tool.
14	LED1_IVRT	1b	**LED 1 Invert** 0b = Active Low 1b = Active High ***Note:*** This field is preserved by Intel NVM update tool.
13:12	Reserved	00b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
11:8	LED1_MODE	0x1	**LED 1 Mode** LED 1 control: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF All other values are reserved. ***Note:*** This field is preserved by Intel NVM update tool.
7	LED0_BLINK	0b	**LED 0 Blink** 0b = Don't Blink 1b = Blink ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
6	LED0_IVRT	1b	**LED 0 Invert** 0b = Active Low 1b = Active High ***Note:*** This field is preserved by Intel NVM update tool.
5	Global Link Mode	0b	**Global Link Mode** 0b = 200 ms 1b = 83 ms ***Note:*** This field is preserved by Intel NVM update tool.
4	Reserved	0b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
3:0	LED0_MODE	0x0	**LED 0 Mode** LED 0 control: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF All other values are reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.5.6 LED Control Upper Word (0x0005)

Bit(s)	Field Name	Default	Description
15	LED3_BLINK	0b	**LED 3 Blink** 0b = Don't Blink 1b = Blink ***Note:*** This field is preserved by Intel NVM update tool.
14	LED3_IVRT	1b	**LED 3 Invert** 0b = Active Low 1b = Active High ***Note:*** This field is preserved by Intel NVM update tool.
13:12	Reserved	00b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
11:8	LED3_MODE	0x5	**LED 3 Mode** LED 3 control: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF All other values are reserved. ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
7	LED2_BLINK	1b	**LED 2 Blink** 0b = Don't Blink 1b = Blink ***Note:*** This field is preserved by Intel NVM update tool.
6	LED2_IVRT	0b	**LED 2 Invert** 0b = Active Low 1b = Active High ***Note:*** This field is preserved by Intel NVM update tool.
5:4	Reserved	00b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
3:0	LED2_MODE	0x4	**LED 2 Mode** LED 2 control: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF All other values are reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.5.7 SDP Control (0x0006)

Bit(s)	Field Name	Default	Description
15	SDP3_NATIVE	0b	**SDP3 Native** Defines SDP3 operating mode is mapped to ESDP.*SDP3_NATIVE* loaded at power up. 0b = Operates as generic software controlled IO. 1b = Native mode operation (hardware function). ***Note:*** This field is preserved by Intel NVM update tool.
14	SDP2_NATIVE	0b	**SDP2 Native** Defines SDP2 operating mode is mapped to ESDP.*SDP2_NATIVE* loaded at power up. 0b = Operates as generic software controlled IO. 1b = Native mode operation (hardware function). ***Note:*** This field is preserved by Intel NVM update tool.
13	SDP1_NATIVE	0b	**SDP1 Native** Defines SDP1 operating mode is mapped to ESDP.*SDP1_NATIVE* loaded at power up. 0b = Operates as generic software controlled IO. 1b = Native mode operation (hardware function). ***Note:*** This field is preserved by Intel NVM update tool.
12	SDP0_NATIVE	0b	**SDP0 Native** Defines SDP0 operating mode is mapped to ESDP.*SDP0_NATIVE* loaded at power up. 0b = Operates as generic software controlled IO. 1b = Native mode operation (hardware function). ***Note:*** This field is preserved by Intel NVM update tool.
11	SDPDIR[3]	0b	**SDP3 Direction** Initial Direction is mapped to ESDP.*SDP3_IODIR* loaded at power up. ***Note:*** This field is preserved by Intel NVM update tool.
10	SDPDIR[2]	0b	**SDP2 Direction** Initial Direction is mapped to ESDP.*SDP2_IODIR* loaded at power up. ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
9	SDPDIR[1]	0b	**SDP1 Direction** Initial Direction is mapped to ESDP.*SDP1_IODIR* loaded at power up. ***Note:*** This field is preserved by Intel NVM update tool.
8	SDPDIR[0]	0b	**SDP0 Direction** Initial Direction is mapped to ESDP.*SDP0_IODIR* loaded at power up. ***Note:*** This field is preserved by Intel NVM update tool.
7	Reserved	0b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
6	REVERSE_DIR	0b	**Reverse Direction** When set, reverses the polarity of the pin direction signal to pads block, such that pads are fed with output enable signal instead of output disable. Should be cleared for normal operation. ***Note:*** This field is preserved by Intel NVM update tool.
5	SDP23_function	0b	**SDP23 Function** Defines the initial value for ESDP.*SDP23_function.* 0b = 1588 functionality 1b = I^2C functionality. Relevant only if *SDP2_NATIVE* or *SDP3_NATIVE* is set. If this bit is set, then *SDP2_NATIVE* and *SDP3_NATIVE* should have the same value. ***Note:*** This field is preserved by Intel NVM update tool.
4	SDP1_function	0b	**SDP1 Function** Defines the initial value for ESDP.*SDP1_function*. 0b = Time Sync functionality. 1b = SDP1 Thermal sensor functionality. *SDPDIR[1]* should be configured as an output. Relevant only if *SDP1_NATIVE* is set. ***Note:*** This field is preserved by Intel NVM update tool.
3	SDPVAL[3]	0b	**SDP3 Value** Initial Output Value is mapped to ESDP.*SDP3_DATA* loaded at power up. ***Note:*** This field is preserved by Intel NVM update tool.
2	SDPVAL[2]	0b	**SDP2 Value** Initial Output Value is mapped to ESDP.*SDP2_DATA* loaded at power up. ***Note:*** This field is preserved by Intel NVM update tool.
1	SDPVAL[1]	0b	**SDP1 Value** Initial Output Value is mapped to ESDP.*SDP1_DATA* loaded at power up. ***Note:*** This field is preserved by Intel NVM update tool.
0	SDPVAL[0]	0b	**SDP0 Value** Initial Output Value is mapped to ESDP.*SDP0_DATA* loaded at power up. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.5.8 Filter Control (0x0007)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0001	Reserved.

6.2.6 PCIe General Configuration Module Section

Table 6-12. PCIe General Configuration Module Section Summary Table

Word Offset	Description	Section Number
0x0000	PCI_CNF2 Low	6.2.6.1
0x0001	PCI_CNF2 High	6.2.6.2
0x0002	PCI_LBARCTRL L	6.2.6.3
0x0003	PCI_LBARCTRL H	6.2.6.4
0x0004	PCI_SERL 1	6.2.6.5
0x0005	PCI_SERL 2	6.2.6.6
0x0006	PCI_SERL 3	6.2.6.7
0x0007	PCI_SERL 4	6.2.6.8
0x0008	PCI_CAPCTRL L	6.2.6.9
0x0009	PCI_CAPCTRL H	6.2.6.10
0x000A	PCI_CAPSUP L	6.2.6.11
0x000B	PCI_CAPSUP H	6.2.6.12
0x000C	PCI_DBGCTL L	6.2.6.13
0x000D	PCI_DBGCTL H	6.2.6.14
0x000E	PCI_UPADD L	6.2.6.15
0x000F	PCI_UPADD H	6.2.6.16
0x0010	PCI_SUBSYSID L	6.2.6.17
0x0011	PCI_SUBSYSID H	6.2.6.18
0x0012	PCI_PWRDATA L	6.2.6.19
0x0013	PCI_PWRDATA H	6.2.6.20
0x0014	PCI_REVID L	6.2.6.21
0x0015	PCI_REVID H	6.2.6.22
0x0016	PCI_VFSUP L	6.2.6.23
0x0017	PCI_VFSUP H	6.2.6.24
0x0018	TPH_CTRL L	6.2.6.25
0x0019	TPH_CTRL H	6.2.6.26
0x001A	PCI_LINKCAP L	6.2.6.27
0x001B	PCI_LINKCAP H	6.2.6.28
0x001C	PCI_PMSUP L	6.2.6.29
0x001D	PCI_PMSUP H	6.2.6.30
0x001E	PCI_GLBL_CNF L	6.2.6.31
0x001F	PCI_GLBL_CNF H	6.2.6.32
0x0020	PCI_VENDORID L	6.2.6.33
0x0021	PCI_VENDORID H	6.2.6.34

Table 6-12. PCIe General Configuration Module Section Summary Table [continued]

Word Offset	Description	Section Number
0x0022	PCI_PCIERR L	6.2.6.35
0x0023	PCI_PCIERR H	6.2.6.36

6.2.6.1 PCI_CNF2 Low (0x0000)

MSI-x and cache line configuration.

Bit(s)	Field Name	Default	Description
15:0	PCI_CNF2 Low	0x400040FC	**PCI_CNF2 Low**

6.2.6.2 PCI_CNF2 High (0x0001)

Bit(s)	Field Name	Default	Description
15:0	PCI_CNF2 High		**PCI_CNF2 High**

6.2.6.3 PCI_LBARCTRL L (0x0002)

PF BAR registers configuration.

Bit(s)	Field Name	Default	Description
15:14	RESERVED	00b	Reserved.
13:11	EXROM_BAR_	011b	**Expansion ROM BAR** 000b = 64 KB 001b = 128 KB 010b = 256 KB 011b = 512 KB 100b = 1 MB 101b = 2 MB 110b = 4 MB 111b = 8 MB
10:9	RESERVED	00b	Reserved.
8:6	FL_BAR_SIZE	101b	**Flash BAR Size** 101b = 2 MB 110b = 4 MB 111b = 8 MB All other values are reserved.
5:4	RESERVED	00b	Reserved.
3	FLASH_EXPOSE	1b	**Flash Expose** 0b = Disabled 1b = Enabled
2	CSRSIZE	1b	**CSR Size** ***Note:*** This field is preserved by Intel NVM update tool.
1	BAR32	0b	**BAR32** 0b = 64-bit Addressing 1b = 2-bit Addressing

Bit(s)	Field Name	Default	Description
0	PREFBAR	1b	**Prefetch BAR** 0b = No Prefetch 1b = Prefetch Enabled

6.2.6.4 PCI_LBARCTRL H (0x0003)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.6.5 PCI_SERL 1 (0x0004)

PCIe Serial Number.

Bit(s)	Field Name	Default	Description
15:0	PCI_SER 1	0x0	**PCIe Serial Number 1** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.6.6 PCI_SERL 2 (0x0005)

Bit(s)	Field Name	Default	Description
15:0	PCI_SERL 2	0xC9	**PCIe Serial Number 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.6.7 PCI_SERL 3 (0x0006)

Bit(s)	Field Name	Default	Description
15:0	PCI_SERL 3	0x0	**PCIe Serial Number 3** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.6.8 PCI_SERL 4 (0x0007)

Bit(s)	Field Name	Default	Description
15:0	PCI_SERL 4	0xC9	**PCIe Serial Number 4** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.6.9 PCI_CAPCTRL L (0x0008)

VPD_EN

Bit(s)	Field Name	Default	Description
15:1	PCI_CAPCTRL L	0x0	**PCIe Capabilities Control Low**
0	VPD_EN	0xC9	**VPD Enable** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.6.10 PCI_CAPCTRL H (0x0009)

Bit(s)	Field Name	Default	Description
15:0	PCI_CAPCTRL H	0x0	**PCIe Capabilities Control High**

6.2.6.11 PCI_CAPSUP L (0x000A)

PCIe Capabilities exposed.

Bit(s)	Field Name	Default	Description
15	ECRC_MCTP_GEN	0b	**ECRC Generation for MCTP** 0b = Disabled — Do not add ECRC to MCTP packets even if ECRC is enabled. 1b = Enabled — Add ECRC to MCTP packets if ECRC is enabled via the ECRC *Generation Enable* field in PCIe Advanced Error Capabilities and Control Register.
14:8	Reserved	0x0	Reserved.
7	SEC_EN	1b	**Secondary Enable** A value of 1b indicates support for the Secondary PCI Express Extended Capability. 0b = Disabled 1b = Enabled
6	ACS_EN	1b	**ACS Enable** A value of 1b indicates support for the PCIe ACS Capability 0b = Disabled 1b = Enabled
5	IOV_EN	1b	**SR-IOV Enable** A value of 1b indicates support for the PCIe SR-IOV Capability 0b = Disabled 1b = Enabled ***Note:*** This field is preserved by Intel NVM update tool.
4	ARI_EN	1b	**ARI Enable** A value of 1b indicates support for the PCIe ARI Capability 0b = Disabled 1b = Enabled
3	TPH_EN	1b	**TPH Enable** A value of 1b indicates support for the PCIe TPH Requester Capability 0b = Disabled 1b = Enabled
2	LTR_EN	1b	**LTR Enable** A value of 1b indicates support for the PCIe Latency Tolerance Reporting (LTR) Capability 0b = Disabled 1b = Enabled
1	Reserved	0b	Reserved.

Bit(s)	Field Name	Default	Description
0	PCIE_VER	1b	PCIe Version 0b = 0x1-Capability version 1b = 0x2-Capability version

6.2.6.12 PCI_CAPSUP H (0x000B)

Bit(s)	Field Name	Default	Description
15	LOAD_DEV_ID	1b	**Load Device ID** When set to 1b, indicates that the device loads its PCI device IDs from NVM. 0b = Disabled 1b = Enabled
14	LOAD_SUBSYS_ID	1b	**Load Subsystem IDs** When set to 1b, indicates that the device loads its PCIe subsystem ID and sub-system vendor ID from NVM. 0b = Disabled 1b = Enabled
13:5	Reserved	0x0	Reserved.
4	CSR_CONF__EN	1b	**CSR Configuration Space Enable** Enables Access to CSRs via the PCI Configuration Space. Section 8.1.3, "Configuration Access to Internal Registers and Memories". 0b = Disabled 1b = Enabled
3	MSI_MASK	1b	**MSI Mask** MSI per-vector masking setting. This bit is loaded to the masking bit (bit 8) in the Message Control of the MSI Configuration Capability structure. 0b = Disabled 1b = Enabled
2	IDO__EN	1b	**ID-Based Ordering (IDO) Enable** 0b = Disabled 1b = Enabled
1	ECRC_CHK__EN	1b	**ECRC Check Enable** Loaded into the ECRC Check Capable bit of the PCIe Configuration registers. 0b = Disabled 1b = Enabled
0	ECRC_GEN__EN	1b	**ECRC Generation Enable** Loaded into the ECRC Generation Capable bit of the PCIe Configuration registers. 0b = Disabled 1b = Enabled

6.2.6.13 PCI_DBGCTL L (0x000C)

Allow access to VF config space.

Bit(s)	Field Name	Default	Description
15:0	PCI_DBGCTL L	0x1	**PCIe Debug Control Low**

6.2.6.14 PCI_DBGCTL H (0x000D)

Bit(s)	Field Name	Default	Description
15:0	PCI_DBGCTL H		**PCIe Debug Control High**

6.2.6.15 PCI_UPADD L (0x000E)

Upper Address accessible.

Bit(s)	Field Name	Default	Description
15:1	ADDRESS	0x0	**Address**
0	MODE	0x0	**Mode**

6.2.6.16 PCI_UPADD H (0x000F)

Bit(s)	Field Name	Default	Description
15:0	ADDRESS	0x0	**Address**

6.2.6.17 PCI_SUBSYSID L (0x0010)

PCIe Subsystem ID.

Bit(s)	Field Name	Default	Description
15:0	SUB_VEN_ID	0x8086	**Sub-Vendor ID** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.6.18 PCI_SUBSYSID H (0x0011)

Bit(s)	Field Name	Default	Description
15:0	SUB_ID	0x0	**Sub-System ID** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.6.19 PCI_PWRDATA L (0x0012)

PCIe Power Data register.

Bit(s)	Field Name	Default	Description
15:0	PCI_PWRDATA L	0x0	**PCIe Power Data Low**

6.2.6.20 PCI_PWRDATA H (0x0013)

Bit(s)	Field Name	Default	Description
15:0	PCI_PWRDATA H		**PCIe Power Data High**

6.2.6.21 PCI_REVID L (0x0014)

Revision ID.

Bit(s)	Field Name	Default	Description
15:0	PCI_REVID L	0x0	**PCIe Revision ID Low**

6.2.6.22 PCI_REVID H (0x0015)

Bit(s)	Field Name	Default	Description
15:0	PCI_REVID H		**PCIe Revision ID High**

6.2.6.23 PCI_VFSUP L (0x0016)

PF Bar registers configuration.

Bit(s)	Field Name	Default	Description
15:0	PCI_VFSUP L	0x2	**PCIe VF Support Low**

6.2.6.24 PCI_VFSUP H (0x0017)

Bit(s)	Field Name	Default	Description
15:0	PCI_VFSUP H		**PCIe VF Support High**

6.2.6.25 TPH_CTRL L (0x0018)

TPH Control registers.

Bit(s)	Field Name	Default	Description
15:0	TPH_CTRL L	0x00001800	**TPH Control Low**

6.2.6.26 TPH_CTRL H (0x0019)

Bit(s)	Field Name	Default	Description
15:0	TPH_CTRL H		**TPH Control High**

6.2.6.27 PCI_LINKCAP L (0x001A)

PCIe Link Capabilities.

Bit(s)	Field Name	Default	Description
15:13	Reserved	000b	Reserved.
12:9	LINK_MAX_WIDTH	0x7	**Link Maximum Width** 0x1 = x1 limit 0x3 = x4 limit 0x4 = x8 limit 0x7 = No limit All other values are reserved.
8:6	LINK_MAX_PAYLOAD	010b	**Link Maximum Payload** 000b = 128 Bytes 001b = 256 Bytes 010b = 512 Bytes 011b = 1024 Bytes 100b = 2048 Bytes All other values are reserved.
5:2	Reserved	0x0	Reserved.
1	LINK_SPEED_VECTOR_8GT	1b	**Link Speed Vector 8 GT/s** 0b = Disabled 1b = Enabled
0	LINK_SPEED_VECTOR_5GT	1b	**Link Speed Vector 5 GT/s** 0b = Disabled 1b = Enabled

6.2.6.28 PCI_LINKCAP H (0x001B)

Bit(s)	Field Name	Default	Description
15:0	PCI_LINKCAP H	0x0	**PCIe Link Capabilities High**

6.2.6.29 PCI_PMSUP L (0x001C)

PCIe Power Management support.

Bit(s)	Field Name	Default	Description
15:0	PCI_PMSUP L	0x00007397	**PCIe Power Management Support Low** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.6.30 PCI_PMSUP H (0x001D)

Bit(s)	Field Name	Default	Description
15:0	PCI_PMSUP H		**PCIe Power Management Support High**

6.2.6.31 PCI_GLBL_CNF L (0x001E)

Wake# enable.

Bit(s)	Field Name	Default	Description
15:8	PCIE_CLKGATE_TIMER	0x10	**PCIe Clock Gate Timer** Clock gating idle timer. This field defines the number of clocks (CSR clock) of idle-detect before gating the PCIE clocks.
7:6	RESERVED	00b	Reserved.
5	PCIE_PCLK_GATE_EN	0b	**PCIe Clock Gate Enable** When set to 1b, and if PCIE_CLKGATE_DIS is 0, the PCIE PCLK will also be dynamically gated.
4	PCIE_CLKGATE_L1_ONLY	0b	**PCIe Clock Gate L1 Only** When set to 1b, and if PCIE_CLKGATE_DIS is 0, the clock gating of the PCIE clocks will be only when the PCIE is in L1 state.
3	PCIE_CLKGAT_DIS	0b	**PCIe Clock Gate Disable** When set to 0b enables dynamic clock gating of the PCIE clocks (LCB, HIU & CSR).
2	WAKE_PIN_EN	0b	**Wake Pin Enable** When set to 1b enables the use of PE_WAKE_N pin for PME event in all power states, otherwise, it is enabled only in Dr state. ***Note:*** This field is preserved by Intel NVM update tool.
1:0	RESERVED	0b	Reserved.

6.2.6.32 PCI_GLBL_CNF H (0x001F)

Bit(s)	Field Name	Default	Description
15:0	RESERVED		Reserved.

6.2.6.33 PCI_VENDORID L (0x0020)

Wake# enable.

Bit(s)	Field Name	Default	Description
15:0	PCI_VENDORID L	0x8086	**PCIe Vendor ID Low**

6.2.6.34 PCI_VENDORID H (0x0021)

Bit(s)	Field Name	Default	Description
15:0	PCI_VENDORID H	0x0	**PCIe Vendor ID High**

6.2.6.35 PCI_PCIERR L (0x0022)

Error enable.

Bit(s)	Field Name	Default	Description
15:0	PCI_PCIERR L	0x3FCB	**PCIe Error Low**

6.2.6.36 PCI_PCIERR H (0x0023)

Bit(s)	Field Name	Default	Description
15:0	PCI_PCIERR H		**PCIe Error High**

6.2.7 PCIe Configuration Space Section

Table 6-13. PCIe Configuration Space Section Summary Table

Word Offset	Description	Section Number
0x0000	PCI_PFDEVID Low - LAN	6.2.7.1
0x0001	PCI_PFDEVID High - SAN	6.2.7.2
0x0002	PCI_CNF Low	6.2.7.3
0x0003	PCI_CNF High	6.2.7.4
0x0004	PCI_VFDEVID Low - LAN	6.2.7.5
0x0005	PCI_VFDEVID High - SAN	6.2.7.6
0x0006	PCI_CLASS Low	6.2.7.7
0x0007	PCI_CLASS High	6.2.7.8

6.2.7.1 PCI_PFDEVID Low - LAN (0x0000)

Bit(s)	Field Name	Default	Description
15:0	PF_DEV_ID_LAN	0x1563	**PF Device ID LAN** If the Load Device ID in offset 0x7 of section PCIe General configuration is set, this word is loaded to the device ID of the LAN function. 0x1563 = Default (2 Port) 0x15D1 = 1 Port

6.2.7.2 PCI_PFDEVID High - SAN (0x0001)

Bit(s)	Field Name	Default	Description
15:0	PF_DEV_ID_SAN	0x1563	**PF Device ID SAN**

6.2.7.3 PCI_CNF Low (0x0002)

Bit(s)	Field Name	Default	Description
15:7	Reserved	0x0	Reserved.
6:5	INT_PIN	00b	**Interrupt Pin** Controls the value advertised in the Interrupt Pin field of the PCI configuration header for this function. 00b = INTA# 01b = INTB# 10b = INTC# 11b = INTD# The value advertised in the PCI configuration header is the value loaded from NVM + 1.
4	IO_BAR	0b	**I/O BAR Support** 0b = I/O BAR is not supported. 1b = I/O BAR is supported.

Bit(s)	Field Name	Default	Description
3	EXROM_DIS	0b	**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. ***Note:*** This field is preserved by Intel NVM update tool.
2:0	Reserved	00b	Reserved.

6.2.7.4 PCI_CNF High (0x0003)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved

6.2.7.5 PCI_VFDEVID Low - LAN (0x0004)

Bit(s)	Field Name	Default	Description
15:0	VF_DEV_ID_LAN	0x1565	**VF Device ID LAN** If the Load Device ID in offset 0x7 of section PCIe General configuration is set, this word is loaded to the device ID of the LAN function.

6.2.7.6 PCI_VFDEVID High - SAN (0x0005)

Bit(s)	Field Name	Default	Description
15:0	VF_DEV_ID_SAN	0x1565	**VF Device ID SAN**

6.2.7.7 PCI_CLASS Low (0x0006)

Bit(s)	Field Name	Default	Description
15:1	Reserved	0x0	Reserved.
0	STORAGE_CLASS	0b	**Storage Class** 0b = The class code of this port is set to 0x020000 (LAN). 1b = The class code of this port is set to 0x010000 (SCSI).

6.2.7.8 PCI_CLASS High (0x0007)

Bit(s)	Field Name	Default	Description
15:1	Reserved	0x0	Reserved.
0	STORAGE_CLASS	0b	**Storage Class** 0b = The class code of this port is set to 0x020000 (LAN). 1b = The class code of this port is set to 0x010000 (SCSI).

6.2.8 Alternate Ethernet MAC Address Section

Table 6-14. Alternate Ethernet MAC Address Section Summary Table

Word Offset	Description	Section Number
0x0000	MAC Address Word 1 Port 0	6.2.8.1
0x0001	MAC Address Word 2 Port 0	6.2.8.2
0x0002	MAC Address Word 3 Port 0	6.2.8.3
0x0003	MAC Address Word 1 Port 1	6.2.8.4
0x0004	MAC Address Word 2 Port 1	6.2.8.5
0x0005	MAC Address Word 3 Port 1	6.2.8.6

6.2.8.1 MAC Address Word 1 Port 0 (0x0000)

Port 0 MAC Address Word 1.

Bit(s)	Field Name	Default	Description
15:0	MAC Address Word 1 Port 0	0xFFFF	**MAC Address Word 1 Port 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.8.2 MAC Address Word 2 Port 0 (0x0001)

Port 0 MAC Address Word 2.

Bit(s)	Field Name	Default	Description
15:0	MAC Address Word 2 Port 0	0xFFFF	**MAC Address Word 2 Port 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.8.3 MAC Address Word 3 Port 0 (0x0002)

Port 0 MAC Address Word 3.

Bit(s)	Field Name	Default	Description
15:0	MAC Address Word 3 Port 0	0xFFFF	**MAC Address Word 3 Port 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.8.4 MAC Address Word 1 Port 1 (0x0003)

Port 1 MAC Address Word 1.

Bit(s)	Field Name	Default	Description
15:0	MAC Address Word 1 Port 1	0xFFFF	**MAC Address Word 1 Port 1** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.8.5 MAC Address Word 2 Port 1 (0x0004)

Port 1 MAC Address Word 2.

Bit(s)	Field Name	Default	Description
15:0	MAC Address Word 2 Port 1	0xFFFF	**MAC Address Word 2 Port 1** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.8.6 MAC Address Word 3 Port 1 (0x0005)

Port 1 MAC Address Word 3.

Bit(s)	Field Name	Default	Description
15:0	MAC Address Word 3 Port 1	0xFFFF	**MAC Address Word 3 Port 1** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.9 FCoE Scratch Pad Header Section

Table 6-15. FCoE Scratch Pad Header Section Summary Table

Word Offset	Description	Section Number
0x0000	FCoE Scratch Pad Pointer	6.2.9.1
0x0001	FCoE Scratch Pad Size	6.2.9.2

6.2.9.1 FCoE Scratch Pad Pointer (0x0000)

The address is in 4 KB sector index units.

Bit(s)	Field Name	Default	Description
15	Pointer Type	1b	**Pointer Type** 0b = Word units. 1b = 4 KB sector units.
14:0	FCoE Scratch Pad Pointer	0x0	**FCoE Scratch Pad Pointer** Points to FCoE Scratch Pad Section. For FCoE Scratch Pad inner structure, see Section 6.2.29.

6.2.9.2 FCoE Scratch Pad Size (0x0001)

Size of scratch pad (in 4 KB resolution).

Bit(s)	Field Name	Default	Description
15:0	Size		**Size** Length in: 4 KB unit First Section -> Word: FCoE Scratch Pad -> Reserved Last Section -> Word: FCoE Scratch Pad -> Reserved

6.2.10 Active SAN MAC Address Section

Table 6-16. Active SAN MAC Address Section Summary Table

Word Offset	Description	Section Number
0x0000	SAN MAC Address Word 0 Port 0	6.2.10.1
0x0001	SAN MAC Address Word 1 Port 0	6.2.10.2
0x0002	SAN MAC Address Word 2 Port 0	6.2.10.3
0x0003	SAN MAC Address Word 0 Port 1	6.2.10.4
0x0004	SAN MAC Address Word 1 Port 1	6.2.10.5
0x0005	SAN MAC Address Word 2 Port 1	6.2.10.6

6.2.10.1 SAN MAC Address Word 0 Port 0 (0x0000)

Bit(s)	Field Name	Default	Description
15:0	SAN MAC Address Word 0 Port 0	0xFFFF	**SAN MAC Address Word 0 Port 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.10.2 SAN MAC Address Word 1 Port 0 (0x0001)

Bit(s)	Field Name	Default	Description
15:0	SAN MAC Address Word 1 Port 0	0xFFFF	**SAN MAC Address Word 1 Port 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.10.3 SAN MAC Address Word 2 Port 0 (0x0002)

Bit(s)	Field Name	Default	Description
15:0	SAN MAC Address Word 2 Port 0	0xFFFF	**SAN MAC Address Word 2 Port 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.10.4 SAN MAC Address Word 0 Port 1 (0x0003)

Bit(s)	Field Name	Default	Description
15:0	SAN MAC Address Word 0 Port 1	0xFFFF	**SAN MAC Address Word 0 Port 1** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.10.5 SAN MAC Address Word 1 Port 1 (0x0004)

Bit(s)	Field Name	Default	Description
15:0	SAN MAC Address Word 1 Port 1	0xFFFF	**SAN MAC Address Word 1 Port 1** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.10.6 SAN MAC Address Word 2 Port 1 (0x0005)

Bit(s)	Field Name	Default	Description
15:0	SAN MAC Address Word 2 Port 1	0xFFFF	**SAN MAC Address Word 2 Port 1** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.11 Alternate SAN MAC Address Section

Table 6-17. Alternate SAN MAC Address Section Summary Table

Word Offset	Description	Section Number
0x0000	Capabilities	6.2.11.1
0x0001	Alternate SAN MAC Address 1, Lower Word - Port 0	6.2.11.2
0x0002	Alternate SAN MAC Address 1, Middle Word - Port 0	6.2.11.3
0x0003	Alternate SAN MAC Address 1, Upper Word - Port 0	6.2.11.4
0x0004	Alternate SAN MAC Address 2, Lower Word - Port 1	6.2.11.5
0x0005	Alternate SAN MAC Address 2, Middle Word - Port 1	6.2.11.6
0x0006	Alternate SAN MAC Address 2, Upper Word - Port 1	6.2.11.7
0x0007	Alternate WWNN Prefix	6.2.11.8
0x0008	Alternate WWPN Prefix	6.2.11.9

6.2.11.1 Capabilities (0x0000)

Bit(s)	Field Name	Default	Description
15:2	Capabilities	0x0	**Capabilities** ***Note:*** This field is preserved by Intel NVM update tool.
1	Alternate WWNN Base	1b	**Alternate WWNN Base** Alternate WWNN base address (0x7) is available and can be written to. ***Note:*** This field is preserved by Intel NVM update tool.
0	<New Field>	1b	**<New Field>** Alternate SAN MAC Address words (0x1-0x6) are available and can be written to. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.11.2 Alternate SAN MAC Address 1, Lower Word - Port 0 (0x0001)

Bit(s)	Field Name	Default	Description
15:0	Alternate SAN MAC Address 1, Lower Word (Port 0)	0xFFFF	**Alternate SAN MAC Address 1, Lower Word (Port 0)** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.11.3 Alternate SAN MAC Address 1, Middle Word - Port 0 (0x0002)

Bit(s)	Field Name	Default	Description
15:0	Alternate SAN MAC Address 1, Middle Word (Port 0)	0xFFFF	**Alternate SAN MAC Address 1, Middle Word (Port 0)** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.11.4 Alternate SAN MAC Address 1, Upper Word - Port 0 (0x0003)

Bit(s)	Field Name	Default	Description
15:0	Alternate SAN MAC Address 1, Upper Word (Port 0)	0xFFFF	**Alternate SAN MAC Address 1, Upper Word (Port 0)** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.11.5 Alternate SAN MAC Address 2, Lower Word - Port 1 (0x0004)

Bit(s)	Field Name	Default	Description
15:0	Alternate SAN MAC Address 2, Lower Word (Port 1)	0xFFFF	**Alternate SAN MAC Address 2, Lower Word (Port 1)** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.11.6 Alternate SAN MAC Address 2, Middle Word - Port 1 (0x0005)

Bit(s)	Field Name	Default	Description
15:0	Alternate SAN MAC Address 2, Middle Word (Port 1)	0xFFFF	**Alternate SAN MAC Address 2, Middle Word (Port 1)** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.11.7 Alternate SAN MAC Address 2, Upper Word - Port 1 (0x0006)

Bit(s)	Field Name	Default	Description
15:0	Alternate SAN MAC Address 2, Upper Word (Port 1)	0xFFFF	**Alternate SAN MAC Address 2, Upper Word (Port 1)** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.11.8 Alternate WWNN Prefix (0x0007)

Bit(s)	Field Name	Default	Description
15:0	Alternate WWNN Base Address (Port 0)	0xFFFF	**Alternate WWNN Base Address (Port 0)** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.11.9 Alternate WWPN Prefix (0x0008)

Bit(s)	Field Name	Default	Description
15:0	Alternate WWNN Base Address (Port 1)	0xFFFF	**Alternate WWNN Base Address (Port 1)** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12 Boot Configuration Block Section

Contains the required setup to be used for the boot operations.

Table 6-18. Boot Configuration Block Section Summary Table

Word Offset	Description	Section Number
0x0000	Boot Signature	6.2.12.1
0x0001	Block Size	6.2.12.2
0x0002	Structure Version	6.2.12.3
0x0003	iSCSI Initiator Name	6.2.12.4
0x0083	Combo Image Version High	6.2.12.5
0x0084	Combo Image Version Low	6.2.12.6
0x0085 + 1*n, n=0...14	Reserved	6.2.12.7
0x0094	iSCSI Flags	6.2.12.8
0x0095 + 1*n, n=0...1	iSCSI Initiator IP	6.2.12.9
0x0097 + 1*n, n=0...1	Subnet Mask	6.2.12.10
0x0099 + 1*n, n=0...1	Gateway IP	6.2.12.11
0x009B	iSCSI Boot LUN	6.2.12.12
0x009C + 1*n, n=0...1	iSCSI Target IP	6.2.12.13
0x009E	iSCSI Target Port	6.2.12.14
0x009F	iSCSI Target Name	6.2.12.15
0x011F	CHAP Password	6.2.12.16
0x0128	CHAP User Name	6.2.12.17
0x0168	VLAN ID	6.2.12.18
0x0169	Mutual CHAP Password	6.2.12.19
0x0172	FCoE Flags	6.2.12.20
0x0173 + 1*n, n=0...2	Reserved	6.2.12.21
0x0176	Target Worldwide Port Name - WWPN	6.2.12.22
0x017A	Boot LUN	6.2.12.23
0x017B	VLAN ID	6.2.12.24
0x017C	Target Boot Order	6.2.12.25
0x017D	Reserved	6.2.12.26
0x017E	Target Worldwide Port Name - WWPN	6.2.12.27
0x0182	Boot LUN	6.2.12.28
0x0183	VLAN ID	6.2.12.29
0x0184	Target Boot Order	6.2.12.30
0x0185	Reserved	6.2.12.31
0x0186	Target Worldwide Port Name - WWPN	6.2.12.32
0x018A	Boot LUN	6.2.12.33
0x018B	VLAN ID	6.2.12.34

Table 6-18. Boot Configuration Block Section Summary Table [continued]

Word Offset	Description	Section Number
0x018C	Target Boot Order	6.2.12.35
0x018D	Reserved	6.2.12.36
0x018E	Target Worldwide Port Name - WWPN	6.2.12.37
0x0192	Boot LUN	6.2.12.38
0x0193	VLAN ID	6.2.12.39
0x0194	Target Boot Order	6.2.12.40
0x0195 + 1*n, n=0...44	Reserved	6.2.12.41
0x01C2	iSCSI Flags	6.2.12.42
0x01C3 + 1*n, n=0...1	iSCSI Initiator IP	6.2.12.43
0x01C5 + 1*n, n=0...1	Subnet Mask	6.2.12.44
0x01C7 + 1*n, n=0...1	Gateway IP	6.2.12.45
0x01C9	iSCSI Boot LUN	6.2.12.46
0x01CA + 1*n, n=0...1	iSCSI Target IP	6.2.12.47
0x01CC	iSCSI Target Port	6.2.12.48
0x01CD	iSCSI Target Name	6.2.12.49
0x024D	CHAP Password	6.2.12.50
0x0256	CHAP User Name	6.2.12.51
0x0296	VLAN ID	6.2.12.52
0x0297	Mutual CHAP Password	6.2.12.53
0x02A0	FCoE Flags	6.2.12.54
0x02A1 + 1*n, n=0...2	Reserved	6.2.12.55
0x02A4	Target Worldwide Port Name - WWPN	6.2.12.56
0x02A8	Boot LUN	6.2.12.57
0x02A9	VLAN ID	6.2.12.58
0x02AA	Target Boot Order	6.2.12.59
0x02AB	Reserved	6.2.12.60
0x02AC	Target Worldwide Port Name - WWPN	6.2.12.61
0x02B0	Boot LUN	6.2.12.62
0x02B1	VLAN ID	6.2.12.63
0x02B2	Target Boot Order	6.2.12.64
0x02B3	Reserved	6.2.12.65
0x02B4	Target Worldwide Port Name - WWPN	6.2.12.66
0x02B8	Boot LUN	6.2.12.67
0x02B9	VLAN ID	6.2.12.68
0x02BA	Target Boot Order	6.2.12.69
0x02BB	Reserved	6.2.12.70
0x02BC	Target Worldwide Port Name - WWPN	6.2.12.71

Table 6-18. Boot Configuration Block Section Summary Table [continued]

Word Offset	Description	Section Number
0x02C0	Boot LUN	6.2.12.72
0x02C1	VLAN ID	6.2.12.73
0x02C2	Target Boot Order	6.2.12.74
0x02C3 + 1*n, n=0...44	Reserved	6.2.12.75

6.2.12.1 Boot Signature (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Boot Signature	0x5369	**Boot Signature** 'i', 'S' (0x5369) ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.2 Block Size (0x0001)

Bit(s)	Field Name	Default	Description
15:0	Block Size		**Block Size** Length in: 1 Byte unit First Section -> Word: Boot Configuration Block -> Boot Signature Last Section -> Word: Boot Configuration Block -> Reserved ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.3 Structure Version (0x0002)

Bit(s)	Field Name	Default	Description
15:8	Reserved	0x00	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Structure Version	0x01	**Structure Version** Version of this structure. Should be set to 1b. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.4 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 MAC Address.

Note: This word is preserved by Intel NVM update tool.

6.2.12.5 Combo Image Version High (0x0083)

Bit(s)	Field Name	Default	Description
15:8	Major	0x0	**Major** Combo image Major version. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Build	0x0	**Build** Combo image Build number high (bits 15:8). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.6 Combo Image Version Low (0x0084)

Bit(s)	Field Name	Default	Description
15:8	Build	0x0	**Build** Combo image Build number low (bits 7:0). ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Patch	0x0	**Patch** Combo image Patch level. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.7 Reserved[n] (0x0085 + 1*n, n=0...14)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved for future use, should be set to zero. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.8 iSCSI Flags (0x0094)

Bit(s)	Field Name	Default	Description
15:10	ARP Timeout	0xF	**ARP Timeout** Timeout value for each try. ***Note:*** This field is preserved by Intel NVM update tool.
9:8	ARP Retries	01b	**ARP Retries** Retry value. ***Note:*** This field is preserved by Intel NVM update tool.
7:6	Reserved	00b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
5:4	Ctrl-D Entry	00b	**Ctrl-D Entry** 00b = Enabled 11b = Disabled — Skip Ctrl-D entry. All other values are reserved. ***Note:*** This field is preserved by Intel NVM update tool.
3:2	Authentication Type	00b	**Authentication Type** 00b = None 01b = One Way CHAP 10b = Mutual CHAP 11b = Reserved ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
1	Enable DHCP for iSCSI Target	1b	**Enable DHCP for iSCSI Target** Enable DHCP for getting iSCSI target information. 0b = Disabled — Use static target configuration. 1b = Enabled — Use DHCP to get target information by the Option 17 Root Path. ***Note:*** This field is preserved by Intel NVM update tool.
0	Enable DHCP	1b	**Enable DHCP** 0b = Disabled — Use static configurations from this structure. 1b = Enabled — Overrides configurations retrieved from DHCP. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.9 iSCSI Initiator IP[n] (0x0095 + 1*n, n=0...1)

Bit(s)	Field Name	Default	Description
15:0	iSCSI Initiator IP	0x0	**iSCSI Initiator IP** Initiator DHCP flag. Not set = This field should contain the initiator IP Address. Set = This field is ignored. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.10 Subnet Mask[n] (0x0097 + 1*n, n=0...1)

Bit(s)	Field Name	Default	Description
15:0	iSCSI Initiator IP	0x0	**iSCSI Initiator IP** Initiator DHCP flag. Not set = This field should contain the subnet mask. Set = This field is ignored. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.11 Gateway IP[n] (0x0099 + 1*n, n=0...1)

Bit(s)	Field Name	Default	Description
15:0	iSCSI Initiator IP	0x0	**iSCSI Initiator IP** Initiator DHCP flag. Not set = This field should contain the gateway IP Address. Set = This field is ignored. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.12 iSCSI Boot LUN (0x009B)

Bit(s)	Field Name	Default	Description
15:0	iSCSI Boot LUN	0x0	**iSCSI Boot LUN** Target DHCP flag. Not set = iSCSI target LUN number should be specified. Set = This field is ignored. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.13 iSCSI Target IP[n] (0x009C + 1*n, n=0...1)

Bit(s)	Field Name	Default	Description
15:0	iSCSI Initiator IP	0x0	**iSCSI Initiator IP** Target DHCP flag. Not set = IP Address of iSCSI target. Set = This field is ignored. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.14 iSCSI Target Port (0x009E)

Bit(s)	Field Name	Default	Description
15:0	iSCSI Target Port	0x0CBC	**iSCSI Target Port** Target DHCP flag. Not set = TCP port used by iSCSI target. Default is 3260. Set = This field is ignored. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.15 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.

Note: This word is preserved by Intel NVM update tool.

6.2.12.16 CHAP Password (0x011F)

Raw data module length: 9 words

The minimum CHAP secret must be 12 octets and maximum CHAP secret size is 16. The last 2 bytes are null alignment padding.

Note: This word is preserved by Intel NVM update tool.

6.2.12.17 CHAP User Name (0x0128)

Raw data module length: 64 words

The user name must be non-null value and maximum size of user name allowed is 127 characters.

Note: This word is preserved by Intel NVM update tool.

6.2.12.18 VLAN ID (0x0168)

Bit(s)	Field Name	Default	Description
15:0	VLAN ID	0x0	**VLAN ID** Reserved area since the function is disabled due to Microsoft restrictions. VLAN ID to include the tag in iSCSI boot frames. A valid VLAN ID is between 1 and 4094. Zero means no VLAN tag support. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.19 Mutual CHAP Password (0x0169)

Raw data module length: 9 words

The minimum mutual CHAP secret must be 12 octets and maximum mutual CHAP secret size is 16. The last 2 bytes are null alignment padding.

Note: This word is preserved by Intel NVM update tool.

6.2.12.20 FCoE Flags (0x0172)

Bit(s)	Field Name	Default	Description
15:2	Reserved	0x0	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
1	Disable FCoE Ctrl-D Menu	0b	**Disable FCoE Ctrl-D Menu** 0b = Enabled — FCoE Ctrl-D menu is enabled and user can configure FCoE ports. 1b = Disabled — FCoE Ctrl-D menu is disabled and user cannot configure FCoE ports. ***Note:*** This field is preserved by Intel NVM update tool.
0	Reserved	0b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.21 Reserved[n] (0x0173 + 1*n, n=0...2)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved for future use, should be set to zero. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.22 Target Worldwide Port Name - WWPN (0x0176)

Raw data module length: 4 words

Byte string of target WWPN.

Note: This word is preserved by Intel NVM update tool.

6.2.12.23 Boot LUN (0x017A)

Bit(s)	Field Name	Default	Description
15:0	Target LUN	0x0	**Target LUN** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.24 VLAN ID (0x017B)

Bit(s)	Field Name	Default	Description
15:0	VLAN ID	0x0	**VLAN ID** VLAN ID for the Port. Default is 0. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.25 Target Boot Order (0x017C)

Bit(s)	Field Name	Default	Description
15:8	Reserved	0x0	Reserved for future use, should be set to zero. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Target Boot Order	0x0	**Target Boot Order** Valid range is 0-4, with 0 meaning no boot order. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.26 Reserved (0x017D)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.27 Target Worldwide Port Name - WWPN (0x017E)

Raw data module length: 4 words

Byte string of target WWPN.

Note: This word is preserved by Intel NVM update tool.

6.2.12.28 Boot LUN (0x0182)

Bit(s)	Field Name	Default	Description
15:0	Target LUN	0x0	**Target LUN** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.29 VLAN ID (0x0183)

Bit(s)	Field Name	Default	Description
15:0	VLAN ID	0x0	**VLAN ID** VLAN ID for the Port. Default is 0. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.30 Target Boot Order (0x0184)

Bit(s)	Field Name	Default	Description
15:8	Reserved	0x0	Reserved for future use, should be set to zero. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Target Boot Order	0x0	**Target Boot Order** Valid range is 0-4, with 0 meaning no boot order. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.31 Reserved (0x0185)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.32 Target Worldwide Port Name - WWPN (0x0186)

Raw data module length: 4 words

Byte string of target WWPN.

Note: This word is preserved by Intel NVM update tool.

6.2.12.33 Boot LUN (0x018A)

Bit(s)	Field Name	Default	Description
15:0	Target LUN	0x0	**Target LUN** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.34 VLAN ID (0x018B)

Bit(s)	Field Name	Default	Description
15:0	VLAN ID	0x0	**VLAN ID** VLAN ID for the Port. Default is 0. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.35 Target Boot Order (0x018C)

Bit(s)	Field Name	Default	Description
15:8	Reserved	0x0	Reserved for future use, should be set to zero. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Target Boot Order	0x0	**Target Boot Order** Valid range is 0-4, with 0 meaning no boot order. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.36 Reserved (0x018D)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.37 Target Worldwide Port Name - WWPN (0x018E)

Raw data module length: 4 words

Byte string of target WWPN.

Note: This word is preserved by Intel NVM update tool.

6.2.12.38 Boot LUN (0x0192)

Bit(s)	Field Name	Default	Description
15:0	Target LUN	0x0	**Target LUN** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.39 VLAN ID (0x0193)

Bit(s)	Field Name	Default	Description
15:0	VLAN ID	0x0	**VLAN ID** VLAN ID for the Port. Default is 0. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.40 Target Boot Order (0x0194)

Bit(s)	Field Name	Default	Description
15:8	Reserved	0x0	Reserved for future use, should be set to zero. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Target Boot Order	0x0	**Target Boot Order** Valid range is 0-4, with 0 meaning no boot order. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.41 Reserved[n] (0x0195 + 1*n, n=0...44)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.12.42 iSCSI Flags (0x01C2)

For inner structure, see Section 6.2.12.8.

Note: This word is preserved by Intel NVM update tool.

6.2.12.43 iSCSI Initiator IP[n] (0x01C3 + 1*n, n=0...1)

For inner structure, see Section 6.2.12.9.

Note: This word is preserved by Intel NVM update tool.

6.2.12.44 Subnet Mask[n] (0x01C5 + 1*n, n=0...1)

For inner structure, see Section 6.2.12.10.

Note: This word is preserved by Intel NVM update tool.

6.2.12.45 Gateway IP[n] (0x01C7 + 1*n, n=0...1)

For inner structure, see Section 6.2.12.11.

Note: This word is preserved by Intel NVM update tool.

6.2.12.46 iSCSI Boot LUN (0x01C9)

For inner structure, see Section 6.2.12.12.

Note: This word is preserved by Intel NVM update tool.

6.2.12.47 iSCSI Target IP[n] (0x01CA + 1*n, n=0...1)

For inner structure, see Section 6.2.12.13.

Note: This word is preserved by Intel NVM update tool.

6.2.12.48 iSCSI Target Port (0x01CC)

For inner structure, see Section 6.2.12.14.

Note: This word is preserved by Intel NVM update tool.

6.2.12.49 iSCSI Target Name (0x01CD)

For inner structure, see Section 6.2.12.15.

Note: This word is preserved by Intel NVM update tool.

6.2.12.50 CHAP Password (0x024D)

For inner structure, see Section 6.2.12.16.

Note: This word is preserved by Intel NVM update tool.

6.2.12.51 CHAP User Name (0x0256)

For inner structure, see Section 6.2.12.17.

Note: This word is preserved by Intel NVM update tool.

6.2.12.52 VLAN ID (0x0296)

For inner structure, see Section 6.2.12.18.

Note: This word is preserved by Intel NVM update tool.

6.2.12.53 Mutual CHAP Password (0x0297)

For inner structure, see Section 6.2.12.19.

Note: This word is preserved by Intel NVM update tool.

6.2.12.54 FCoE Flags (0x02A0)

For inner structure, see Section 6.2.12.20.

Note: This word is preserved by Intel NVM update tool.

6.2.12.55 Reserved[n] (0x02A1 + 1*n, n=0...2)

For inner structure, see Section 6.2.12.21.

Note: This word is preserved by Intel NVM update tool.

6.2.12.56 Target Worldwide Port Name - WWPN (0x02A4)

For inner structure, see Section 6.2.12.22.

Note: This word is preserved by Intel NVM update tool.

6.2.12.57 Boot LUN (0x02A8)

For inner structure, see Section 6.2.12.23.

Note: This word is preserved by Intel NVM update tool.

6.2.12.58 VLAN ID (0x02A9)

For inner structure, see Section 6.2.12.24.

Note: This word is preserved by Intel NVM update tool.

6.2.12.59 Target Boot Order (0x02AA)

For inner structure, see Section 6.2.12.25.

Note: This word is preserved by Intel NVM update tool.

6.2.12.60 Reserved (0x02AB)

For inner structure, see Section 6.2.12.26.

Note: This word is preserved by Intel NVM update tool.

6.2.12.61 Target Worldwide Port Name - WWPN (0x02AC)

For inner structure, see Section 6.2.12.27.

Note: This word is preserved by Intel NVM update tool.

6.2.12.62 Boot LUN (0x02B0)

For inner structure, see Section 6.2.12.28.

Note: This word is preserved by Intel NVM update tool.

6.2.12.63 VLAN ID (0x02B1)

For inner structure, see Section 6.2.12.29.

Note: This word is preserved by Intel NVM update tool.

6.2.12.64 Target Boot Order (0x02B2)

For inner structure, see Section 6.2.12.30.

Note: This word is preserved by Intel NVM update tool.

6.2.12.65 Reserved (0x02B3)

For inner structure, see Section 6.2.12.31.

Note: This word is preserved by Intel NVM update tool.

6.2.12.66 Target Worldwide Port Name - WWPN (0x02B4)

For inner structure, see Section 6.2.12.32.

Note: This word is preserved by Intel NVM update tool.

6.2.12.67 Boot LUN (0x02B8)

For inner structure, see Section 6.2.12.33.

Note: This word is preserved by Intel NVM update tool.

6.2.12.68 VLAN ID (0x02B9)

For inner structure, see Section 6.2.12.34.

Note: This word is preserved by Intel NVM update tool.

6.2.12.69 Target Boot Order (0x02BA)

For inner structure, see Section 6.2.12.35.

Note: This word is preserved by Intel NVM update tool.

6.2.12.70 Reserved (0x02BB)

For inner structure, see Section 6.2.12.36.

Note: This word is preserved by Intel NVM update tool.

6.2.12.71 Target Worldwide Port Name - WWPN (0x02BC)

For inner structure, see Section 6.2.12.37.

Note: This word is preserved by Intel NVM update tool.

6.2.12.72 Boot LUN (0x02C0)

For inner structure, see Section 6.2.12.38.

Note: This word is preserved by Intel NVM update tool.

6.2.12.73 VLAN ID (0x02C1)

For inner structure, see Section 6.2.12.39.

Note: This word is preserved by Intel NVM update tool.

6.2.12.74 Target Boot Order (0x02C2)

For inner structure, see Section 6.2.12.40.

Note: This word is preserved by Intel NVM update tool.

6.2.12.75 Reserved[n] (0x02C3 + 1*n, n=0...44)

For inner structure, see Section 6.2.12.41.

Note: This word is preserved by Intel NVM update tool.

6.2.13 Firmware Module Header Section

Table 6-19. Firmware Module Header Section Summary Table

Word Offset	Description	Section Number
0x0000	Section Header	6.2.13.1
0x0001	Test Configuration Pointer	6.2.13.2
0x0002	Common Firmware Parameters Pointer	6.2.13.3
0x0003	Pass-Through LAN 0 Configuration Pointer	6.2.13.4
0x0004	Sideband Configuration Pointer	6.2.13.5
0x0005	Flexible TCO Filter Configuration Pointer	6.2.13.6
0x0006	Pass-Through LAN 1 Configuration Pointer	6.2.13.7
0x0007	OEM Support Structure Pointer	6.2.13.8
0x0008	LESM Configuration Pointer	6.2.13.9
0x0009 - 0x000A	Reserved	6.2.13.10
0x000B	Section Footer	6.2.13.11

6.2.13.1 Section Header (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Block Length		**Block Length** Length in words of the section covered by CRC.

6.2.13.2 Test Configuration Pointer (0x0001)

Bit(s)	Field Name	Default	Description
15:0	Test Configuration Pointer	0x0	**Test Configuration Pointer** Points to Test Configuration Module Section. For Test Configuration Module inner structure, see Section 6.2.15.

6.2.13.3 Common Firmware Parameters Pointer (0x0002)

Bit(s)	Field Name	Default	Description
15:0	Common Firmware Parameters Pointer	0x0	**Common Firmware Parameters Pointer** Points to Common Firmware Parameters Module Section. For Common Firmware Parameters Module inner structure, see Section 6.2.16.

6.2.13.4 Pass-Through LAN 0 Configuration Pointer (0x0003)

Bit(s)	Field Name	Default	Description
15:0	Pass Through LAN 0 Configuration Pointer	0x0	**Pass-Through LAN 0 Configuration Pointer** Points to Pass Through Control Words Section. For Pass Through Control Words inner structure, see Section 6.2.18.

6.2.13.5 Sideband Configuration Pointer (0x0004)

This module is long by 28 bytes and must be mapped in the first valid 4 KB sector of the Flash.

Bit(s)	Field Name	Default	Description
15:0	Sideband Configuration Pointer	0x0	**Sideband Configuration Pointer** Points to Sideband Configuration Structure Section. For Sideband Configuration Structure inner structure, see Section 6.2.17.

6.2.13.6 Flexible TCO Filter Configuration Pointer (0x0005)

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.

Bit(s)	Field Name	Default	Description
15:0	Flexible TCO Filter Configuration Pointer	0x0	**Flexible TCO Filter Configuration Pointer** Points to Flexible TCO Filter Configuration Structure Section. For Flexible TCO Filter Configuration Structure inner structure, see Section 6.2.19.

6.2.13.7 Pass-Through LAN 1 Configuration Pointer (0x0006)

Bit(s)	Field Name	Default	Description
15:0	Pass Through LAN 1 Configuration Pointer	0x0	**Pass Through LAN 1 Configuration Pointer** Points to Pass Through Control Words Section. For Pass Through Control Words inner structure, see Section 6.2.18.

6.2.13.8 OEM Support Structure Pointer (0x0007)

Bit(s)	Field Name	Default	Description
15:0	OEM Support Structure Pointer	0x0	**OEM Support Structure Pointer**

6.2.13.9 LESM Configuration Pointer (0x0008)

Bit(s)	Field Name	Default	Description
15:0	LESM Configuration Pointer	0x0	**LESM Configuration Pointer** Points to LESM Configurations (not in SGVL) Section. For LESM Configurations (not in SGVL) inner structure, see Section 6.2.20.

6.2.13.10 Reserved (0x0009 - 0x000A)

6.2.13.11 Section Footer (0x000B)

Bit(s)	Field Name	Default	Description
15:8	Block CRC8		**Block CRC8** CRC-8-CCITT: Start Section -> Word: Firmware Module Header -> Section Header End Section -> Word: Firmware Module Header -> OEM HP Support Structure Pointer
7:0	Reserved	0x0	Reserved. Block length in words.

6.2.14 Firmware Header Reserved Word Section

Placeholder to make header offsets following be the same in BDX/MGPK (they have LESM, one word longer Firmware module header).

Table 6-20. Firmware Header Reserved Word Section Summary Table

Word Offset	Description	Section Number
0x0000	Reserved	6.2.14.1

6.2.14.1 Reserved (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.15 Test Configuration Module Section

Table 6-21. Test Configuration Module Section Summary Table

Word Offset	Description	Section Number
0x0000	Section Header	6.2.15.1
0x0001	Reserved	6.2.15.2
0x0002	Miscellaneous	6.2.15.3
0x0003	Section Footer	6.2.15.4

6.2.15.1 Section Header (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Block Length		**Block Length** Length in words of the section covered by CRC. Block length in words.

6.2.15.2 Reserved (0x0001)

6.2.15.3 Miscellaneous (0x0002)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.15.4 Section Footer (0x0003)

Bit(s)	Field Name	Default	Description
15:8	Block CRC		**Block CRC** CRC-8-CCITT: Start Section -> Word: Test Configuration Module -> Section Header En Section -> Word: Test Configuration Module -> Miscellaneous
7:0	Reserved	0x0	Reserved. Block length in words.

6.2.16 Common Firmware Parameters Module Section

Table 6-22. Common Firmware Parameters Module Section Summary Table

Word Offset	Description	Section Number
0x0000	Section Header	6.2.16.1
0x0001	Common Firmware Parameters 1	6.2.16.2
0x0002	Common Firmware Parameters 2	6.2.16.3
0x0003	Section Footer	6.2.16.4

6.2.16.1 Section Header (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Block Length		**Block Length** Length in words of the section covered by CRC. Block length in words. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.16.2 Common Firmware Parameters 1 (0x0001)

Bit(s)	Field Name	Default	Description
15	Enable Firmware Reset	1b	**Enable Firmware Reset** 0b = Disabled 1b = Enabled ***Note:*** This field is preserved by Intel NVM update tool.
14:13	Redirection Sideband Interface	00b	**Redirection Sideband Interface** 00b = SMBus 01b = NC-SI 10b = PT Disable 11b = MCTP (over PCIe and SMBus) ***Note:*** This field is preserved by Intel NVM update tool.
12	Restore MAC Address	1b	**Restore MAC Address** 0b = Do not restore MAC Address at power-on. 1b = Restore MAC Address at power-on. ***Note:*** This field is preserved by Intel NVM update tool.
11	Reserved	1b	Reserved.
10:8	Manageability Pass Through Mode	010b	**Manageability Pass-Through Mode** 000b = None 010b = Pass Through (PT) Mode All other values are reserved. ***Note:*** This field is preserved by Intel NVM update tool.
7:6	Control Interface	00b	**Control Interface** 00b = None 01b = MCTP over SMBus only 10b = MCTP over PCIe only 11b = MCTP over PCIe or SMBus ***Note:*** This field is preserved by Intel NVM update tool.
5:3	Reserved	111b	Reserved.

Bit(s)	Field Name	Default	Description
2	OS2BMC Capable	0b	**OS2BMC Capable** 0b = Disabled 1b = Enabled ***Note:*** This field is preserved by Intel NVM update tool.
1	LAN1 TCO Isolate Disable	1b	**LAN1 TCO Isolate Disable** 0b = Enabled 1b = Disabled ***Note:*** This field is preserved by Intel NVM update tool.
0	LAN0 TCO Isolate Disable	1b	**LAN0 TCO Isolate Disable** 0b = Enabled 1b = Disabled ***Note:*** This field is preserved by Intel NVM update tool.

6.2.16.3 Common Firmware Parameters 2 (0x0002)

Bit(s)	Field Name	Default	Description
15:13	Reserved	000b	Reserved.
12	Reserved	0b	Reserved.
11	Multi-Drop NC-SI	1b	**Multi-Drop NC-SI Topology** When this bit is set, the NCSI_CRS_DV and NCSI_RXD[1:0] pins are High-Z following power-up. Otherwise, the pins are driven. 0b = Point-to-Point 1b = Multi-drop (default) ***Note:*** This field is preserved by Intel NVM update tool.
10	Proxying Capable	0b	**Proxying Capable** 0b = Disable Protocol Offload. 1b = Enable Protocol Offload. ***Note:*** This field is preserved by Intel NVM update tool.
9	Reserved	0b	Reserved.
8	PLDM Over MCTP	0b	**PLDM Over MCTP** 0b = Disabled 1b = Enabled
7	OEM Commands Over MCTP	0b	**OEM Commands Over MCTP** 0b = Disabled 1b = Enabled
6	NC-SI Over MCTP	0b	**NC-SI Over MCTP** Reserved. 0b = Disabled 1b = Enabled
5:1	Semaphore Backoff Interval	0x2	**Semaphore Backoff Interval** Number of 10 ms ticks that firmware must wait before taking semaphore ownership again since it has released it. ***Note:*** This field is preserved by Intel NVM update tool.
0	Reserved	0b	Reserved.

6.2.16.4 Section Footer (0x0003)

Bit(s)	Field Name	Default	Description
15:8	Block CRC		**Block CRC** CRC-8-CCITT: Start Section -> Word: Common Firmware Parameters Module -> Section Header End Section -> Word: Common Firmware Parameters Module -> Common Firmware Parameters 2 ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Reserved	0x0	Reserved. Block length in words.

6.2.17 Sideband Configuration Structure Section

Table 6-23. Sideband Configuration Structure Section Summary Table

Word Offset	Description	Section Number
0x0000	Block Length	6.2.17.1
0x0001	SMBus Maximum Fragment Size	6.2.17.2
0x0002	SMBus Notification Timeout and Flags	6.2.17.3
0x0003	SMBus Slave Addresses	6.2.17.4
0x0004	NC-SI Configuration 1	6.2.17.5
0x0005	NC-SI Configuration 2	6.2.17.6
0x0006	NCSI Flow Control XOFF	6.2.17.7
0x0007	NCSI Flow Control XON	6.2.17.8
0x0008	NC-SI HW Arbitration TOKEN Timeout	6.2.17.9
0x0009 - 0x000D	Reserved	6.2.17.10
0x000E	OEM IANA	6.2.17.11
0x000F	NC-SI over MCTP Message Types	6.2.17.12
0x0010	NC-SI over MCTP Configuration	6.2.17.13
0x0011	Traffic Types Parameters	6.2.17.14
0x0012	MCTP Rate Limiter Config 1	6.2.17.15
0x0013	MCTP Rate Limiter Config 2	6.2.17.16
0x0014	NC-SI Channel to Port Mapping	6.2.17.17
0x0015	Block CRC8	6.2.17.18

6.2.17.1 Block Length (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Block Length	0x13	**Block Length** Length in words of the section covered by CRC. Section length in words. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.2 SMBus Maximum Fragment Size (0x0001)

Bit(s)	Field Name	Default	Description
15:0	Fragment Size	0x20	**Fragment Size** 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) ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.3 SMBus Notification Timeout and Flags (0x0002)

Bit(s)	Field Name	Default	Description
15:8	SMBus Notification Timeout (ms)	0xFF	**SMBus Notification Timeout (ms)** ***Note:*** This field is preserved by Intel NVM update tool.
7:6	SMBus Connection Speed	00b	**SMBus Connection Speed** 00b = Standard SMBus connection. 01b = Reserved 10b = Reserved 11b = Reserved ***Note:*** This field is preserved by Intel NVM update tool.
5	SMBus Block Read Command	0b	**SMBus Block Read Command** 0b = Block read command is 0xC0. 1b = Block read command is 0xD0. ***Note:*** This field is preserved by Intel NVM update tool.
4	Reserved	1b	Reserved. Must be 1b. 0b = Single address mode. 1b = Dual address mode.
3	Reserved	0b	Reserved.
2	Disable SMBus ARP Functionality	0b	**Disable SMBus ARP Functionality** ***Note:*** This field is preserved by Intel NVM update tool.
1	SMBus ARP PEC	1b	**SMBus ARP PEC** SMBus Transactions PEC. Should be set. 0b = Disable SMBus ARP PEC. 1b = Enable SMBus ARP PEC. ***Note:*** This field is preserved by Intel NVM update tool.
0	SMBus Transaction PEC	0b	**SMBus Transactions PEC** Should be set in MCTP modes. 0b = 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. 1b = Enable PEC — If this bit is set, PEC is added for master SMBus write transactions. a PEC is added to slave read transactions and can be received in slave write transaction. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.4 SMBus Slave Addresses (0x0003)

Bit(s)	Field Name	Default	Description
15:9	SMBus 1 Slave Address	0x62	**SMBus 1 Slave Address** Dual address mode only. ***Note:*** This field is preserved by Intel NVM update tool.
8	Reserved	0x0	Reserved.
7:1	SMBus 0 Slave Address	0x63	**SMBus 0 Slave Address** ***Note:*** This field is preserved by Intel NVM update tool.
0	Reserved	0x0	Reserved.

6.2.17.5 NC-SI Configuration 1 (0x0004)

Bit(s)	Field Name	Default	Description
15	Reserved	0b	Reserved.
14:11	NC-SI Version	0x0	**NC-SI Version** 0x0 = NCSI 1.0 0x1 = NCSI 1.1 All other values are reserved. ***Note:*** This field is preserved by Intel NVM update tool.
10	Flow Control	0b	**Flow Control** 0b = Disable 1b = Enable ***Note:*** This field is preserved by Intel NVM update tool.
9	RMII HW Arbitration Enable	0b	**RMII Hardware Arbitration Enable** Must be 0. 0b = Not supported 1b = Supported ***Note:*** This field is preserved by Intel NVM update tool.
8	RMII HW-Based Packet Copy Enable	1b	**RMII Hardware-Based Packet Copy Enable** 0b = Disable 1b = Enable ***Note:*** This field is preserved by Intel NVM update tool.
7:5	Package ID	000b	**Package ID** Meaningful only when bit 15 of NC-SI Configuration 2 word (offset 0x07) is cleared. ***Note:*** This field is preserved by Intel NVM update tool.
4:0	LAN 0 Internal Channel ID	0x0	**LAN 0 Internal Channel ID** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.6 NC-SI Configuration 2 (0x0005)

Bit(s)	Field Name	Default	Description
15	Read NC-SI Package ID from SDP	0b	**Read NC-SI Package ID from SDP** 0b = Disabled — NVM, NC-SI Configuration 1 word bits 7:5 (offset 0x06) 1b = Enabled — SDP, SDPn_0 pins of LAN ports n=1,0 (where SDP1_0, SDP0 of LAN port 1 is the MS bit of NC-SI Package ID). ***Note:*** This field is preserved by Intel NVM update tool.
14	NC-SI Package ID from SDP Workaround	0b	**NC-SI Package ID from SDP Workaround** 0b = No Workaround — NC-SI Package ID should be configured according to value of *Read NC-SI Package ID* from SDP field (Bit 15 of this word). 1b = SDP-SDn_0 pin of LAN port 0, where {0,0,SDP0_0} is the LS bits of NC-SI Package ID.
13:4	Reserved	0x0	Reserved.

Bit(s)	Field Name	Default	Description
3:0	Max XOFF Renewal	0x6	**Max XOFF Renewal** NC-SI Flow Control MAX XOFF Renewal (# of XOFF renewals allowed). 0x0 = Disabled (unlimited allowed) 0x1 = Up to 2 consecutive 0x2 = Up to 0x3 consecutive 0x3 = Up to 0x4 consecutive 0x4 = Up to 0x5 consecutive 0x5 = Up to 0x6 consecutive 0x6 = Up to 0x7 consecutive 0x7 = Up to 0x8 consecutive 0x8 = Up to 0x9 consecutive 0x9 = Up to 0xA consecutive 0xA = Up to 0xB consecutive 0xB = Up to 0xC consecutive 0xC = Up to 0xD consecutive 0xD = Up to 0xE consecutive 0xE = Up to 0xF consecutive 0xF = Up to 0x10 consecutive ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.7 NCSI Flow Control XOFF (0x0006)

Bit(s)	Field Name	Default	Description
15:0	XOFF Threshold	0xAB8	**XOFF Threshold** Tx buffer watermark for sending a XOFF NC-SI flow control packet in bytes. The XOFF Threshold value refers to the occupied space in the buffer. The value should be 16 bytes aligned. ***Notes:*** 1. Field relevant for NC-SI operation mode only. 2. To support a maximum packet size of 1.5 KB, the value programmed assuming a Tx buffer size of 8 KB value of field should be 0x12C0 (4,800 bytes). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.8 NCSI Flow Control XON (0x0007)

Bit(s)	Field Name	Default	Description
15:0	XON Threshold	0x1340	**XON Threshold** Tx buffer water mark for sending a XON NC-SI flow control packet in bytes. The XON Threshold value refers to the available space in the TX buffer. The value should be 16 bytes aligned. ***Notes:*** 1. Field relevant for NC-SI operation mode only. 2. To support maximum packet size of 1.5 KB, the value programmed should be a positive value that equals: Buffer size - XOFF Threshold (refer to Section 6.5.5.7) + 1536 bytes. Assuming a TX Buffer size is 8 KB and the XOFF Threshold is 4800 bytes value of field should be 0x1340 (4,928 bytes). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.9 NC-SI HW Arbitration TOKEN Timeout (0x0008)

Bit(s)	Field Name	Default	Description
15:0	NC-SI HW Arbitration TOKEN Timeout	0xA000	**NC-SI Hardware Arbitration TOKEN Timeout** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.10 Reserved (0x0009 - 0x000D)

6.2.17.11 OEM IANA (0x000E)

Bit(s)	Field Name	Default	Description
15:0	OEM IANA	0x0	**OEM IANA** If not zero and not 0x157, the X550 will accept NC-SI OEM commands with this IANA number. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.12 NC-SI Over MCTP Message Types (0x000F)

Bit(s)	Field Name	Default	Description
15:8	NC-SI Command Packet Type	0x2	**NC-SI Command Packet Type** Defines the MCTP packet type used to identify NC-SI Control Packets. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	NC-SI Pass Through Packet Type	0x3	**NC-SI Pass Through Packet Type** Defines the MCTP packet type used to identify NC-SI Pass Through Packets. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.13 NC-SI Over MCTP Configuration (0x0010)

Bit(s)	Field Name	Default	Description
15:8	Reserved	0x0	Reserved.
7	Use Payload Type	1b	**Use Payload Type** If set, a payload type byte is expected in NC-SI over MCTP packets after the packet type. Otherwise, the control and pass through are controlled via the NC-SI over MCTP Message Types defined in the previous word. ***Note:*** This field is preserved by Intel NVM update tool.
6	Simplified MCTP	0b	**Simplified MCTP** If set, only SOM & EOM bits are used for the reassembly process. Relevant only in SMBus mode. ***Note:*** This field is preserved by Intel NVM update tool.
5	Disable ACLs	1b	**Disable ACLs** If set, the ACLs on the PCIe VDMs are disabled. ***Note:*** This field is preserved by Intel NVM update tool.
4:1	Reserved	0x0	Reserved.

Bit(s)	Field Name	Default	Description
0	Endpoint Discovery Route	0b	**Endpoint Discovery Route** Use Route by ID for Endpoint Discovery responses 0b = To Root Complex — Use "Route to root complex" type in outgoing messages and "Broadcast from root complex" in incoming messages. 1b = By ID — Use "Route by ID" for MCTP discovery messages. This mode should be selected only in debug mode. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.14 Traffic Types Parameters (0x0011)

Bit(s)	Field Name	Default	Description
15:6	Reserved	0x0	Reserved.
5:4	Port1 Traffic Types	01b	**Port1 Traffic Types** 00b = Reserved. 01b = Network to BMC traffic only allowed through Port 1. 10b = OS2BMC traffic only allowed through Port 1. 11b = Both Network to BMC traffic and OS2BMC traffic allowed through Port 1. This field is valid only if the Port1 Manageability Capable field in the Common Firmware Parameters NVM word is set. ***Note:*** This field is preserved by Intel NVM update tool.
3:2	Reserved	00b	Reserved.
1:0	Port 0 Traffic Types	01b	**Port 0 Traffic Types** 00b = Reserved. 01b = Network to BMC traffic only allowed through Port 0. 10b = OS2BMC traffic only allowed through Port 0. 11b = Both Network to BMC traffic and OS2BMC traffic allowed through Port 0. This field is valid only if the Port0 Manageability Capable field in the Common Firmware Parameters NVM word is set. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.15 MCTP Rate Limiter Config 1 (0x0012)

Bit(s)	Field Name	Default	Description
15:0	MCTP Rate	0x2800	**MCTP Rate** Defines the number of cycles between accesses of the MCTP send client to the memory arbiter. Default value assumes a clock of 80 MHz and a bus width of 128 bit. This value provides a bit rate of 1 Mb/s. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.16 MCTP Rate Limiter Config 2 (0x0013)

Bit(s)	Field Name	Default	Description
15	Decision Point	0b	**Decision Point** Defines if, when credits are available, a full MCTP message is sent or a single VDM is sent. 0b = Per VDM 1b = Per Packet ***Note:*** This field is preserved by Intel NVM update tool.
14:0	MCTP Max Credits	0x5	**MCTP Max Credits** 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). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.17 NC-SI Channel to Port Mapping (0x0014)

Bit(s)	Field Name	Default	Description
15	Table Valid	0b	**Table Valid** If set, the values below are used to define the port to channel mapping. ***Note:*** This field is preserved by Intel NVM update tool.
14:7	Reserved	0x0	Reserved.
6:5	Port 1 Channel ID	01b	**Port 1 Channel ID** The Channel ID to use for port 1. ***Note:*** This field is preserved by Intel NVM update tool.
4	Port 1 Channel Enable	1b	**Port 1 Channel Enable** 0b = Disabled 1b = Enabled ***Note:*** This field is preserved by Intel NVM update tool.
3	Reserved	0b	Reserved.
2:1	Port 0 Channel ID	00b	**Port 0 Channel ID** The Channel ID to use for port 1. ***Note:*** This field is preserved by Intel NVM update tool.
0	Port 0 Channel Enable	1b	**Port 0 Channel Enable** 0b = Disabled 1b = Enabled ***Note:*** This field is preserved by Intel NVM update tool.

6.2.17.18 Block CRC8 (0x0015)

Bit(s)	Field Name	Default	Description
15:8	Block CRC8		**Block CRC8** CRC-8-CCITT: Start Section -> Word: Sideband Configuration Structure -> Block Length End Section -> Word: Sideband Configuration Structure -> NC-SI Channel to Port Mapping ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Reserved	0x0	Reserved. Block length in words.

6.2.18 Pass-Through Control Words Section

Table 6-24. Pass-Through Control Words Section Summary Table

Word Offset	Description	Section Number
0x0000	Block Length	6.2.18.1
0x0001	LAN IPv4 Address 0 (LSB) MIPAF0	6.2.18.2
0x0002	LAN IPv4 Address 0 (MSB) MIPAF0	6.2.18.3
0x0003	LAN IPv4 Address 1 (LSB) MIPAF1	6.2.18.4
0x0004	LAN IPv4 Address 1 (MSB) MIPAF1	6.2.18.5
0x0005	LAN IPv4 Address 2 (LSB) MIPAF2	6.2.18.6
0x0006	LAN IPv4 Address 2 (MSB) MIPAF2	6.2.18.7
0x0007	LAN IPv4 Address 3 (LSB) MIPAF3	6.2.18.8
0x0008	LAN IPv4 Address 3 (MSB) MIPAF4	6.2.18.9
0x0009	LAN Ethernet MAC Address 0 (LSB) MMAL0	6.2.18.10
0x000A	LAN Ethernet MAC Address 0 (Mid) MMAL0	6.2.18.11
0x000B	LAN Ethernet MAC Address 0 (MSB) MMAH0	6.2.18.12
0x000C	LAN Ethernet MAC Address 1 (LSB) MMAL1	6.2.18.13
0x000D	LAN Ethernet MAC Address 1 (Mid) MMAL1	6.2.18.14
0x000E	LAN Ethernet MAC Address 1 (MSB) MMAH1	6.2.18.15
0x000F	LAN Ethernet MAC Address 2 (LSB) MMAL2	6.2.18.16
0x0010	LAN Ethernet MAC Address 2 (Mid) MMAL2	6.2.18.17
0x0011	LAN Ethernet MAC Address 2 (MSB) MMAH2	6.2.18.18
0x0012	LAN Ethernet MAC Address 3 (LSB) MMAL3	6.2.18.19
0x0013	LAN Ethernet MAC Address 3 (Mid) MMAL3	6.2.18.20
0x0014	LAN Ethernet MAC Address 3 (MSB) MMAH3	6.2.18.21
0x0015	LAN TCP/UDP Flexible Filter Port0	6.2.18.22
0x0016	LAN TCP/UDP Flexible Filter Port1	6.2.18.23
0x0017	LAN TCP/UDP Flexible Filter Port2	6.2.18.24
0x0018	LAN TCP/UDP Flexible Filter Port3	6.2.18.25
0x0019	LAN TCP/UDP Flexible Filter Port4	6.2.18.26
0x001A	LAN TCP/UDP Flexible Filter Port5	6.2.18.27
0x001B	LAN TCP/UDP Flexible Filter Port6	6.2.18.28
0x001C	LAN TCP/UDP Flexible Filter Port7	6.2.18.29
0x001D	LAN TCP/UDP Flexible Filter Port8	6.2.18.30
0x001E	LAN TCP/UDP Flexible Filter Port9	6.2.18.31
0x001F	LAN TCP/UDP Flexible Filter Port10	6.2.18.32
0x0020	LAN TCP/UDP Flexible Filter Port11	6.2.18.33
0x0021	LAN TCP/UDP Flexible Filter Port12	6.2.18.34
0x0022	LAN TCP/UDP Flexible Filter Port13	6.2.18.35

Table 6-24. Pass-Through Control Words Section Summary Table [continued]

Word Offset	Description	Section Number
0x0023	LAN TCP/UDP Flexible Filter Port14	6.2.18.36
0x0024	LAN TCP/UDP Flexible Filter Port15	6.2.18.37
0x0025	LAN VLAN Filter 0	6.2.18.38
0x0026	LAN VLAN Filter 1	6.2.18.39
0x0027	LAN VLAN Filter 2	6.2.18.40
0x0028	LAN VLAN Filter 3	6.2.18.41
0x0029	LAN VLAN Filter 4	6.2.18.42
0x002A	LAN VLAN Filter 5	6.2.18.43
0x002B	LAN VLAN Filter 6	6.2.18.44
0x002C	LAN VLAN Filter 7	6.2.18.45
0x002D	MANC Value LSB - LMANC LSB	6.2.18.46
0x002E	MANC Value LSB - LMANC MSB	6.2.18.47
0x002F	Receive Enable 1 - LRXEN1	6.2.18.48
0x0030	Receive Enable 2 - LRXEN2	6.2.18.49
0x0031	MNGONLY Value - MNGONLY LSB	6.2.18.50
0x0032	MNGONLY Value - MNGONLY MSB	6.2.18.51
0x0033	Manageability Decision Filters - MDEF0 LSB	6.2.18.52
0x0034	Manageability Decision Filters - MDEF0 MSB	6.2.18.53
0x0035	Manageability Decision Filters - MDEF_EXT0 LSB	6.2.18.54
0x0036	Manageability Decision Filters - MDEF_EXT0 MSB	6.2.18.55
0x0037	Manageability Decision Filters - MDEF1 LSB	6.2.18.56
0x0038	Manageability Decision Filters - MDEF1 MSB	6.2.18.57
0x0039	Manageability Decision Filters - MDEF_EXT1 LSB	6.2.18.58
0x003A	Manageability Decision Filters - MDEF_EXT1 MSB	6.2.18.59
0x003B	Manageability Decision Filters - MDEF2 LSB	6.2.18.60
0x003C	Manageability Decision Filters - MDEF2 MSB	6.2.18.61
0x003D	Manageability Decision Filters - MDEF_EXT2 LSB	6.2.18.62
0x003E	Manageability Decision Filters - MDEF_EXT2 MSB	6.2.18.63
0x003F	Manageability Decision Filters - MDEF3 LSB	6.2.18.64
0x0040	Manageability Decision Filters - MDEF3 MSB	6.2.18.65
0x0041	Manageability Decision Filters - MDEF_EXT3 LSB	6.2.18.66
0x0042	Manageability Decision Filters - MDEF_EXT3 MSB	6.2.18.67
0x0043	Manageability Decision Filters - MDEF4 LSB	6.2.18.68
0x0044	Manageability Decision Filters - MDEF4 MSB	6.2.18.69
0x0045	Manageability Decision Filters - MDEF_EXT4 LSB	6.2.18.70
0x0046	Manageability Decision Filters - MDEF_EXT4 MSB	6.2.18.71
0x0047	Manageability Decision Filters - MDEF5 LSB	6.2.18.72

Table 6-24. Pass-Through Control Words Section Summary Table [continued]

Word Offset	Description	Section Number
0x0048	Manageability Decision Filters - MDEF5 MSB	6.2.18.73
0x0049	Manageability Decision Filters - MDEF_EXT5 LSB	6.2.18.74
0x004A	Manageability Decision Filters - MDEF_EXT5 MSB	6.2.18.75
0x004B	Manageability Decision Filters - MDEF6 LSB	6.2.18.76
0x004C	Manageability Decision Filters - MDEF6 MSB	6.2.18.77
0x004D	Manageability Decision Filters - MDEF_EXT6 LSB	6.2.18.78
0x004E	Manageability Decision Filters - MDEF_EXT6 MSB	6.2.18.79
0x004F	Manageability EtherType filter 0.1 - METF0.1	6.2.18.80
0x0050	Manageability EtherType filter 0.2 - METF0.2	6.2.18.81
0x0051	Manageability EtherType filter 1.1 - METF1.1	6.2.18.82
0x0052	Manageability EtherType filter 1.2 - METF1.2	6.2.18.83
0x0053	Manageability EtherType filter 2.1 - METF2.1	6.2.18.84
0x0054	Manageability EtherType filter 2.2 - METF2.2	6.2.18.85
0x0055	Manageability EtherType filter 3.1 - METF3.1	6.2.18.86
0x0056	Manageability EtherType filter 3.2 - METF3.2	6.2.18.87
0x0057	ARP Response IPv4 Address 0 - LSB	6.2.18.88
0x0058	ARP Response IPv4 Address 0 - MSB	6.2.18.89
0x0059	IPv6 Address 0 - 0	6.2.18.90
0x005A	IPv6 Address 0 - 1	6.2.18.91
0x005B	IPv6 Address 0 - 2	6.2.18.92
0x005C	IPv6 Address 0 - 3	6.2.18.93
0x005D	IPv6 Address 0 - 4	6.2.18.94
0x005E	IPv6 Address 0 - 5	6.2.18.95
0x005F	IPv6 Address 0 - 6	6.2.18.96
0x0060	IPv6 Address 0 - 7	6.2.18.97
0x0061	IPv6 Address 1 - 0	6.2.18.98
0x0062	IPv6 Address 1 - 1	6.2.18.99
0x0063	IPv6 Address 1 - 2	6.2.18.100
0x0064	IPv6 Address 1 - 3	6.2.18.101
0x0065	IPv6 Address 1 - 4	6.2.18.102
0x0066	IPv6 Address 1 - 5	6.2.18.103
0x0067	IPv6 Address 1 - 6	6.2.18.104
0x0068	IPv6 Address 1 - 7	6.2.18.105
0x0069	IPv6 Address 2 - 0	6.2.18.106
0x006A	IPv6 Address 2 - 1	6.2.18.107
0x006B	IPv6 Address 2 - 2	6.2.18.108
0x006C	IPv6 Address 2 - 3	6.2.18.109

Table 6-24. Pass-Through Control Words Section Summary Table [continued]

Word Offset	Description	Section Number
0x006D	IPv6 Address 2 - 4	6.2.18.110
0x006E	IPv6 Address 2 - 5	6.2.18.111
0x006F	IPv6 Address 2 - 6	6.2.18.112
0x0070	IPv6 Address 2 - 7	6.2.18.113
0x0071	Block CRC8	6.2.18.114

6.2.18.1 Block Length (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Block Length		**Block Length** Length in words of the section covered by CRC. Block Length in words. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.2 LAN IPv4 Address 0 (LSB) MIPAF0 (0x0001)

Bit(s)	Field Name	Default	Description
15:8	IPv4 Address 0, Byte 1	0x0	**IPv4 Address 0, Byte 1** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	IPv4 Address 0, Byte 0	0x0A	**IPv4 Address 0, Byte 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.3 LAN IPv4 Address 0 (MSB) MIPAF0 (0x0002)

Bit(s)	Field Name	Default	Description
15:8	IPv4 Address 0, Byte 3	0x01	**IPv4 Address 0, Byte 3** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	IPv4 Address 0, Byte 2	0x0	**IPv4 Address 0, Byte 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.4 LAN IPv4 Address 1 (LSB) MIPAF1 (0x0003)

Bit(s)	Field Name	Default	Description
15:8	IPv4 Address 1, Byte 1	0x0	**IPv4 Address 1, Byte 1** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	IPv4 Address 1, Byte 0	0x0A	**IPv4 Address 1, Byte 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.5 LAN IPv4 Address 1 (MSB) MIPAF1 (0x0004)

Bit(s)	Field Name	Default	Description
15:8	IPv4 Address 1, Byte 3	0x02	**IPv4 Address 1, Byte 3** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	IPv4 Address 1, Byte 2	0x0	**IPv4 Address 1, Byte 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.6 LAN IPv4 Address 2 (LSB) MIPAF2 (0x0005)

Bit(s)	Field Name	Default	Description
15:8	IPv4 Address 2, Byte 1	0x0	**IPv4 Address 2, Byte 1** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	IPv4 Address 2, Byte 0	0x0A	**IPv4 Address 2, Byte 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.7 LAN IPv4 Address 2 (MSB) MIPAF2 (0x0006)

Bit(s)	Field Name	Default	Description
15:8	IPv4 Address 2, Byte 3	0x03	**IPv4 Address 2, Byte 3** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	IPv4 Address 2, Byte 2	0x0	**IPv4 Address 2, Byte 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.8 LAN IPv4 Address 3 (LSB) MIPAF3 (0x0007)

Bit(s)	Field Name	Default	Description
15:8	IPv4 Address 3, Byte 1	0x0	**IPv4 Address 3, Byte 1** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	IPv4 Address 3, Byte 0	0x0A	**IPv4 Address 3, Byte 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.9 LAN IPv4 Address 3 (MSB) MIPAF3 (0x0008)

Bit(s)	Field Name	Default	Description
15:8	IPv4 Address 3, Byte 3	0x04	**IPv4 Address 3, Byte 3** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	IPv4 Address 3, Byte 2	0x0	**IPv4 Address 3, Byte 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.10 LAN Ethernet MAC Address 0 (LSB) MMAL0 (0x0009)

This word is loaded by the firmware to the 16 LS bits of the MMAL[0] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 0, Byte 1	0x88	**Ethernet MAC Address 0, Byte 1** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 0, Byte 0	0x88	**Ethernet MAC Address 0, Byte 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.11 LAN Ethernet MAC Address 0 (Mid) MMAL0 (0x000A)

This word is loaded by the firmware to the 16 MS bits of the MMAL[0] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 0, Byte 3	0x88	**Ethernet MAC Address 0, Byte 3** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 0, Byte 2	0x88	**Ethernet MAC Address 0, Byte 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.12 LAN Ethernet MAC Address 0 (MSB) MMAH0 (0x000B)

This word is loaded by the firmware to the MMAH[0] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 0, Byte 5	0x01	**Ethernet MAC Address 0, Byte 5** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 0, Byte 4	0x88	**Ethernet MAC Address 0, Byte 4** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.13 LAN Ethernet MAC Address 1 (LSB) MMAL1 (0x000C)

This word is loaded by the firmware to the 16 LS bits of the MMAL[1] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 1, Byte 1	0x88	**Ethernet MAC Address 1, Byte 1** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 1, Byte 0	0x88	**Ethernet MAC Address 1, Byte 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.14 LAN Ethernet MAC Address 1 (Mid) MMAL1 (0x000D)

This word is loaded by the firmware to the 16 MS bits of the MMAL[1] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 1, Byte 3	0x88	**Ethernet MAC Address 1, Byte 3** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 1, Byte 2	0x88	**Ethernet MAC Address 1, Byte 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.15 LAN Ethernet MAC Address 1 (MSB) MMAH1 (0x000E)

This word is loaded by the firmware to the MMAH[1] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 1, Byte 5	0x02	**Ethernet MAC Address 1, Byte 5** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 1, Byte 4	0x88	**Ethernet MAC Address 1, Byte 4** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.16 LAN Ethernet MAC Address 2 (LSB) MMAL2 (0x000F)

This word is loaded by the firmware to the 16 LS bits of the MMAL[2] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 2, Byte 1	0x88	**Ethernet MAC Address 2, Byte 1** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 2, Byte 0	0x88	**Ethernet MAC Address 2, Byte 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.17 LAN Ethernet MAC Address 2 (Mid) MMAL2 (0x0010)

This word is loaded by the firmware to the 16 MS bits of the MMAL[2] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 2, Byte 3	0x88	**Ethernet MAC Address 2, Byte 3** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 2, Byte 2	0x88	**Ethernet MAC Address 2, Byte 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.18 LAN Ethernet MAC Address 2 (MSB) MMAH2 (0x0011)

This word is loaded by the firmware to the MMAH[2] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 2, Byte 5	0x03	**Ethernet MAC Address 2, Byte 5** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 2, Byte 4	0x88	**Ethernet MAC Address 2, Byte 4** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.19 LAN Ethernet MAC Address 3 (LSB) MMAL3 (0x0012)

This word is loaded by the firmware to the 16 LS bits of the MMAL[3] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 3, Byte 1	0x88	**Ethernet MAC Address 3, Byte 1** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 3, Byte 0	0x88	**Ethernet MAC Address 3, Byte 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.20 LAN Ethernet MAC Address 3 (Mid) MMAL3 (0x0013)

This word is loaded by the firmware to the 16 MS bits of the MMAL[3] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 3, Byte 3	0x88	**Ethernet MAC Address 3, Byte 3** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 3, Byte 2	0x88	**Ethernet MAC Address 3, Byte 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.21 LAN Ethernet MAC Address 3 (MSB) MMAH3 (0x0014)

This word is loaded by the firmware to the MMAH[3] register.

Bit(s)	Field Name	Default	Description
15:8	Ethernet MAC Address 3, Byte 5	0x04	**Ethernet MAC Address 3, Byte 5** ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Ethernet MAC Address 3, Byte 4	0x88	**Ethernet MAC Address 3, Byte 4** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.22 LAN TCP/UDP Flexible Filter Port0 (0x0015)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port0	0x0	**LAN UDP Flexible Filter Port 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.23 LAN TCP/UDP Flexible Filter Port1 (0x0016)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port1	0x0	**LAN UDP Flexible Filter Port 1** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.24 LAN TCP/UDP Flexible Filter Port2 (0x0017)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port2	0x0	**LAN UDP Flexible Filter Port 2** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.25 LAN TCP/UDP Flexible Filter Port3 (0x0018)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port3	0x0	**LAN UDP Flexible Filter Port 3** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.26 LAN TCP/UDP Flexible Filter Port4 (0x0019)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port4	0x0	**LAN UDP Flexible Filter Port 4** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.27 LAN TCP/UDP Flexible Filter Port5 (0x001A)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port5	0x0	**LAN UDP Flexible Filter Port 5** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.28 LAN TCP/UDP Flexible Filter Port6 (0x001B)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port6	0x0	**LAN UDP Flexible Filter Port 6** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.29 LAN TCP/UDP Flexible Filter Port7 (0x001C)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port7	0x0	**LAN UDP Flexible Filter Port 7** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.30 LAN TCP/UDP Flexible Filter Port8 (0x001D)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port8	0x0	**LAN UDP Flexible Filter Port 8** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.31 LAN TCP/UDP Flexible Filter Port9 (0x001E)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port9	0x0	**LAN UDP Flexible Filter Port 9** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.32 LAN TCP/UDP Flexible Filter Port10 (0x001F)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port10	0x0	**LAN UDP Flexible Filter Port 10** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.33 LAN TCP/UDP Flexible Filter Port11 (0x0020)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port11	0x0	**LAN UDP Flexible Filter Port 11** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.34 LAN TCP/UDP Flexible Filter Port12 (0x0021)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port12	0x0	**LAN UDP Flexible Filter Port 12** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.35 LAN TCP/UDP Flexible Filter Port13 (0x0022)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port13	0x0	**LAN UDP Flexible Filter Port 13** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.36 LAN TCP/UDP Flexible Filter Port14 (0x0023)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port14	0x0	**LAN UDP Flexible Filter Port 14** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.37 LAN TCP/UDP Flexible Filter Port15 (0x0024)

Bit(s)	Field Name	Default	Description
15:0	LAN UDP Flexible Filter Port15	0x0	**LAN UDP Flexible Filter Port 15** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.38 LAN VLAN Filter 0 (0x0025)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved.
11:0	VLAN Filter 0 Value	0x0	**VLAN Filter 0 Value** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.39 LAN VLAN Filter 1 (0x0026)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved.
11:0	VLAN Filter 1 Value	0x0	**VLAN Filter 1 Value** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.40 LAN VLAN Filter 2 (0x0027)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved.
11:0	VLAN Filter 2 Value	0x0	**VLAN Filter 3 Value** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.41 LAN VLAN Filter 3 (0x0028)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved.
11:0	VLAN Filter 3 Value	0x0	**VLAN Filter 3 Value** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.42 LAN VLAN Filter 4 (0x0029)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved.
11:0	VLAN Filter 4 Value	0x0	**VLAN Filter 4 Value** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.43 LAN VLAN Filter 5 (0x002A)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved.
11:0	VLAN Filter 5 Value	0x0	**VLAN Filter 5 Value** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.44 LAN VLAN Filter 6 (0x002B)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved.
11:0	VLAN Filter 6 Value	0x0	**VLAN Filter 6 Value** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.45 LAN VLAN Filter 7 (0x002C)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved.
11:0	VLAN Filter 7 Value	0x0	**VLAN Filter 7 Value** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.46 MANC Value LSB - LMANC LSB (0x002D)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.18.47 MANC Value MSB - LMANC MSB (0x002E)

Bit(s)	Field Name	Default	Description
15:11	Reserved	0x0	Reserved.
10	Net Type	0b	**Net Type** This bit is loaded to the *NET_TYPE* bit in the MANC register. ***Note:*** This field is preserved by Intel NVM update tool.
9	Fixed Net Type	0b	**Fixed Net Type** This bit is loaded to the *FIXED_NET_TYPE* bit in the MANC register. ***Note:*** This field is preserved by Intel NVM update tool.
8	Enable IPv4 Address Filters	0b	**Enable IPv4 Address Filters** 0b = These bits store a single IPv6 filter. 1b = The last 128 bits of the MIPAF register are used to store four IPv4 addresses for IPv4 filtering. ***Note:*** This field is preserved by Intel NVM update tool.
7	Enable Xsum Filtering to MNG	0b	**Enable Xsum Filtering to MNG** When this bit is set, only packets that pass the L3 and L4 checksum are sent to the manageability block. This feature does not support tunneled IPv4/IPv6 packet inspection. ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
6:0	Reserved	0x0	Reserved.

6.2.18.48 Receive Enable 1 - LRXEN1 (0x002F)

Bit(s)	Field Name	Default	Description
15:8	Receive Enable byte 12	0x0	**Receive Enable byte 12** BMC SMBus slave address. ***Note:*** This field is preserved by Intel NVM update tool.
7	Enable BMC Dedicated	0b	**Enable BMC Dedicated** ***Note:*** This field is preserved by Intel NVM update tool.
6	Reserved	1b	Reserved.
5:4	Notification Method	00b	**Notification Method** 00b = SMBus alert 01b = Asynchronous notify 10b = Direct receive 11b = Reserved ***Note:*** This field is preserved by Intel NVM update tool.
3	Enable ARP Response	0b	**Enable ARP Response** ***Note:*** This field is preserved by Intel NVM update tool.
2	Enable Status Reporting	0b	**Enable Status Reporting** ***Note:*** This field is preserved by Intel NVM update tool.
1	Enable Receive All	0b	**Enable Receive All** ***Note:*** This field is preserved by Intel NVM update tool.
0	Enable Receive TCO	0b	**Enable Receive TCO** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.49 Receive Enable 2 - LRXEN2 (0x0030)

Bit(s)	Field Name	Default	Description
15:8	Receive Enable byte 14	0x0	**Receive Enable byte 14** Alert value. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Receive Enable byte 13	0x0	**Receive Enable byte 13** Interface data. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.50 MNGONLY Value - MNGONLY LSB (0x0031)

Bit(s)	Field Name	Default	Description
15:8	Reserved	0x0	Reserved.
7:0	Exclusive to MNG	0x0	**Exclusive to Manageability** When set, indicates that packets forwarded by the manageability filters to manageability are not sent to the host. Bit 0 corresponds to decision rule 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.51 MNGONLY Value - MNGONLY MSB (0x0032)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.18.52 Manageability Decision Filters - MDEF0 LSB (0x0033)

Bit(s)	Field Name	Default	Description
15:12	Flex Port (OR)	0x0	**Flex Port (OR)** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11	Port 0x26F (OR)	0b	**Port 0x26F (OR)** Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section) ***Note:*** This field is preserved by Intel NVM update tool.
10	Port 0x298 (OR)	0b	**Port 0x298 (OR)** Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
9	Neighbor Discovery (OR)	0b	**Neighbor Discovery (OR)** Controls the inclusion of neighbor discovery filtering in the manageability filter decision (OR section). The neighbor types accepted by this filter are types 0x86, 0x87, 0x88 and 0x89. ***Note:*** This field is preserved by Intel NVM update tool.
8	ARP Response (OR)	0b	**ARP Response (OR)** Controls the inclusion of ARP response filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7	ARP Request (OR)	0b	**ARP Request (OR)** Controls the inclusion of ARP request filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
6	VLAN (OR)	0b	**VLAN (OR)** Controls the inclusion of VLAN addresses filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
5	Broadcast (OR)	0b	**Broadcast (OR)** Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
4	Unicast (OR)	0b	**Unicast (OR)** Controls the inclusion of unicast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
3	IP Address (AND)	0b	**IP Address (AND)** Controls the inclusion of IP Address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
2	VLAN (AND)	0b	**VLAN (AND)** Controls the inclusion of VLAN address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
1	Broadcast (AND)	0b	**Broadcast (AND)** Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
0	Unicast (AND)	0b	**Unicast (AND)** Controls the inclusion of unicast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.53 Manageability Decision Filters - MDEF0 MSB (0x0034)

Bit(s)	Field Name	Default	Description
15:12	Flex TCO	0x0	**Flex TCO** Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 28 corresponds to Flex TCO filter 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11:0	Flex Port	0x0	**Flex Port** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.54 Manageability Decision Filters - MDEF_EXT0 LSB (0x0035)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved for additional L2 EtherType OR filters.
11:8	L2 EtherType OR L2 EtherType	0x0	**L2 EtherType OR L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7:4	Reserved	0x0	Reserved for additional L2 EtherType AND filters.
3:0	L2 EtherType AND L2 EtherType	0x0	**L2 EtherType AND L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.55 Manageability Decision Filters - MDEF_EXT0 MSB (0x0036)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.18.56 Manageability Decision Filters - MDEF1 LSB (0x0037)

Bit(s)	Field Name	Default	Description
15:12	Flex Port (OR)	0x0	**Flex Port (OR)** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11	Port 0x26F (OR)	0b	**Port 0x26F (OR)** Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section) ***Note:*** This field is preserved by Intel NVM update tool.
10	Port 0x298 (OR)	0b	**Port 0x298 (OR)** Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
9	Neighbor Discovery (OR)	0b	**Neighbor Discovery (OR)** Controls the inclusion of neighbor discovery filtering in the manageability filter decision (OR section). The neighbor types accepted by this filter are types 0x86, 0x87, 0x88 and 0x89. ***Note:*** This field is preserved by Intel NVM update tool.
8	ARP Response (OR)	0b	**ARP Response (OR)** Controls the inclusion of ARP response filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7	ARP Request (OR)	0b	**ARP Request (OR)** Controls the inclusion of ARP request filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
6	VLAN (OR)	0b	**VLAN (OR)** Controls the inclusion of VLAN addresses filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
5	Broadcast (OR)	0b	**Broadcast (OR)** Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
4	Unicast (OR)	0b	**Unicast (OR)** Controls the inclusion of unicast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
3	IP Address (AND)	0b	**IP Address (AND)** Controls the inclusion of IP Address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
2	VLAN (AND)	0b	**VLAN (AND)** Controls the inclusion of VLAN address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
1	Broadcast (AND)	0b	**Broadcast (AND)** Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
0	Unicast (AND)	0b	**Unicast (AND)** Controls the inclusion of unicast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.57 Manageability Decision Filters - MDEF1 MSB (0x0038)

Bit(s)	Field Name	Default	Description
15:12	Flex TCO	0x0	**Flex TCO** Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 28 corresponds to Flex TCO filter 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11:0	Flex Port	0x0	**Flex Port** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.58 Manageability Decision Filters - MDEF_EXT1 LSB (0x0039)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved for additional L2 EtherType OR filters.
11:8	L2 EtherType OR L2 EtherType	0x0	**L2 EtherType OR L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7:4	Reserved	0x0	Reserved for additional L2 EtherType AND filters.
3:0	L2 EtherType AND L2 EtherType	0x0	**L2 EtherType AND L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.59 Manageability Decision Filters - MDEF_EXT1 MSB (0x003A)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.18.60 Manageability Decision Filters - MDEF2 LSB (0x003B)

Bit(s)	Field Name	Default	Description
15:12	Flex Port (OR)	0x0	**Flex Port (OR)** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11	Port 0x26F (OR)	0b	**Port 0x26F (OR)** Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section) ***Note:*** This field is preserved by Intel NVM update tool.
10	Port 0x298 (OR)	0b	**Port 0x298 (OR)** Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
9	Neighbor Discovery (OR)	0b	**Neighbor Discovery (OR)** Controls the inclusion of neighbor discovery filtering in the manageability filter decision (OR section). The neighbor types accepted by this filter are types 0x86, 0x87, 0x88 and 0x89. ***Note:*** This field is preserved by Intel NVM update tool.
8	ARP Response (OR)	0b	**ARP Response (OR)** Controls the inclusion of ARP response filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7	ARP Request (OR)	0b	**ARP Request (OR)** Controls the inclusion of ARP request filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
6	VLAN (OR)	0b	**VLAN (OR)** Controls the inclusion of VLAN addresses filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
5	Broadcast (OR)	0b	**Broadcast (OR)** Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
4	Unicast (OR)	0b	**Unicast (OR)** Controls the inclusion of unicast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
3	IP Address (AND)	0b	**IP Address (AND)** Controls the inclusion of IP Address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
2	VLAN (AND)	0b	**VLAN (AND)** Controls the inclusion of VLAN address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
1	Broadcast (AND)	0b	**Broadcast (AND)** Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
0	Unicast (AND)	0b	**Unicast (AND)** Controls the inclusion of unicast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.61 Manageability Decision Filters - MDEF2 MSB (0x003C)

Bit(s)	Field Name	Default	Description
15:12	Flex TCO	0x0	**Flex TCO** Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 28 corresponds to Flex TCO filter 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11:0	Flex Port	0x0	**Flex Port** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.62 Manageability Decision Filters - MDEF_EXT2 LSB (0x003D)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved for additional L2 EtherType OR filters.
11:8	L2 EtherType OR L2 EtherType	0x0	**L2 EtherType OR L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7:4	Reserved	0x0	Reserved for additional L2 EtherType AND filters.
3:0	L2 EtherType AND L2 EtherType	0x0	**L2 EtherType AND L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.63 Manageability Decision Filters - MDEF2_EXT2 MSB (0x003E)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.18.64 Manageability Decision Filters - MDEF3 LSB (0x003F)

Bit(s)	Field Name	Default	Description
15:12	Flex Port (OR)	0x0	**Flex Port (OR)** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11	Port 0x26F (OR)	0b	**Port 0x26F (OR)** Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section) ***Note:*** This field is preserved by Intel NVM update tool.
10	Port 0x298 (OR)	0b	**Port 0x298 (OR)** Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
9	Neighbor Discovery (OR)	0b	**Neighbor Discovery (OR)** Controls the inclusion of neighbor discovery filtering in the manageability filter decision (OR section). The neighbor types accepted by this filter are types 0x86, 0x87, 0x88 and 0x89. ***Note:*** This field is preserved by Intel NVM update tool.
8	ARP Response (OR)	0b	**ARP Response (OR)** Controls the inclusion of ARP response filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7	ARP Request (OR)	0b	**ARP Request (OR)** Controls the inclusion of ARP request filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
6	VLAN (OR)	0b	**VLAN (OR)** Controls the inclusion of VLAN addresses filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
5	Broadcast (OR)	0b	**Broadcast (OR)** Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
4	Unicast (OR)	0b	**Unicast (OR)** Controls the inclusion of unicast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
3	IP Address (AND)	0b	**IP Address (AND)** Controls the inclusion of IP Address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
2	VLAN (AND)	0b	**VLAN (AND)** Controls the inclusion of VLAN address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
1	Broadcast (AND)	0b	**Broadcast (AND)** Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
0	Unicast (AND)	0b	**Unicast (AND)** Controls the inclusion of unicast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.65 Manageability Decision Filters - MDEF3 MSB (0x0040)

Bit(s)	Field Name	Default	Description
15:12	Flex TCO	0x0	**Flex TCO** Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 28 corresponds to Flex TCO filter 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11:0	Flex Port	0x0	**Flex Port** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.66 Manageability Decision Filters - MDEF_EXT3 LSB (0x0041)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved for additional L2 EtherType OR filters.
11:8	L2 EtherType OR L2 EtherType	0x0	**L2 EtherType OR L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7:4	Reserved	0x0	Reserved for additional L2 EtherType AND filters.

Bit(s)	Field Name	Default	Description
3:0	L2 EtherType AND L2 EtherType	0x0	**L2 EtherType AND L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.67 Manageability Decision Filters - MDEF_EXT3 MSB (0x0042)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.18.68 Manageability Decision Filters - MDEF4 LSB (0x0043)

Bit(s)	Field Name	Default	Description
15:12	Flex Port (OR)	0x0	**Flex Port (OR)** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11	Port 0x26F (OR)	0b	**Port 0x26F (OR)** Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section) ***Note:*** This field is preserved by Intel NVM update tool.
10	Port 0x298 (OR)	0b	**Port 0x298 (OR)** Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
9	Neighbor Discovery (OR)	0b	**Neighbor Discovery (OR)** Controls the inclusion of neighbor discovery filtering in the manageability filter decision (OR section). The neighbor types accepted by this filter are types 0x86, 0x87, 0x88 and 0x89. ***Note:*** This field is preserved by Intel NVM update tool.
8	ARP Response (OR)	0b	**ARP Response (OR)** Controls the inclusion of ARP response filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7	ARP Request (OR)	0b	**ARP Request (OR)** Controls the inclusion of ARP request filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
6	VLAN (OR)	0b	**VLAN (OR)** Controls the inclusion of VLAN addresses filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
5	Broadcast (OR)	0b	**Broadcast (OR)** Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
4	Unicast (OR)	0b	**Unicast (OR)** Controls the inclusion of unicast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
3	IP Address (AND)	0b	**IP Address (AND)** Controls the inclusion of IP Address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
2	VLAN (AND)	0b	**VLAN (AND)** Controls the inclusion of VLAN address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
1	Broadcast (AND)	0b	**Broadcast (AND)** Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
0	Unicast (AND)	0b	**Unicast (AND)** Controls the inclusion of unicast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.69 Manageability Decision Filters - MDEF4 MSB (0x0044)

Bit(s)	Field Name	Default	Description
15:12	Flex TCO	0x0	**Flex TCO** Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 28 corresponds to Flex TCO filter 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11:0	Flex Port	0x0	**Flex Port** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.70 Manageability Decision Filters - MDEF_EXT4 LSB (0x0045)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved for additional L2 EtherType OR filters.
11:8	L2 EtherType OR L2 EtherType	0x0	**L2 EtherType OR L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7:4	Reserved	0x0	Reserved for additional L2 EtherType AND filters.
3:0	L2 EtherType AND L2 EtherType	0x0	**L2 EtherType AND L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.71 Manageability Decision Filters - MDEF_EXT4 MSB (0x0046)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.18.72 Manageability Decision Filters - MDEF5 LSB (0x0047)

Bit(s)	Field Name	Default	Description
15:12	Flex Port (OR)	0x0	**Flex Port (OR)** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11	Port 0x26F (OR)	0b	**Port 0x26F (OR)** Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section) ***Note:*** This field is preserved by Intel NVM update tool.
10	Port 0x298 (OR)	0b	**Port 0x298 (OR)** Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
9	Neighbor Discovery (OR)	0b	**Neighbor Discovery (OR)** Controls the inclusion of neighbor discovery filtering in the manageability filter decision (OR section). The neighbor types accepted by this filter are types 0x86, 0x87, 0x88 and 0x89. ***Note:*** This field is preserved by Intel NVM update tool.
8	ARP Response (OR)	0b	**ARP Response (OR)** Controls the inclusion of ARP response filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7	ARP Request (OR)	0b	**ARP Request (OR)** Controls the inclusion of ARP request filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
6	VLAN (OR)	0b	**VLAN (OR)** Controls the inclusion of VLAN addresses filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
5	Broadcast (OR)	0b	**Broadcast (OR)** Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
4	Unicast (OR)	0b	**Unicast (OR)** Controls the inclusion of unicast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
3	IP Address (AND)	0b	**IP Address (AND)** Controls the inclusion of IP Address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
2	VLAN (AND)	0b	**VLAN (AND)** Controls the inclusion of VLAN address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
1	Broadcast (AND)	0b	**Broadcast (AND)** Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
0	Unicast (AND)	0b	**Unicast (AND)** Controls the inclusion of unicast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.73 Manageability Decision Filters - MDEF5 MSB (0x0048)

Bit(s)	Field Name	Default	Description
15:12	Flex TCO	0x0	**Flex TCO** Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 28 corresponds to Flex TCO filter 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11:0	Flex Port	0x0	**Flex Port** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.74 Manageability Decision Filters - MDEF_EXT5 LSB (0x0049)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved for additional L2 EtherType OR filters.
11:8	L2 EtherType OR L2 EtherType	0x0	**L2 EtherType OR L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7:4	Reserved	0x0	Reserved for additional L2 EtherType AND filters.
3:0	L2 EtherType AND L2 EtherType	0x0	**L2 EtherType AND L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.75 Manageability Decision Filters - MDEF5_EXT5 MSB (0x004A)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.18.76 Manageability Decision Filters - MDEF6 LSB (0x004B)

Bit(s)	Field Name	Default	Description
15:12	Flex Port (OR)	0x0	**Flex Port (OR)** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11	Port 0x26F (OR)	0b	**Port 0x26F (OR)** Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section) ***Note:*** This field is preserved by Intel NVM update tool.
10	Port 0x298 (OR)	0b	**Port 0x298 (OR)** Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.

Bit(s)	Field Name	Default	Description
9	Neighbor Discovery (OR)	0b	**Neighbor Discovery (OR)** Controls the inclusion of neighbor discovery filtering in the manageability filter decision (OR section). The neighbor types accepted by this filter are types 0x86, 0x87, 0x88 and 0x89. ***Note:*** This field is preserved by Intel NVM update tool.
8	ARP Response (OR)	0b	**ARP Response (OR)** Controls the inclusion of ARP response filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7	ARP Request (OR)	0b	**ARP Request (OR)** Controls the inclusion of ARP request filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
6	VLAN (OR)	0b	**VLAN (OR)** Controls the inclusion of VLAN addresses filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
5	Broadcast (OR)	0b	**Broadcast (OR)** Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
4	Unicast (OR)	0b	**Unicast (OR)** Controls the inclusion of unicast address filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
3	IP Address (AND)	0b	**IP Address (AND)** Controls the inclusion of IP Address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
2	VLAN (AND)	0b	**VLAN (AND)** Controls the inclusion of VLAN address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
1	Broadcast (AND)	0b	**Broadcast (AND)** Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.
0	Unicast (AND)	0b	**Unicast (AND)** Controls the inclusion of unicast address filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.77 Manageability Decision Filters - MDEF6 MSB (0x004C)

Bit(s)	Field Name	Default	Description
15:12	Flex TCO	0x0	**Flex TCO** Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 28 corresponds to Flex TCO filter 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.
11:0	Flex Port	0x0	**Flex Port** Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.78 Manageability Decision Filters - MDEF_EXT6 LSB (0x004D)

Bit(s)	Field Name	Default	Description
15:12	Reserved	0x0	Reserved for additional L2 EtherType OR filters.
11:8	L2 EtherType OR L2 EtherType	0x0	**L2 EtherType OR L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section). ***Note:*** This field is preserved by Intel NVM update tool.
7:4	Reserved	0x0	Reserved for additional L2 EtherType AND filters.
3:0	L2 EtherType AND L2 EtherType	0x0	**L2 EtherType AND L2 EtherType** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.79 Manageability Decision Filters - MDEF_EXT6 MSB (0x004E)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0x0	Reserved.

6.2.18.80 Manageability EtherType Filter 0.1 - METF0.1 (0x004F)

Bit(s)	Field Name	Default	Description
15:0	METF0_L	0x0	**METF0 LSB** Loaded to 16 LS bits of METF[0] register. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.81 Manageability EtherType Filter 0.2 - METF0.2 (0x0050)

Bit(s)	Field Name	Default	Description
15:0	METF0_M	0x0	**METF0 MSB** Loaded to 16 MS bits of METF[0] register (reserved bits in the METF registers should be set in the NVM to the register's default values). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.82 Manageability EtherType Filter 1.1 - METF1.1 (0x0051)

Bit(s)	Field Name	Default	Description
15:0	METF1_L	0x0	**METF1 LSB** Loaded to 16 LS bits of METF[1] register. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.83 Manageability EtherType Filter 1.2 - METF1.2 (0x0052)

Bit(s)	Field Name	Default	Description
15:0	METF1_M	0x0	**METF1 MSB** Loaded to 16 MS bits of METF[1] register (reserved bits in the METF registers should be set in the NVM to the register's default values). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.84 Manageability EtherType filter 2.1 - METF2.1 (0x0053)

Bit(s)	Field Name	Default	Description
15:0	METF2_L	0x0	**METF2 LSB** Loaded to 16 LS bits of METF[2] register. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.85 Manageability EtherType Filter 2.2 - METF2.2 (0x0054)

Bit(s)	Field Name	Default	Description
15:0	METF2_M	0x0	**METF2 MSB** Loaded to 16 MS bits of METF[2] register (reserved bits in the METF registers should be set in the NVM to the register's default values). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.86 Manageability EtherType Filter 3.1 - METF3.1 (0x0055)

Bit(s)	Field Name	Default	Description
15:0	METF3_L	0x0	**METF3 LSB** Loaded to 16 LS bits of METF[3] register. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.87 Manageability EtherType Filter 3.2 - METF3.2 (0x0056)

Bit(s)	Field Name	Default	Description
15:0	METF3_M	0x0	**METF3 MSB** Loaded to 16 MS bits of METF[3] register (reserved bits in the METF registers should be set in the NVM to the register's default values). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.88 ARP Response IPv4 Address 0 - LSB (0x0057)

Bit(s)	Field Name	Default	Description
15:8	Byte 1	0x0	**Byte 1** Firmware use. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Byte 0	0x0	**Byte 0** Firmware use. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.89 ARP Response IPv4 Address 0 - MSB (0x0058)

Bit(s)	Field Name	Default	Description
15:8	Byte 3	0x0	**Byte 3** Firmware use. ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Byte 2	0x0	**Byte 4** Firmware use. ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.90 IPv6 Address 0 - 0 (0x0059)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.91 IPv6 Address 0 - 1 (0x005A)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.92 IPv6 Address 0 - 2 (0x005B)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.93 IPv6 Address 0 - 3 (0x005C)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.94 IPv6 Address 0 - 4 (0x005D)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.95 IPv6 Address 0 - 5 (0x005E)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.96 IPv6 Address 0 - 6 (0x005F)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.97 IPv6 Address 0 - 7 (0x0060)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.98 IPv6 Address 1 - 0 (0x0061)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.99 IPv6 Address 1 - 1 (0x0062)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.100 IPv6 Address 1 - 2 (0x0063)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.101 IPv6 Address 1 - 3 (0x0064)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.102 IPv6 Address 1 - 4 (0x0065)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.103 IPv6 Address 1 - 5 (0x0066)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.104 IPv6 Address 1 - 6 (0x0067)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.105 IPv6 Address 1 - 7 (0x0068)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.106 IPv6 Address 2 - 0 (0x0069)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.107 IPv6 Address 2 - 1 (0x006A)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.108 IPv6 Address 2 - 2 (0x006B)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.109 IPv6 Address 2 - 3 (0x006C)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.110 IPv6 Address 2 - 4 (0x006D)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.111 IPv6 Address 2 - 5 (0x006E)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.112 IPv6 Address 2 - 6 (0x006F)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.113 IPv6 Address 2 - 7 (0x0070)

Bit(s)	Field Name	Default	Description
15:0	IPv6 Address	0x0	**IPv6 Address** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.18.114 Block CRC8 (0x0071)

Bit(s)	Field Name	Default	Description
15:8	Block CRC8		**Block CRC8** CRC-8-CCITT: Start Section -> Word: Pass Through Control Words -> Block Length End Section -> Word: Pass Through Control Words -> IPv6 Address 2 - 7 ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Reserved	0x0	Reserved. Block length in words.

6.2.19 Flexible TCO Filter Configuration Structure Section

Table 6-25. Flexible TCO Filter Configuration Structure Section Summary Table

Word Offset	Description	Section Number
0x0000	Block Length	6.2.19.1
0x0001	Flexible Filter Length and Control	6.2.19.2
0x0002 + 1*n, n=0...54	Flexible Filter Enable Mask	6.2.19.3
0x0039 + 1*n, n=0...63	Flexible Filter Data	6.2.19.4
0x0079	Block CRC8	6.2.19.5

6.2.19.1 Block Length (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Block Length		**Block Length** Length in words of the section covered by CRC Section length in words ***Note:*** This field is preserved by Intel NVM update tool.

6.2.19.2 Flexible Filter Length and Control (0x0001)

Bit(s)	Field Name	Default	Description
15:8	Flexible Filter Length (bytes)	0x0	**Flexible Filter Length (bytes)** ***Note:*** This field is preserved by Intel NVM update tool.
7:5	Reserved	000b	Reserved.
4	Last Filter	1b	**Last Filter** ***Note:*** This field is preserved by Intel NVM update tool.
3:2	Filter Index (0-3)	01b	**Filter Index (0-3)** ***Note:*** This field is preserved by Intel NVM update tool.
1	Apply Filter to LAN 1	0b	**Apply Filter to LAN 1** ***Note:*** This field is preserved by Intel NVM update tool.
0	Apply Filter to LAN 0	0b	**Apply Filter to LAN 0** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.19.3 Flexible Filter Enable Mask[n] (0x0002 + 1*n, n=0...54)

Bit(s)	Field Name	Default	Description
15:0	Flexible Filter Enable Mask	0x0	**Flexible Filter Enable Mask** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.19.4 Flexible Filter Data[n] (0x0039 + 1*n, n=0...63)

Bit(s)	Field Name	Default	Description
15:0	Flexible Filter Data	0x0	**Flexible Filter Data** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.19.5 Block CRC8 (0x0079)

Bit(s)	Field Name	Default	Description
15:8	Block CRC8		**Block CRC8** CRC-8-CCITT: Start Section -> Word: Flexible TCO Filter Configuration Structure -> Block Length End Section -> Word: Flexible TCO Filter Configuration Structure -> Flexible Filter Data ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Reserved	0x0	Reserved. Block length in words.

6.2.20 LESM Configurations Section

Table 6-26. LESM Configurations Section Summary Table

Word Offset	Description	Section Number
0x0000	Section Header	6.2.20.1
0x0001	LESM Global Configurations	6.2.20.2
0x0002	LESM Per State Configurations	6.2.20.3
0x0003	LESM Per State Configurations LSB	6.2.20.4
0x0004	LESM Per State Configurations MSB	6.2.20.5
0x0005	Section Footer	6.2.20.6

6.2.20.1 Section Header (0x0000)

Bit(s)	Field Name	Default	Description
15:8	Reserved	0x0	Reserved.
7:0	Block Length	0x5	**Block Length** Length in words of the section covered by CRC Block length in words (including the CRC word and not the length word).

6.2.20.2 LESM Global Configurations (0x0001)

Bit(s)	Field Name	Default	Description
15:6	Reserved	0x0	Reserved.
5:1	10G Only Counter	0x3	**10G Only Counter** Number of iterations to go through before enabling 1G parallel detection.
0	LESM Global Configurations	1b	**LESM Global Configurations** 0b = LESM is disabled. 1b = LESM is enabled.

6.2.20.3 LESM Per State Configurations (0x0002)

Bit(s)	Field Name	Default	Description
15:1	[New Field]	0x3	**[New Field]** Number of iterations to go through before enabling 1G parallel detection.
0	State Enable	1b	**State Enable** 0b = LESM is disabled. 1b = LESM is enabled.

6.2.20.4 LESM Per State Configurations LSB (0x0003)

Bit(s)	Field Name	Default	Description
15:0	Register Pointer LSB	0x0	**Register Pointer LSB**

6.2.20.5 LESM Per State Configurations MSB (0x0004)

Bit(s)	Field Name	Default	Description
15:0	Register Pointer MSB	0x0	**Register Pointer MSB**

6.2.20.6 Section Footer (0x0005)

Bit(s)	Field Name	Default	Description
15:8	Section Footer		**Block CRC8** CRC-8-CCITT: Start Section -> Word: LESM Configurations (not in SGVL) -> Section Header End Section -> Word: LESM Configurations (not in SGVL) -> LESM Per State Configurations MSB CRC8 of the section above.
7:0	Reserved	0x0	Reserved.

6.2.21 PXE VLAN Configuration Section

Table 6-27. PXE VLAN Configuration Section Summary Table

Word Offset	Description	Section Number
0x0000	VLAN Block Signature	6.2.21.1
0x0001	Version and Size	6.2.21.2
0x0002	Port 0 VLAN Tag	6.2.21.3
0x0003	Port 1 VLAN Tag	6.2.21.4

6.2.21.1 VLAN Block Signature (0x0000)

ASCII 'V', 'L'.

Bit(s)	Field Name	Default	Description
15:0	VLAN Block Signature	0x4C56	**VLAN Block Signature** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.21.2 Version and Size (0x0001)

Bit(s)	Field Name	Default	Description
15:8	Size		**Size** Length in: 1 Byte unit First Section -> Word: PXE VLAN Configuration -> VLAN Block Signature Last Section -> Word: PXE VLAN Configuration -> Port 1 VLAN Tag total size in bytes of section ***Note:*** This field is preserved by Intel NVM update tool.
7:0	Version	0x01	**Version** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.21.3 Port 0 VLAN Tag (0x0002)

Bit(s)	Field Name	Default	Description
15:13	Priority (0-7)	0x0	**Priority (0-7)** ***Note:*** This field is preserved by Intel NVM update tool.
12	Reserved	0b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
11:0	VLAN ID (1- 4095)	0x0	**VLAN ID (1- 4095)** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.21.4 Port 1 VLAN Tag (0x0003)

Bit(s)	Field Name	Default	Description
15:13	Priority (0-7)	0x0	**Priority (0-7)** ***Note:*** This field is preserved by Intel NVM update tool.
12	Reserved	0b	Reserved. ***Note:*** This field is preserved by Intel NVM update tool.
11:0	VLAN ID (1- 4095)	0x0	**VLAN ID (1- 4095)** ***Note:*** This field is preserved by Intel NVM update tool.

6.2.22 VPD Module Section

Note: This section is preserved by Intel NVM update tool.

6.2.23 PBA Number Module Section

Table 6-28. PBA Number Module Section Summary Table

Word Offset	Description	Section Number
0x0000	PBA Section Length	6.2.23.1
0x0001	Word1	6.2.23.2
0x0002	Word2	6.2.23.3
0x0003	Word3	6.2.23.4
0x0004	Word4	6.2.23.5
0x0005	Word5	6.2.23.6

6.2.23.1 PBA Section Length (0x0000)

Bit(s)	Field Name	Default	Description
15:0	PBA Section Length Field	0x6	Length in words of the PBA Block.

6.2.23.2 Word1 (0x0001)

Bit(s)	Field Name	Default	Description
15:0	Word1 Field		**Word1 Field** PBA Number stored in hexadecimal ASCII values.

6.2.23.3 Word2 (0x0002)

Bit(s)	Field Name	Default	Description
15:0	Word2 Field		**Word2 Field** PBA Number stored in hexadecimal ASCII values.

6.2.23.4 Word3 (0x0003)

Bit(s)	Field Name	Default	Description
15:0	Word3 Field		**Word3 Field** PBA Number stored in hexadecimal ASCII values.

6.2.23.5 Word4 (0x0004)

Bit(s)	Field Name	Default	Description
15:0	Word4 Field		**Word4 Field** PBA Number stored in hexadecimal ASCII values.

6.2.23.6 Word5 (0x0005)

Bit(s)	Field Name	Default	Description
15:0	Word5 Field		**Word5 Field** PBA Number stored in hexadecimal ASCII values.

6.2.24 Mini Loader Module Section

Table 6-29. Mini Loader Module Section Summary Table

Word Offset	Description	Section Number
0x0000	Mini Loader Section Header	6.2.24.1
0x0001	Mini Loader Code	6.2.24.2
0x0002	Mini Loader Section Footer	6.2.24.3

6.2.24.1 Mini Loader Section Header (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Block Length		**Block Length** Length in words of the section covered by CRC. For now, ROM 0.3 (include len-crc, later only crc).

6.2.24.2 Mini Loader Code (0x0001)

Raw data module length: variable

6.2.24.3 Mini Loader Section Footer (0x0002)

Bit(s)	Field Name	Default	Description
15:8	CRC8		**CRC8** CRC-8-CCITT: Start Section -> Word: Mini Loader Module -> Mini Loader Section Header End Section -> Word: Mini Loader Module -> Mini Loader Code For now, ML 0.3 code by default: bu in ROM 0.3.
7:0	Reserved	0x0	Reserved.

6.2.25 PHY Config Section

MDIO Indirect Register Programming List.

Address 0xFC2 to 0x1000 has room for 20 registers in case of list expansion in modified images.

Table 6-30. PHY Config Section Summary Table

Word Offset	Description	Section Number
0x0000	Section Length	6.2.25.1
0x0001	PMA RX Prov Port0 LSA	6.2.25.2
0x0002	PMA RX Prov Port0 MSA	6.2.25.3
0x0003	PMA RX Prov Port0 Data	6.2.25.4
0x0004	PMA RX Prov Port0 Field Enables	6.2.25.5
0x0005	PMA RX Prov Port1 LSA	6.2.25.6
0x0006	PMA RX Prov Port1 MSA	6.2.25.7
0x0007	PMA RX Prov Port1 Data	6.2.25.8
0x0008	PMA RX Prov Port1 Field Enables	6.2.25.9
0x0009	Glob Interrupt Vendor Mask LSA	6.2.25.10
0x000A	Glob Interrupt Vendor Mask MSA	6.2.25.11
0x000B	Glob Interrupt Vendor Mask Data	6.2.25.12
0x000C	Glob Interrupt Vendor Mask Field Enables	6.2.25.13
0x000D	Glob Interrupt Standard Mask LSA	6.2.25.14
0x000E	Glob Interrupt Standard Mask MSA	6.2.25.15
0x000F	Glob Interrupt Standard Mask Data	6.2.25.16
0x0010	Glob Interrupt Standard Mask Field Enables	6.2.25.17
0x0011	CSR RAW1	6.2.25.18
0x0012	Section Footer	6.2.25.19

6.2.25.1 Section Length (0x0000)

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word. Section Length = 3*n (n = number of CSRs to configure).

Bit(s)	Field Name	Default	Description
15:0	Section Length		**Section Length** Length in words of the section covered by CRC. Section length in words.

6.2.25.2 PMA RX Prov Port0 LSA (0x0001)

Bit(s)	Field Name	Default	Description
15:0	Address LS	0xE400	**Address LS**

6.2.25.3 PMA RX Prov Port0 MSA (0x0002)

Bit(s)	Field Name	Default	Description
15:10	Reserved	0x0	Reserved.
9:5	Port	0x0	**Port**
4:0	Device	0x1	**Device** 0x0 = None 0x1 = PMA/PMD 0x3 = PCS 0x7 = Auto-negotiation 0x1D = Clause22 Ext 0x1E = Global All other values are reserved.

6.2.25.4 PMA RX Prov Port0 Data (0x0003)

Bit(s)	Field Name	Default	Description
15	P0 Ena PHY Loopback	0b	**P0 Enable PHY Loopback**
14:1	Reserved	0x0	Reserved.
0	P0 MDI Config	0b	**P0 MDI Config** 0b = Normal — MDI normal (ABCD -> ABCD). 1b = Reverse — MDI reversed (ABCD -> DCBA). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.25.5 PMA RX Prov Port0 Field Enables (0x0004)

Bit(s)	Field Name	Default	Description
15	P0 Ena PHY Loopback Field Enable	0b	**P0 Enable PHY Loopback Field Enable**
14:1	Reserved	0x0	Reserved.
0	P0 MDI Config Field Enable	1b	**P0 MDI Config Field Enable** 0b = No Change — No Change to the field. 1b = Enable — Allow field update.

6.2.25.6 PMA RX Prov Port1 LSA (0x0005)

Bit(s)	Field Name	Default	Description
15:0	Address LS	0xE400	**Address LS**

6.2.25.7 PMA RX Prov Port1 MSA (0x0006)

Bit(s)	Field Name	Default	Description
15:10	Reserved	0x0	Reserved.
9:5	Port	0x1	**Port**

Bit(s)	Field Name	Default	Description
4:0	Device	0x1	**Device** 0x0 = None 0x1 = PMA/PMD 0x3 = PCS 0x7 = Auto-negotiation 0x1D = Clause22 Ext 0x1E = Global All other values are reserved.

6.2.25.8 PMA RX Prov Port1 Data (0x0007)

Bit(s)	Field Name	Default	Description
15	P1 Ena PHY Loopback	0b	**P1 Enable PHY Loopback**
14:1	Reserved	0x0	Reserved.
0	P1 MDI Config	1b	**P1 MDI Config** 0b = Normal — MDI normal (ABCD -> ABCD). 1b = Reverse — MDI reversed (ABCD -> DCBA). ***Note:*** This field is preserved by Intel NVM update tool.

6.2.25.9 PMA RX Prov Port1 Field Enables (0x0008)

Bit(s)	Field Name	Default	Description
15	P1 Ena PHY Loopback Field Enable	0b	**P1 Enable PHY Loopback Field Enable**
14:1	Reserved	0x0	Reserved.
0	P1 MDI Config Field Enable	1b	**P1 MDI Config Field Enable** 0b = No Change — No Change to the field. 1b = Enable — Allow field update.

6.2.25.10 Glob Interrupt Vendor Mask LSA (0x0009)

Bit(s)	Field Name	Default	Description
15:0	Address LS	0xFF01	**Address LS**

6.2.25.11 Glob Interrupt Vendor Mask MSA (0x000A)

Bit(s)	Field Name	Default	Description
15:10	Reserved	0x0	Reserved.
9:5	Port	0x0	**Port**

Bit(s)	Field Name	Default	Description
4:0	Device	0x1E	**Device** 0x0 = None 0x1 = PMA/PMD 0x3 = PCS 0x7 = Auto-negotiation 0x1D = Clause22 Ext 0x1E = Global All other values are reserved.

6.2.25.12 Glob Interrupt Vendor Mask Data (0x000B)

Bit(s)	Field Name	Default	Description
15	PMA Vendor Alarm Int Ena	0b	**PMA Vendor Alarm Int Enable**
14	PCS Vendor Alarm Int Ena	0b	**PCS Vendor Alarm Int Enable**
13	PHY XS Vendor Alarm Int Ena	0b	**PHY XS Vendor Alarm Int Enable**
12	AutoNeg Vendor Alarm Int Ena	0b	**Auto-Negotiation Vendor Alarm Int Enable**
11	GBE Vendor Alarm Int Ena	0b	**GBE Vendor Alarm Int Enable**
10:3	Reserved	0x0	Reserved.
2	Global Alarms 1 Int Ena	1b	**Global Alarms 1 Int Enable**
1	Global Alarms 2 Int Ena	0b	**Global Alarms 2 Int Enable**
0	Global Alarms 3 Int Ena	1b	**Global Alarms 3 Int Enable**

6.2.25.13 Glob Interrupt Vendor Mask Field Enables (0x000C)

Bit(s)	Field Name	Default	Description
15	PMA Vendor Alarm Int Field Ena	0b	**PMA Vendor Alarm Int Field Enable**
14	PCS Vendor Alarm Int Field Ena	0b	**PCS Vendor Alarm Int Field Enable**
13	PHY XS Vendor Alarm Int Field Ena	0b	**PHY XS Vendor Alarm Int Field Enable**
12	AutoNeg Vendor Alarm Int Field Ena	0b	**Auto-Negotiation Vendor Alarm Int Field Enable**
11	GBE Vendor Alarm Int Field Ena	0b	**GBE Vendor Alarm Int Field Enable**
10:3	Reserved	0x0	Reserved.
2	Global Alarms 1 Int Field Ena	1b	**Global Alarms 1 Int Field Enable**
1	Global Alarms 2 Int Field Ena	0b	**Global Alarms 2 Int Field Enable**
0	Global Alarms 3 Int Field Ena	1b	**Global Alarms 3 Int Field Enable**

6.2.25.14 Glob Interrupt Standard Mask LSA (0x000D)

Bit(s)	Field Name	Default	Description
15:0	Address LS	0xFF00	**Address LS**

6.2.25.15 Glob Interrupt Standard Mask MSA (0x000E)

Bit(s)	Field Name	Default	Description
15:10	Reserved	0x0	Reserved.
9:5	Port	0x0	**Port**
4:0	Device	0x1E	**Device** 0x0 = None 0x1 = PMA/PMD 0x3 = PCS 0x7 = Auto-negotiation 0x1D = Clause22 Ext 0x1E = Global All other values are reserved.

6.2.25.16 Glob Interrupt Standard Mask Data (0x000F)

Bit(s)	Field Name	Default	Description
15	PMA Standard Alarm 1 Int Ena	0b	**PMA Standard Alarm 1 Int Enable**
14	PMA Standard Alarm 2 Int Ena	0b	**PMA Standard Alarm 2 Int Enable**
13	PCS Standard Alarm 1 Int Ena	0b	**PCS Standard Alarm 1 Int Enable**
12	PCS Standard Alarm 2 Int Ena	0b	**PCS Standard Alarm 2 Int Enable**
11	PCS Standard Alarm 3 Int Ena	0b	**PCS Standard Alarm 3 Int Enable**
10	PHY XS Standard Alarm 1 Int Ena	0b	**PHY XS Standard Alarm 1 Int Enable**
9	PHY XS Standard Alarm 2 Int Ena	0b	**PHY XS Standard Alarm 2 Int Enable**
8	AutoNeg Standard Alarm 1 Int Ena	0b	**Auto-Negotiation Standard Alarm 1 Int Enable**
7	AutoNeg Standard Alarm 2 Int Ena	0b	**Auto-Negotiation Standard Alarm 2 Int Enable**
6	GbE Standard Alarm 1 Int Ena	0b	**GbE Standard Alarm 1 Int Enable**
5:1	Reserved	0x0	Reserved.
0	All Vendor Alarms Int Ena	1b	**All Vendor Alarms Int Enable**

6.2.25.17 Glob Interrupt Standard Mask Field Enables (0x0010)

Bit(s)	Field Name	Default	Description
15	PMA Standard Alarm 1 Int Field Ena	0b	**PMA Standard Alarm 1 Int Field Enable**
14	PMA Standard Alarm 2 Int Field Ena	0b	**PMA Standard Alarm 2 Int Field Enable**
13	PCS Standard Alarm 1 Int Field Ena	0b	**PCS Standard Alarm 1 Int Field Enable**
12	PCS Standard Alarm 2 Int Field Ena	0b	**PCS Standard Alarm 2 Int Field Enable**
11	PCS Standard Alarm 3 Int Field Ena	0b	**PCS Standard Alarm 3 Int Field Enable**
10	PHY XS Standard Alarm 1 Int Field Ena	0b	**PHY XS Standard Alarm 1 Int Field Enable**
9	PHY XS Standard Alarm 2 Int Field Ena	0b	**PHY XS Standard Alarm 2 Int Field Enable**
8	AutoNeg Standard Alarm 1 Int Field Ena	0b	**Auto-Negotiation Standard Alarm 1 Int Field Enable**
7	AutoNeg Standard Alarm 2 Int Field Ena	0b	**Auto-Negotiation Standard Alarm 2 Int Field Enable**
6	GbE Standard Alarm 1 Int Field Ena	0b	**GbE Standard Alarm 1 Int Field Enable**
5:1	Reserved	0x0	Reserved.
0	All Vendor Alarms Int Field Ena	1b	**All Vendor Alarms Int Field Enable**

6.2.25.18 CSR RAW1 (0x0011)

Raw data module length: variable

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word. Section Length = 3*n (n = number of CSRs to configure).

6.2.25.19 Section Footer (0x0012)

Bit(s)	Field Name	Default	Description
15:8	CRC8		**CRC8** CRC-8-CCITT: Start Section -> Word: PHY Config -> Section length End Section -> Word: PHY Config -> CSR RAW1
7:0	Reserved	0x0	Reserved. Block length in words.

6.2.26 PCIe Link (LCB) Configuration Section

Table 6-31. PCIe Link (LCB) Configuration Section Summary Table

Word Offset	Description	Section Number
0x0000	Section Length	6.2.26.1
0x0001	Reg Write Indirect List	6.2.26.2

6.2.26.1 Section Length (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Section Length		**Section Length** Length in: 2 Bytes unit - 1 First Section -> Word: PCIe Link (LCB) Configuration -> Section Length Last Section -> Word: PCIe Link (LCB) Configuration -> Reg Write Indirect List

6.2.26.2 Reg Write Indirect List (0x0001)

Raw data module length: variable

6.2.27 PCIe Analog Configuration Module Section

Table 6-32. PCIe Analog Configuration Module Section Summary Table

Word Offset	Description	Section Number
0x0000	Section Length	6.2.27.1
0x0001	PCI PHY FW	6.2.27.2

6.2.27.1 Section Length (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Section Length		**Section Length** Length in: 2 Bytes unit - 1 First Section -> Word: PCIe* Analog Configuration Module -> Section Length Last Section -> Word: PCIe* Analog Configuration Module -> PCI PHY FW

6.2.27.2 PCI PHY FW (0x0001)

Raw data module length: variable

6.2.28 2'nd Init Module Section

Table 6-33. 2'nd Init Module Section Summary Table

Word Offset	Description	Section Number
0x2000	Data	6.2.28.1

6.2.28.1 Data (0x2000)

Bit(s)	Field Name	Default	Description
15:0	Data	0xFFFF	**Data**

6.2.29 FCoE Scratch Pad Section

Length is (4KB * FCoE Scratch Pad Size). For 2M flash, max is 16KB (8192 words).

Table 6-34. FCoE Scratch Pad Section Summary Table

Word Offset	Description	Section Number
0x0000	Reserved	6.2.29.1

6.2.29.1 Reserved (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

6.2.30 Firmware Module Section

0x6000 -- 500K for each large section (2MB flash, 488K used for now).
0x4000 + FCoE size -- 512K for each large section (flash larger than 2MB).

Table 6-35. Firmware Module Section Summary Table

Word Offset	Description	Section Number
0x0000	FW 2M	6.2.30.1

6.2.30.1 FW 2M (0x0000)

Raw data module length: variable

6.2.31 PXE/OROM Module Section

0x45800 -- 500K for each large section (2MB flash, 488K used for now).
0x44000 + FCoE size -- 512K for each large section (flash larger than 2MB).

Table 6-36. PXE/OROM Module Section Summary Table

Word Offset	Description	Section Number
0x0000	OROM 2M	6.2.31.1

6.2.31.1 OROM 2M (0x0000)

Raw data module length: variable

Note: This word is preserved by Intel NVM update tool.

6.2.32 AQ PHY Module Section

0x83000 -- 500K for each large section (2MB flash, 488K used for now).
0x86000 + FCoE size -- 512K for each large section (flash larger than 2MB).

Table 6-37. AQ PHY Module Section Summary Table

Word Offset	Description	Section Number
0x0000	PHY 2M	6.2.32.1

6.2.32.1 PHY 2M (0x0000)

Raw data module length: variable

6.2.33 Free Provisioning Module Section

0xC1800 -- 500K for each large section (2MB flash, 488K used for now).
0xC6000 + FCoE size -- 512K for each large section (flash larger than 2MB).

Table 6-38. Free Provisioning Module Section Summary Table

Word Offset	Description	Section Number
0x0000	Reserved	6.2.33.1

6.2.33.1 Reserved (0x0000)

Bit(s)	Field Name	Default	Description
15:0	Reserved	0xFFFF	Reserved.

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

Chapter 7 Inline Functions

7.1 Receive Functionality

Packet reception consists of:

- Recognizing the presence of a packet on the wire (Section 7.1.1)
- Performing address filtering (Section 7.1.2)
- Optional IPsec decryption and authentication (Section 7.12)
- Checksum off-loads (Section 7.1.6)
- DMA queue assignment (Section 7.1.3)
- Storing the packet in the receive data FIFO (Section 7.6.3.1)
- Transferring the data to assigned receive queues in host memory (Section 7.1.4)
- Optional Receive Side Coalescing (Section 7.9)
- Updating the state of a receive descriptor (Section 7.1.5).

7.1.1 MAC Layer - Receive

7.1.1.1 Packet Acceptance Criteria

In addition to the filtering rules described in the next sections, a packet must also meet the following criteria to be accepted by the device:

1. Normally, only good packets are received (packets with none of the following errors: Under Size Error, Over Size Error, Packet Error, Length Error and CRC Error). However, if the *Store Bad Packets* bit is set (FCTRL.*SBP*), bad packets that don't pass the filter function are stored in host memory. Packet errors are indicated by error bits in the receive descriptor (RDESC.*ERRORS*). It is possible to receive all packets, regardless of whether they are bad, by setting the promiscuous enables bit and the *Store Bad Packets* bit. In this case, bad packets are queued according to the same rules as regular packets.
2. Min. Packet Size (Runt packets) — Rx packets, smaller than 48 bytes, cannot be posted to host memory regardless of save bad frame setting.
3. Max Packet Size — Any Rx packet posted from the MAC unit to the DMA unit cannot exceed 15.5 KB. Further restrictions per queue are described in Section 7.1.4.
4. CRC errors before the Start Frame Delimiter (SFD) are ignored. All packets must have a valid SFD to be recognized by the device (even bad packets).

7.1.1.2 CRC Strip

The X550 potentially strips the L2 CRC on incoming packets.

CRC strip is enabled by the HLREG0.*RXCRCSTRP* bit. When set, CRC is stripped from all received packets.

The global CRC strip bit (HLREG0.*RXCRCSTRP*) must be set in the following cases were the packet changes before being handled to the driver:

- RSC is enabled in any queue.
- VLAN is hidden (PFQDE.*HIDE_VLAN = 1*) in any queue.
- E-tag is removed in any queue (PFQDE.*STRIP_TAG* is set)
- L2 tags are stripped from packets (PFQDE.*STRIP_TAG = 1*) in any queue.
- IPsec is enabled.
- Time stamp is added to the packets in any TC (any bit of TSYNCRXCTL.*TSIP_UT_EN* or TSYNCRXCTL.*TSIP_UP_EN* is set).
- Short received packets are padded (RDRXCTL.*PSP* is set).

7.1.2 Packet Filtering

A received packet goes through up to three stages of filtering as depicted in Figure 7-1. The figure describes a switch-like structure that is used in virtualization mode to route packets between the network port (top of drawing) and one of many virtual ports (bottom of drawing), where each virtual port might be associated with a Virtual Machine (VM), a Virtual Machine Monitor (VMM), or the like. The three stages are:

1. **First stage — Admission Control:** Ensure that the packet should be received by the port. This is done by a set of L2 filters and is described in detail in this section.
2. **Second stage — Pooling:** This stage is specific to virtualization environments and defines the virtual ports (called pools in this document) that are the targets for the Rx packet. A packet can be associated with any number of ports/pools and the selection process is described in Section 7.1.3.2. In non virtualization mode, this stage is skipped and all the queues used in the next stage are considered as part of the same default pool.

   Note: A pool is equivalent to a VSI as defined in IEEE 802.1Qbg specification.

   Note: FCoE packets are not expected in virtual machines, and thus bypasses the pooling mechanism.
3. **Third stage — Queueing:** A receive packet that successfully passed the Rx filters is associated with one of many receive descriptor queues as described in Section 7.1.3.

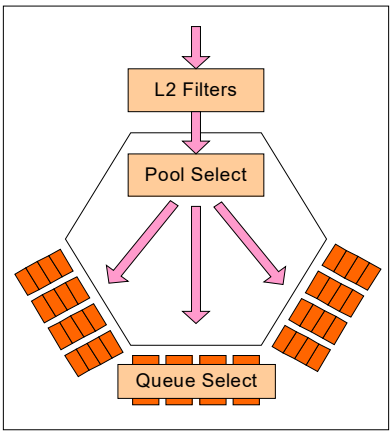


Figure 7-1. Stages in Packet Filtering - Virtualization Mode

The receive packet filtering role is to determine which of the incoming packets are allowed to pass to the local machine and which of the incoming packets should be dropped since they are not targeted to the local machine. Received packets that are targeted for the local machine can be destined to the host, to a manageability controller, or to both. This section describes how host filtering is done, and the interaction with management filtering.

As depicted in Figure 7-2, host filtering is done in two stages:

1. Packets are filtered by L2 filters (Ethernet MAC Address, unicast/multicast/broadcast). See Section 7.1.2.1.
2. Packets are filtered by VLAN if a VLAN tag is present. See Section 7.1.2.2.

A packet is not forwarded to the host if any of the following occurs:

- The packet does not pass L2 filters, as described in Section 7.1.2.1.
- The packet does not pass VLAN filtering, as described in Section 7.1.2.2.
- The packet passes manageability filtering and the manageability filters determine that the packet should not pass to the host (see MNGONLY register and Section 11.3.5.2).

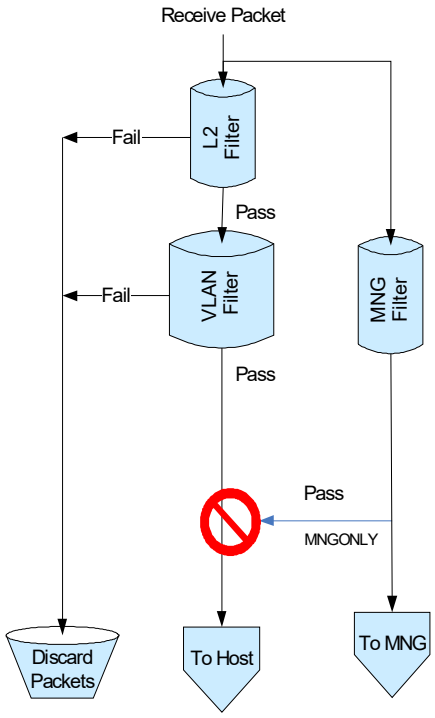


Figure 7-2. Rx Filtering Flow Chart

7.1.2.1 L2 Filtering

A packet passes successfully through L2 Ethernet MAC Address filtering if any of the following conditions are met:

- **Unicast packet filtering** — Promiscuous unicast filtering is enabled (FCTRL.*UPE* = 1b) or the packet passes unicast MAC filters (host).
- **Multicast packet filtering** — Promiscuous multicast filtering is enabled (FCTRL.*MPE* = 1b) or the packet matches one of the multicast filters.
- **Broadcast packet filtering to host** — Promiscuous multicast filtering is enabled (FCTRL.*MPE* = 1b) or Broadcast Accept Mode is enabled (*FCTRL.BAM* = 1b).

7.1.2.1.1 Unicast Filter

The Ethernet MAC Address is checked against the 128 host unicast addresses and 4 KB hash-based unicast address filters (if enabled). The host unicast addresses are controlled by the host interface. The destination address of an incoming packet must exactly match one of the pre-configured host address filters. These addresses can be unicast or multicast. Those filters are configured through Receive Address Low (RAL), Receive Address High (RAH), In addition, there are 4 KB unicast hash filters used for host defined by the PFUTA registers. The unicast hash filters are useful mainly for virtualization settings in those cases that more than 128 filters might be required.

Promiscuous Unicast — Receive all unicasts. Promiscuous unicast mode is usually used when the LAN device is used as a sniffer.

7.1.2.1.2 Multicast Filter (Partial)

The 12-bit portion of the incoming packet multicast address must be set in the multicast filter address table (*MTA*) to pass the partial multicast filter. The bits (out of 48 bits of the destination address) used to index the MTA table can be selected by the *MO* field in the MCSTCTRL register.

Promiscuous Multicast — Receive all multicasts. Promiscuous multicast mode is usually used when the LAN device is used as a sniffer.

7.1.2.2 VLAN Filtering

The X550 provides exact VLAN filtering as follows:

- Host VLAN filters are programmed by the VFTA[n] registers.
- A VLAN match might relate to the *DEI* bit in the VLAN header. It is enabled for host filtering only by the VLNCTRL.*DEIEN* while the expected value is defined by the VLNCTRL.*DEI*.

If double VLAN is enabled (see Section 7.4.5), filtering is done on the second (internal) VLAN tag. All the filtering functions of the X550 ignore the first (external) VLAN in this mode.

A receive packet that passes MAC Address filtering successfully is subjected to VLAN header filtering as illustrated in Figure 7-3:

1. If the packet does not have a VLAN header, it passes to the next filtering stage.
2. Else, if VLAN filters are not enabled (VLNCTRL.*VFE* = 0b), the packet is forwarded to the next filtering stage.
3. Else, if the packet matches an enabled VLAN filter and DEI checking (if enabled), the packet is forwarded to the next filtering stage.
4. Otherwise, the packet is dropped.

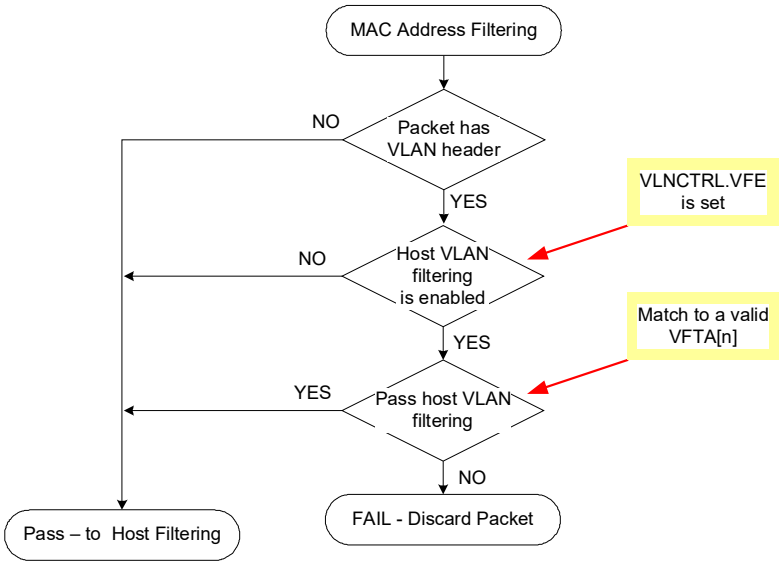


Figure 7-3. VLAN Filtering

7.1.2.3 E-tag Filtering

If the PFVTCTL.*POOLING_MODE* is E-tag (01b), special L2 filtering rules are applied to tagged packets.

- If the tag matches one of the RAH registers used for tag filtering (RAH.*ADTYPE* = 1), it is considered as a packet that passed L2.
- If the FCTRL.*TPE* bit is set (*Tag Promiscuous Enable*), all the tagged packets pass L2 filtering.
- Otherwise, packets with E-tag (if PFVTCTL.*POOLING_MODE*is E-tag (01b)) are dropped.

Note: E-tag packets are not expected when double VLAN are use, thus PFVTCTL.*POOLING_MODE* should be cleared if CTRL_EXT.*EXTENDED_VLAN* is set.

7.1.2.4 Manageability/Host Filtering

The host and manageability filtering process are mostly independent. Each entity defines which packet it should receive. The only exception is that the manageability filters can define a packet as exclusive and thus prevent it from reaching the host. See Section 11.3.5.2 for details. Figure 7-4 describes this flow.

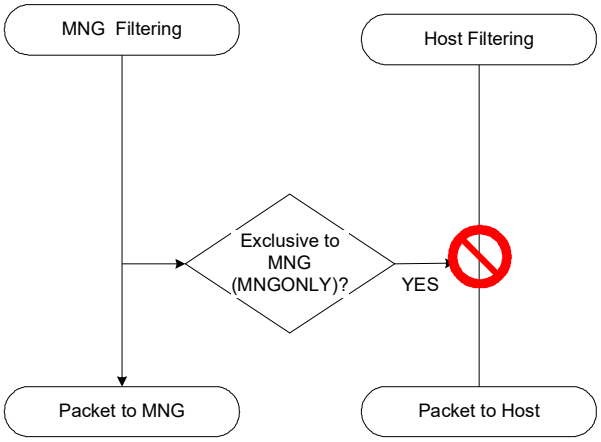


Figure 7-4. Manageability/Host Filtering

7.1.3 Rx Queues Assignment

The following filters/mechanisms determine the destination of a received packet. These filters are described briefly while more detailed descriptions are provided in the following sections:

- **Virtualization** — In a virtual environment, DMA resources are shared between more than one software entity (operating system and/or device driver). This is done by allocating receive descriptor queues to virtual partitions (VMM, IOVM, VMs, or VFs). Allocating queues to virtual partitions is done in sets, each with the same number of queues, called queue pools, or pools. Virtualization assigns to each received packet one or more pool indices. Packets are routed to a pool based on their pool index and other considerations such as DCB and RSS. See Section 7.1.3.2 for more on routing for virtualization.
- **DCB** — Data Center Bridging (DCB) provides QoS through priority queues, priority flow control, and congestion management. Packets are classified into one of several (up to eight) Traffic Classes (TCs). Each TC is associated with a single unique packet buffer. Packets that reside in a specific packet buffer are then routed to one of a set of Rx queues based on their TC value and other considerations such as RSS and virtualization. See Section 7.6 for details on DCB.
  - DCB is enabled via the *DCB_ENA* bit
- **Receive Side Scaling (RSS)** — RSS distributes packet processing between several processor cores by assigning packets into different descriptor queues. RSS assigns to each received packet an RSS index. Packets are routed to one queue from a set of Rx queues based on their RSS index and other considerations such as DCB and virtualization. See Section 7.1.3.7 for details.
- **L2 EtherType Filters** — These filters identify packets by their L2 EtherType and assigns them to receive queues. Examples of possible uses are LLDP packets, and 802.1X packets. See Section 7.1.3.3 for details. The X550 incorporates eight EtherType filters.
- **FCoE Redirection Table** — FCoE packets that match the L2 filters might be directed to a single legacy Rx queue or multiple queues to ease multi-core processing. See Section 7.1.3.4 for details. See also Section 7.10.3.3 for Large FC receive and direct data placement. Up to 64 queues (any 64 of the 128 queues) can be allocated for FCoE traffic by the FCoE redirection table defined by FCRETA[n] registers.
- **Flow Director Filters** — These filters that provide up to additional 32 K filters. See Section 7.1.3.6 for details.
- **TCP SYN Filters** — The X550 might route TCP packets with their SYN flag set into a separate queue. SYN packets are often used in SYN attacks to load the system with numerous requests for new connections. By filtering such packets to a separate queue, security software can monitor and act on SYN attacks. See Section 7.1.3.5 for details.

A received packet is allocated to a queue based on the above criteria and the following order:

- Queue by L2 EtherType filters (if match)
- Queue by FCoE redirection table (relevant for FCoE packets)
- Queue by SYN filter (if match)
- Queue by flow director filters
- Queue (in case of virtualization, within a pool) by DCB and/or RSS as described in Section 7.1.3.1 and Section 7.1.3.2.
- Send to queue zero.

7.1.3.1 Queuing in a Non-virtualized Environment

Table 7-1 lists the queuing schemes for packets that do not match any special filters (L2 EtherType, FCoE redirection, SYN and flow director filters). Table 7-2 illustrates the queue indexing. Selecting a scheme is done via the *Multiple Receive Queues Enable* (*MRQE*) field in MRQC register.

Table 7-1. Rx Queuing Schemes Supported (No Virtualization)

DCB	RSS	DCB/RSS Queues	Special Filters[1]
No	No	1 queue Rx queue 0	Supported
No	Yes	64 RSS queues	Supported
Yes	No	8 TCs x 1 queue 4 TCs x 1 queue Assign Rx queue 0 of each TC	Supported
Yes	Yes	8 TCs x 16 RSS 4 TCs x 32 RSS	Supported

1. Special filters include: L2 filters; FCoE redirection; SYN filter. When possible it is recommended to assign Rx queues not used by the DCB / RSS queues.

Table 7-2. Queue Indexing Illustration in Non-virtualization Mode

MRQC.*MRQE*	Queue Index Bits	6	5	4	3	2	1	0
0000b	Default only	0						
0001b	RSS		RSS					
0101b	DCB(4) + RSS	TC		RSS[1]				
0011b	DCB(4)	TC		0				
0100b	DCB(8) + RSS	TC			RSS[1]			
0010b	DCB(8)	TC			0			

1. The number of bits used for each TC is set according to the RQTC.*RQTCx* fields

A received packet is assigned to a queue according to the ordering shown in Figure 7-5:

- DCB and RSS filters — Packets that do not meet any of the previous filtering conditions described in Section 7.1.3 are assigned to one of 128 queues as listed in Table 7-1. The following modes are supported:
  - **No DCB, No RSS** — Queue 0 is used for all packets.
  - **RSS only** — A set of 64 queues is allocated for RSS. The queue is identified through the RSS index. Note that it is possible to use a subset of these queues.
  - **DCB only** — A single queue is allocated per TC to a total of eight queues (if the number of TCs is eight), or to a total of four queues (if the number of TCs is four). The queue is identified through the TC index.
  - **DCB with RSS** — A packet is assigned to one of 128 queues (8 TCs x 16 RSS or 4 TCs x 32 RSS) through the DCB traffic class of the packet and the RSS index. The TC index is used as the MS bits of the Rx queue index, and the LS bits are defined by the RSS index. The number of queues used for each TC are set according to the RQTC.*RQTCx* fields

When operating in conjunction with DCB, the number of RSS queues can vary per DCB TC. Each TC can be configured to a different number of RSS queues (1/2/4/8/16/32 queues). The output of the RSS redirection table is masked accordingly to generate an RSS index of the right width. When configured to less than the maximum number of queues, the respective MS bits of the RSS index are set to zero. The number of RSS queues per TC is configured in the RQTC register.

- Example — Assume a 4 TCs x 32 RSS configuration and that the number of RSS queues for TC=3 is set to 4. The queue numbers for TC=3 are 64, 65, 66, and 67 (decimal).

Figure 7-5 depicts the flow of allocation of Rx queues by the various queue filters previously described:

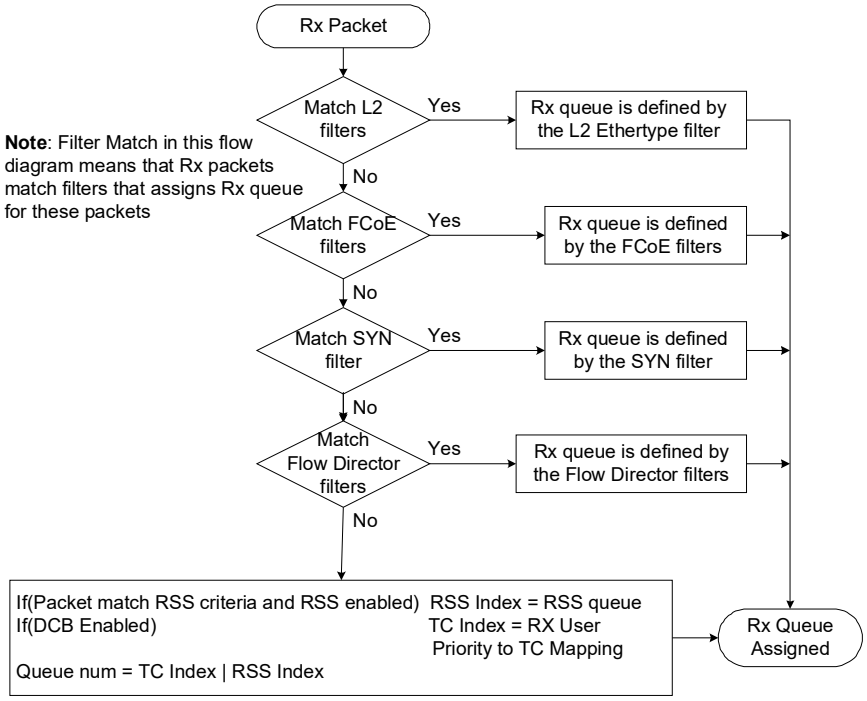


Figure 7-5. Rx Queuing Flow (Non-virtualized)

7.1.3.2 Queuing in a Virtualized Environment

The 128 Rx queues are allocated to a pre-configured number of queue sets, called pools. In non-IOV mode, system software allocates the pools to the VMM or to VMs. In IOV mode, each pool may be associated with a VF.

Incoming packets are associated with pools based on their L2 characteristics as described in Section 7.7.10. This section describes the following stage, where an Rx queue is assigned to each replication of the Rx packet as determined by its pools association.

Table 7-3 lists the queuing schemes supported with virtualization. Table 7-4 lists the queue indexing.

Table 7-3. Rx Queuing Schemes Supported with Virtualization

DCB	RSS	DCB/RSS Queues	Special Filters[1]
No	No	16 pools x 1 queue 32 pools x 1 queue 64 pools x 1 queue - Rx queue 0 of each pool	Supported
No	Yes	32 pools x 4 RSS 64 pools x 2 RSS	Supported
Yes	No	16 pools x 8 TCs 32 pools x 4 TCs	Supported
Yes	Yes	N/A	N/A

1. Special filters include: L2 filters; FCoE redirection; SYN filter. When possible it is recommended to assign Rx queues not used by the DCB / RSS queues.

Table 7-4. Queue Indexing Illustration in Virtualization Mode

MRQC.MRQE	Queue Index Bits	6	5	4	3	2	1	0
1000b		Pool Index						0
1011b	VT(64) + RSS	Pool Index						RSS
1001b/1111b	VT(64) + RSS + Mega Pool	Pool Index					VF Index/ RSS[1]	RSS
1010b	VT(32) + RSS	Pool Index					RSS	
N/A	VT(16) + RSS	Not Supported						
1101b	VT(32) + DCB(4)	Pool Index					TC	
1100b	VT(16) + DCB(8)	Pool Index				TC		

1. When *MRQC.MRQE* = 0xF, VF index for pools 0-59, part of RSS for pool 60, 62 and pools 61 and 63 are disabled.When *MRQC.MRQE* = 0x0, VF index for pools 0-61, part of RSS for pool 62 and pool 63 is disabled.

Selecting a scheme is done in the following manner:

- Non-IOV mode
  - Select one of the above schemes via the *Multiple Receive Queues Enable* field in the MRQC register.
- IOV mode
  - Determine the number of pools: the number must support the value configured by the operating system in the PCIe *NumVFs* field (see Section 9.2.4.4.5). Therefore, the number of pools is min. of {16, 32, 64} that is still >= NumVFs.
  - Determine DCB mode via the *DCB_ENA* bit in MTQC register.

A received packet is assigned to an absolute queue index according to the ordering shown in Figure 7-6). It is software responsibility to define a queue that belongs to the matched pool:

- DCB and RSS filters — The supported modes are listed in Table 7-3 and detailed as follows. The associated queue indexes are listed in Table 7-4.
  - **No DCB, No RSS** — A single queue is used as default queue of the pool with either 16, 32, or 64 pools enabled. In a 64 pools setting, queues '2xN'...'2xN+1' are allocated to pool 'N'; In a 32 pools setting, queues '4xN'...'4xN+3' are allocated to pool 'N'. The queues not used as default may be directed to by other filters.
  - **RSS only** — All 128 queues are allocated to pools. Configurations supported:
    - 32 pools with 4 RSS queues each
    - 64 pools with 2 RSS queues each.
    - 63 pools. The first 62 with 2 queues each and the last one with 4 RSS queues.In this mode the last pool is 62 and pool 63 is disabled.
    - 62 pools. The first 60 with 2 queues each and the last two with 4 RSS queues each. In this mode the last two pools are 60 and 62 and pools 61/63 are disabled.

      Note: It is possible to use a subset of the RSS queues in each pool. The LS bits of the queue indexes are defined by the RSS index, and the pool index is used as the MS bits.See description below.

      Note: In the 62 and 63 pools modes, the pools with 4 queues should be assigned to the PF and can not be assigned to a VF.
  - **DCB only** — All 128 queues are allocated to pools. Two configurations are supported: 16 pools with 8 TCs each or 32 pools with 4 TCs each. The LS bits of the queue indexes are defined by the TC index, and the pool index is used as the MS bits.
  - **DCB and RSS** — All 128 queues are allocated to pools. One configuration is supported: 16 pools with 4 TCs each TC with 2 RSS queues. The LS bit of the queue indexes is defined by the RSS index, the next 2 bits of the queue indexes are defined by the TC index, and the pool index is used as the MS bits.

When operating in conjunction with RSS, the number of RSS queues can vary per pool as defined by the PSRTYPE[n].*RQPL*. Each pool can be configured to a different number of RSS queues (1/2/4- queues) up to the maximum possible queues in the selected mode of operation. The output of the RSS redirection table is masked accordingly to generate an RSS index of the right width. When configured to less than the maximum number of queues, the respective MS bits of the RSS index are set to zero.

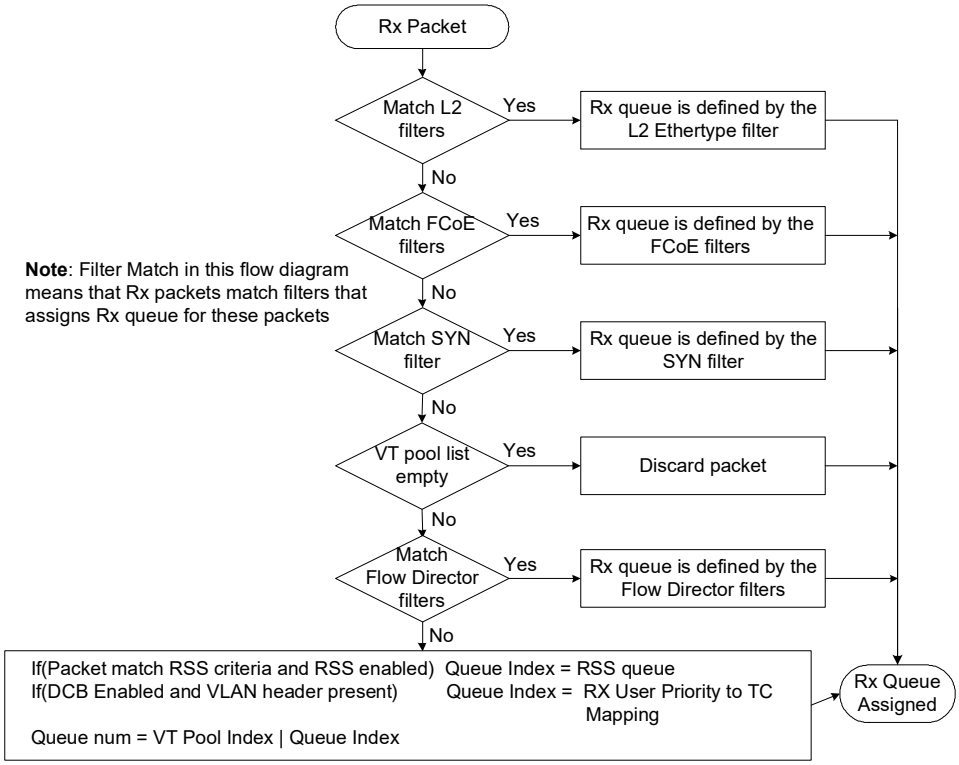


Figure 7-6. Rx Queuing Flow (Virtualization Case)

7.1.3.3 L2 EtherType Filters

These filters identify packets by their L2 EtherType, 802.1Q user priority and optionally assign them to a receive queue. The following possible usages have been identified at this time:

- DCB LLDP packets — Identifies DCB control packets
- IEEE 802.1X packets — Extensible Authentication Protocol (EAPOL) over LAN
- Time sync packets (such as IEEE 1588) — Identifies Sync or Delay_Req packets
- FCoE packets (possibly two UP values)
- The L2 type filters should not be set to IP packet type as this might cause unexpected results

The X550 incorporates eight EtherType filters defined by a set of two registers per filter: ETQF[n] and ETQS[n].

The L2 packet type is defined by comparing the *EtherType* field in the Rx packet with the ETQF[n].*ETYPE* (regardless of the pool and UP matching). The *Packet Type* field in the Rx descriptor captures the filter number that matched with the L2 EtherType. See Section 7.1.5.2.2 for a description of the *Packet Type* field.

The following flow is used by the EtherType filters:

1. If the ETQF.*FILTER_ENABLE* bit is cleared, the filter is disabled and the following steps are ignored.
2. Receive packet matches any ETQF filters if the *EtherType* field in the packet matches the *ETYPE* field of the filter. Note that the following steps are ignored if the packet does not match the ETQF filters.

   Note: EtherType Filters should not be configured in a way that may cause a packet to match multiple filters.
3. Packets that match any ETQF filters are candidate for the host. If the packet also matches the manageability filters, it is directed to the host as well regardless of the MNGONLY register setting.
4. If the ETQF.*FCOE* field is set, the packet is identified as an FCoE packet.
5. If the ETQF.*IEEE_1588_TIME_STAMP* field is set, the packet is identified as an IEEE 1588 packet.
6. If the ETQS.*QUEUE_ENABLE* bit is cleared, the filter completed its action on the packet. Else, the filter is also used for queuing purposes as described in the sections that follow.
7. If the ETQF.*POOL_ENABLE* field is set, the *Pool* field of the filter determines the target pool for the packet. The packet can still be mirrored to other pools as described in Section 7.7.10. See the sections that follow for more details on the use of the *Pool* field.
8. The ETQS.*RX_QUEUE* field determines the destination queue for the packet. In case of a mirrored packet, only the copy of the packet that is targeted to the pool defined by the *Pool* field in the *ETQF* register is routed according to the *Rx Queue* field.

Setting the ETQF[n] registers is described as follows:

- The *FILTER_ENABLE* bit enables identification of Rx packets by EtherType according to this filter. If this bit is cleared, the filter is ignored.
- The *ETYPE* field contains the 16-bit EtherType compared against all L2 type fields in the Rx packet.
- The *FCOE* bit indicates that the EtherType defined in the *EType* field is an FCoE EType. Packets that match this filter are identified as FCoE packets.
- The *IEEE_1588_TIME_STAMP* bit indicates that the EtherType defined in the *EType* field is identified as IEEE 1588 EType. Packets that match this filter are time stamped according to the IEEE 1588 specification.
- The *Pool* field defines the target pool for a packet that matches the filter.
  - It applies only in virtualization modes. The pool index is meaningful only if the *POOL_ENABLE* bit is set.
  - If the *POOL_ENABLE* bit is set, the *QUEUE_ENABLE* bit in the ETQS register must be set as well. In this case, the *RX_QUEUE* field in the ETQS must be part of the pool number defined in the ETQF.

Setting the ETQS[n] registers is described as follows:

- The *QUEUE_ENABLE* bit enables routing of the Rx packet that match the filter to Rx queue as defined by the *RX_QUEUE* field.
- The *RX_QUEUE* field contains the destination queue (one of 128 queues) for the packet.

Special considerations for virtualization modes:

- Packets that match an EtherType filter are diverted from their original pool (as defined by the VLAN and Ethernet MAC Address filters) to the pool defined in the *Pool* field in the ETQF registers.
- The same applies for multicast packets. A single copy is posted to the pool defined by the filter.
- Mirroring rules
  - A packet sent to a pool by an ETQF filter, is still candidate to mirroring using the standard mirroring rules.
  - The EtherType filter does not take part in the decision on the destination of the mirrored packet.

7.1.3.4 FCoE Redirection Table

The FCoE redirection table is a mechanism to distribute received FCoE packets into several descriptor queues. Software might assign each queue to a different processor, sharing the load of packet processing among multiple processors. The FCoE redirection table assigns Rx queues to packets that are identified as FCoE in the ETQF[n] registers but not assigned to queues in the ETQS[n] registers.

Figure 7-7 illustrates the computing of the assigned Rx queue index by the FCoE redirection table.

- The receive packet is parsed and the OX_ID is extracted for the filtering to Rx LAN queue (named as EX_ID).
- The six LS bits of the OX_ID are used as an address to the redirection table (FCRETA[n] register index).
- The FCoE redirection table is enabled by the FCRECTL.*ENA* bit. If enabled, the content of the selected FCRETA[n] register is the assigned Rx queue index.
- FCoE Packets with SNAP header are routed according to FCRETA[0]*.TABLE_ENTRY.*
- If the FCoE redirection table is disabled, FCoE packets are assigned to queue 0.

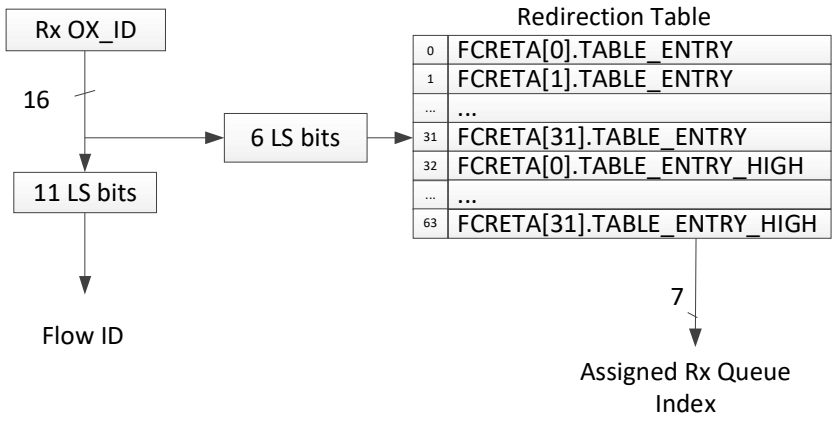


Figure 7-7. FCoE Redirection Table

7.1.3.5 SYN Packet Filters

The X550 might route TCP packets whose SYN flag is set into a separate queue. SYN packets are used in SYN attacks to load the system with numerous requests for new connections. By filtering such packets to a separate queue, security software can monitor and act on SYN attacks.

The following rules apply:

- A single SYN filter is provided.

The SYN filter is configured via the SYNQF register as follows:

- The *QUEUE_ENABLE* bit enables SYN filtering capability.
- The *RX_QUEUE* field contains the destination queue for the packet (one of 128 queues). In case of mirroring (in virtualization mode), only the original copy of the packet is routed according to this filter.

7.1.3.6 Flow Director Filters

The flow director filters identify specific flows or sets of flows and routes them to specific queues. The flow director filters are programmed by FDIRCTRL and all other FDIR registers. The X550 shares the Rx packet buffer for the storage of these filters.

The X550 supports three different modes of flow director (set in the FDIRCTRL.*FILTERMODE* field - Section 8.2.2.15.1):

- IP filtering (000b)
- MAC, VLAN filtering (001b)
- Cloud: NVGRE or VXLAN filtering (010b)

Table 7-5 describes the packets that are candidate to compared to the flow director filters according to the different modes:

Table 7-5. Candidate for Flow Director per Mode

Mode	IP packets TCP/UDP/SCTP (Not Tunneled)	Non IP Packets	NVGRE Packets	VXLAN packets	Other IP packets or Fragmented IP
00 - IP (L4 fields unmasked)	Candidate	Not Candidate	Candidate (inner IP)[1]	Candidate (inner IP)[1]	Not Candidate
00 - IP (L4 fields masked)	Candidate	Not Candidate	Candidate (inner IP)[2]	Candidate (inner IP)[2]	Candidate[2,3]
01 - MAC VLAN	Candidate	Candidate	Candidate	Candidate	Candidate
10 - Cloud	Not Candidate	Not Candidate	Candidate	Candidate	Not Candidate

1. Only if a known L4 header is present (TCP/UDP/SCTP).
2. VXLAN and NVGRE packets without inner IP header (e.g. tunneled ARP) are not candidate in IP mode.
3. IP in IP packets are not candidate in IP mode.

Basic rules for the flow director filters are:

In VT mode, the *Pool* field in FDIRCMD must be valid. If the packet is replicated, only the copy that goes to the pool that matches the *Pool* field is impacted by the filter.The flow director filters cover the following fields:

Table 7-6. Lookup Fields for Flow Director per Mode

Mode	Lookup Fields
00 - IP	• VLAN header • Source IP and destination IP Addresses • Source port and destination port numbers • IPv4 / IPv6[1] and UDP / TCP or SCTP protocol match • Flexible 2-byte tuple anywhere in the first 64 bytes of the packet • Target pool number (relevant only for VT mode)
01 - MAC VLAN	• MAC • VLAN • Flexible 2-byte tuple anywhere in the first 64 bytes of the packet • Target pool number (relevant only for VT mode)
10 - Cloud	• Tunnel type - NVGRE or VXLAN • TNI (NVGRE) / VNI (VXLAN) • Inner MAC • Inner VLAN • Flexible 2-byte tuple anywhere in the first 64 bytes of the packet • Target pool number (relevant only for VT mode)

1. IPv6 extended headers are parsed by the X550, enabling TCP layer header recognition. Still the IPv6 extended header fields are not taken into account for the queue classification by Flow Director filter. This rule do not apply for security headers and fragmentation header. Packets with fragmentation header miss this filter. Packets with security extended headers are parsed only up to these headers and therefore can match only filters that do not require fields from the L4 protocol.

The X550 supports two types of filtering modes (static setting by the FDIRCTRL.*PERFECT_MATCH* bit):

- **Perfect Match Filters** — The hardware checks a match between the masked fields of the received packets and the programmed filters. Masked fields should be programmed as zeros in the filter context. The X550 supports up to 8 K - 1 perfect match filters.
- **Signature Filters** — The hardware checks a match between a hash-based signature of the masked fields of the received packet. The X550 supports up to 32 K - 1 signature filters. This mode can be used only when the filtering mode is IPMODE (FDIRCTRL.*FILTERMODE* = 000b).

  Note: The *Perfect Match* fields and *Signature* field are denoted as *Flow ID* fields.

The X550 supports masking / range for the previously described fields. These masks are defined globally for all filters in the FDIR…M registers.

- The following fields can be masked per bit enabling power of two ranges up to complete enable / disable of the fields: IPv4 addresses and L4 port numbers.
- The following fields can be masked per byte enabling lower granularity ranges up to complete enable / disable of the fields: IPv6 addresses; Inner MAC; Outer MAC; *TNI/VNI* field.

  Note: In perfect match filters the destination IPv6 address can only be compared as a whole (with no range support) to the *IP6AT* filters. A match to any of the *IP6AT* filter is considered as an IPv6 destination match.

- The following fields can be either enabled or disabled completely for the match functionality: VLAN ID tag; Outer VLAN Priority + *DEI* bit (the user priority is taken from the outermost tag with *UP* bits if an inner VLAN exists; otherwise it is ignored. Inner or outer VLAN tag; Flexible 2-byte tuple and target pool. Target pool can be enabled by software only when VT is enabled as well.

Flow director filters have the following functionality in virtualization mode:

- Flow director filters are programmed by the registers in the PF described in Section 7.1.3.6.13 and Section 7.1.3.6.14.
- Flow director filters can be utilized in virtualization mode to filter on MAC-VLAN by setting FDIRCTRL.*FILTERMODE* = *MACVLANMODE* (001b). *MACVLANMODE* is only used in flow director perfect match mode.

7.1.3.6.1 Flow Director Filters Actions

Flow director filters might have one of the following actions programmed per filter in the *FDIRCMD* register:

- Drop packet or pass to host as defined by the *Drop* bit.
  - Matched packets to a flow director filter is directed to the assigned Rx queue only if the packet does not match the L2 filters for queue assignment nor the SYN filter for queue assignment.
  - Packets that match a filter are directed to the Rx queue defined in the filter context as programmed by the FDIRCMD.*RX_QUEUE*. The *RX_QUEUE* field is an absolute receive queue index. In a non-VT setting, it can be programmed to any value. In VT mode, the software should set the *RX_QUEUE* to an index that belongs to the matched pool.
  - Packets that match drop filters are directed to the Rx queue defined per all filters in the FDIRCTRL.*DROP_QUEUE*. The X550 drops these packets if software does not enable the specific Rx queue.

7.1.3.6.2 Flow Director Default Action

A default drop action may be applied by the flow director for packets that are candidate to flow director as defined in Table 7-5. If the FDIRCTRL.*DROP_NO_MATCH* bit is set, any packet candidate for flow director that does not match any of the filters is sent to the queue defined in the FDIRCTRL.*DROP_QUEUE* field.

7.1.3.6.3 Flow Director Filters Status Reporting

Shared status indications for all packets:

- The X550 increments the FDIRMATCH counter for packets that match a flow director filter. It also increments the FDIRMISS counter for packets that do not match any flow director filter.
- The *Flow Director Filter Match* (*FLM*) bit in the *Extended Status* field of the Rx descriptor is set for packets that match a flow director filter.
- The flow ID parameters are reported in the *Flow Director Filter ID* field in the Rx descriptor if enabled by the FDIRCTRL.*REPORT_STATUS*. When the *Report-Status* bit is set, the RXCSUM.*PCSD* bit should be set as well. This field is indicated for all packets that match or do not match the flow director filters.
  - For packets that do not match a flow director filter, if the FDIRCTRL.*REPORT_STATUS_ALWAYS* is set, the *Flow Director Filter ID* field can be used by software for future programming of a matched filter, otherwise, the RSS hash value is reported. Table 7-7 describes the value of the *RSS_TYPE* field in the receive descriptor in the different cases.

Table 7-7. RSS Type Values According to Flow Director Match

REPORT_STATUS (FDIRCTRL[5])	*REPORT_STATUS_ALWAYS* (FDIRCTRL[7])	Packet matches a Flow Director Entry	RSS Type Field Value
0	0	X	RSS Type
1	0	0	RSS Type
1	0	1	0xF
0	1	X	illegal configuration
1	1	X	0xF

For packets that match a flow director filter, the *Flow Director Filter ID* field can be used by software to identify the flow of the Rx packet. Too long linked list exception (linked list and too long terms are illustrated in Figure 7-8):

- The maximum recommended linked list length is programmed in the FDIRCTRL.*MAX_LENGTH* field
- The length exception is reported in the *FDIRERR* field in the Rx descriptor
- Packets that do not match any flow director filter, reports this exception if the length of the existing linked list is above the maximum recommended length. Software can use it to avoid further programming of additional filters to this linked list before other filters are removed.
- Packets that match a pass filter report this exception if the distance of the matched filter from the beginning of the linked list is higher than the above recommended length.
- Packets that match a drop filter are posted to the Rx queue programmed in the filter context instead of the global FDIRCTRL.*DROP_QUEUE*. The drop exception is reported in addition to the length exception (in the same field in the Rx descriptor).

Collision exception:

- Packets that matches a collided filter report this exception in the *FDIRERR* field in the Rx descriptor.
- Collision events for signature-based filters should be rare. Still it might happen because multiple flows can have the same hash and signature values. Software might leave the setting as is while the collided flows are handled according to the actions of the first programmed flow. Only one flow (out of the collided ones) might remain in the flow director filters. To clear the collision indication in the programmed filter, software should remove the filter and then re-program it once again.
- Collision events for a perfect match filter should never happen. A collision error might indicate a programming fault that software might decide to fix.

7.1.3.6.4 Flow Director Filters Block Diagram

Figure 7-8 shows a block diagram of the flow director filters. Received flows are identified to buckets by a hash function on the relevant tuples as defined by the FDIR...M registers. Each bucket is organized in a linked list indicated by the hash lookup table. Buckets can have a variable length while the last filter in each bucket is indicated as a last. There is no upper limit for a linked list length during programming. However, a received packet that matches a filter that exceeds the FDIRCTRL.*MAX_LENGTH* are reported to software (see Section 7.1.3.6.6).

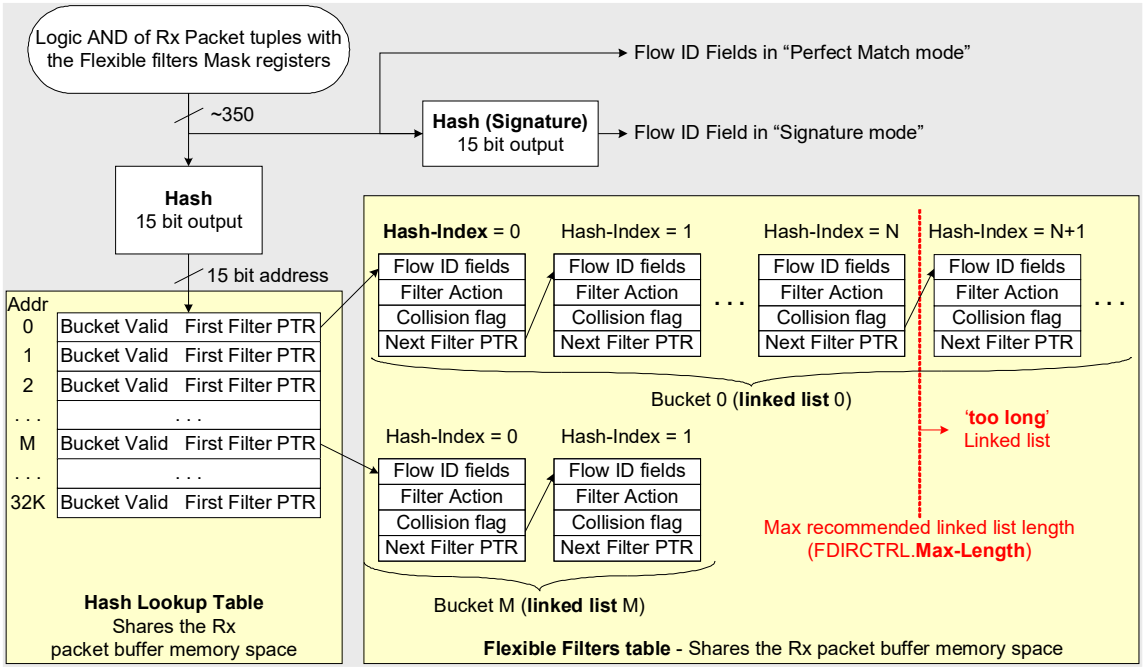


Figure 7-8. Flow Director Filters Block Diagram

7.1.3.6.5 Rx Packet Buffer Allocation

Flow director filters can consume zero space (when disabled) up to ~256 KB of memory. As shown in Figure 7-8, flow director filters share the same memory with the Rx packet buffer. By setting the *PBALLOC* field in the *FDIRCTRL* register, the software can enable and allocate memory for the flow director filters. The memory allocated to received traffic is the remaining part of the Rx packet buffer.

Table 7-8. Rx Packet Buffer Allocation

PBALLOC (2)	Effective Rx Packet Buffer Size (see following note)	Flow Director Filters Memory	Supported Flow Director Filters			
			Signature		Perfect Match	
			Filters	Bucket Hash	Filters	Bucket Hash
00 Flow Director is disabled	384 KB	0	0	N/A	0	N/A
01	320 KB	64 KB	8 K -1 filters	13 bit	2 K -1 filters	11 bit
10	256 KB	128 KB	16 K -1 filters	14 bit	4 K -1 filters	12 bit

Table 7-8. Rx Packet Buffer Allocation [continued]

PBALLOC (2)	Effective Rx Packet Buffer Size (see following note)	Flow Director Filters Memory	Supported Flow Director Filters			
			Signature		Perfect Match	
			Filters	Bucket Hash	Filters	Bucket Hash
11	128 KB	256 KB	32 K -1 filters	15 bit	8 K -1 filters	13 bit

Notes:

- It is the user's responsibility to ensure that sufficient buffer space is left for reception of traffic. The required buffer space for receive traffic depends on the number of traffic classes, and flow control upper and lower threshold values. If flow director is enabled (*PBALLOC* > 0), software should set the *RXPBSIZE[n]* registers according to the total remaining part of the Rx packet buffer for reception of traffic.
- When allocating 256 KB to the Flow Director table, it is likely that the 128 KB packet buffer left is too small to permit operating the port in DCB enabled mode.
- Refer to Section 3.7.4.3.2 through Section 3.7.4.3.5 for recommended setting of the Rx packet buffer sizes and flow control thresholds.

7.1.3.6.6 Flow Director Filtering Reception Flow

- Rx packet is digested by the filter unit which parse the packet extracting the relevant tuples for the filtering functionality.
- The X550 calculates a 15-bit hash value out of the masked tuples (logic mask of the tuples and the relevant mask registers) using the hash function described in Section 7.1.3.6.16.
- The address in the hash lookup table points to the selected linked list of flow director filters.
- The X550 checks the *Bucket Valid* flag. If it is inactive, the packet does not match any filter. Otherwise, *Bucket Valid* flag is active, proceed for the next steps.
- The X550 checks the linked list until it reaches the last filter in the linked list or until a matched filter is found.
- Case 1: Matched filter is found:
  - Increment the FDIRMATCH statistic counter.
  - Process the filter's actions (queue assignment) according to queue assignment priority. Meaning, the actions defined in this filter takes place only if the packet did not match any L2 filter or SYN filter that assigns an Rx queue to the packet.
  - Rx queue assignment according to the filter context takes place if *QUEUE_EN* is set.
  - If the *DROP* bit is set in the filter context, the packet is sent to the queue pointed by the *FDIRCTRL.DROP_QUEUE* field.
  - Post the packet to host including the flow director filter match indications as described in Section 7.1.3.6.3.
- Case 2: Matched filter is not found:
  - Increment the FDIRMISS statistic counter.
  - Post the packet to host including the flow director filter miss indications as described in Section 7.1.3.6.3.

7.1.3.6.7 Add Filter Flow

The software programs the filters parameters in the registers described in Section 7.1.3.6.13 and Section 7.1.3.6.14 while keeping the FDIRCMD.*FILTER_UPDATE* bit inactive. As a result, the X550 checks the bucket valid indication in the hash lookup table (that matches the FDIRHASH.*HASH*) for the presence of an existing linked list. Following are the two programming flows for the case a link list exists or for the case a new list is required.

- Case 1: Add a filter to existing linked list:

  The X550 checks the linked list until it reaches the last filter in the list or until a matched filter is found. The following cases may occur:

  — Matched filter is found (equal flow ID) with the same action parameters — The programming is discarded silently. This is a successful case since the programmed flow is treated as requested.

  — Matched filter is found (equal flow ID) with different action parameters — The X550 keeps the old setting of the filter while setting the *Collision* flag in the filter context (see Section 7.1.3.6.3 for software handling of collision during packet reception).

  — Matched filter is found (equal flow ID) with different action parameters and the *Collision* flag is already set — The programming is discarded silently. Software gets the same indications as the previous case.

  — Matched filter is not found (no collision) — The X550 checks for a free space in the flow director filters table.

  — No space case — Discard programming; increment the *FADD* counter in the FDIRFSTAT register and assert the flow director interrupt. Following this interrupt software should read the FDIRFSTAT register and FDIRFREE.*FREE* field, for checking the interrupt cause.

  — Free space is found — Good programming case: Add the new filter at the end of the linked list while indicating it as the last one. Program the *Next Filter PTR* field and then clear the *Last* flag in the filter that was previously the last one.

- Case 2: Create a new linked list:

  The X550 looks for an empty space in the flow director filters table:

  — Handle no empty space the same as in Case 1.

  — Good programming case: Add the new filter while indicating it as the last one in the linked list. Then, program the hash lookup table entry by setting the *Valid* flag and the *First Filter PTR* pointing to the new programmed filter.

Additional successful add flow indications:

- Increment the *ADD* statistic counter in the FDIRUSTAT register.
- Reduce the *FREE* counter in the FDIRFREE register, and then indicate the number of free filters. If the *FREE* counter crosses the FULL_THRESH value in the FDIRCTRL register, assert the flow director filter interrupt. Following this interrupt software should read the FDIRFSTAT register and FDIRFREE.*FREE* field, for checking the interrupt cause.
- Compare the length of the new linked list with *MAXLEN* in the FDIRLEN register. If the new linked list is longer than *MAXLEN*, update the FDIRLEN by the new flow.

7.1.3.6.8 Update Filter Flow

In some applications, it is useful to update the filter parameters, such as the destination Rx queue. Programing filter parameters is described in Section 7.1.3.6.7.

Setting the *FILTER_UPDATE* bit in the FDIRCMD register has the following result:

- Case 1: Matched filter does not exist in the filter table — Setting the *FILTER_UPDATE* bit has no impact and the command is treated as add filter.
- Case 2: Matched filter already exists in the filter table — Setting the *FILTER_UPDATE* bit enables filter parameter's update while keeping the collision indication as is.

When updating an existing filter the software device driver should program the same filter (i.e. the same FDIRHASH.*HASH* and the same FDIRHASH.*SIGNATURE_SW_INDEX*) while keeping the *FDIRCMD.FILTER_UPDATE* = 1

7.1.3.6.9 Remove Filter Flow

Software programs the filter Hash and Signature/Software-Index in the FDIRHASH register. It then should set the FDIRCMD.*CMD* field to *Remove Flow*. Software might use a single 64-bit access to the two registers for atomic operation. As a result, the X550 follows these steps:

1. Check if such a filter exists in the flow director filters table.
2. If there is no flow, increment the *FREMOVE* counter in the FDIRFSTAT register and skip the next steps.
3. If the requested filter is the only filter in the linked list, invalidate its entry in the hash lookup table by clearing the *Valid* bit.
4. Else, if the requested filter is the last filter in the linked list, invalidate the entry by setting the *Last* flag in the previous filter in the linked list.
5. Else, invalidate its entry by programming the *Next Filter PTR* in the previous filter in the linked list, pointing it to the filter that was linked to the removed filter.

Additional indications for successful filter removal:

1. Increment the remove statistic counter in the FDIRUSTAT register.
2. Increment the *FREE* counter in the FDIRFREE register.

7.1.3.6.10 Remove All Flow Director Filters

In some cases there is a need to clear the entire flow director table. It might be useful in some applications that might cause the flow director table becoming too occupied. Then, software might clear the entire table enabling its re-programming with new active flows.

Following are steps required to clear the flow director table:

- Poll the FDIRCMD.*CMD* until it is zero indicating any previous pending commands to the flow director table is completed (at worst case the FDIRCMD.*CMD* should be found cleared on the second read cycle). Note that software must not initiate any additional commands (add/remove/query) before this step starts and until this flow completes.
- Clear the FDIRFREE register (set *FREE* field to zero).
- Set FDIRCMD.*CLEARHT* to 1b and then clear it back to 0b.
- Clear the FDIRHASH register to zero.

- Re-write FDIRCTRL by its previous value while clearing the *INIT_DONE* flag.
- Poll the *INIT_DONE* flag until it is set to 1b by hardware.
- Clear the following statistic registers: FDIRUSTAT, FDIRFSTAT, FDIRMATCH, FDIRMISS, FDIRLEN (note that some of these registers are read clear and some are read write).

7.1.3.6.11 Flow Director Filters Initializing Flow

Following a device reset, the flow director is enabled by programming the FDIRCTRL register, as follows:

- Set *PBALLOC* to non-zero value according to the required buffer allocation to reception and flow director filter (see Section 7.1.3.6.5). All other fields in the register should be valid as well (according to required setting), as the FDIRCTRL register is expected to be programmed by a single cycle.
- Poll the *INIT_DONE* flag until it is set to 1b by hardware (expected initialization flow should take about 55 µs at 10 Gb/s and 550 µs at 1 Gb/s.

Note: The value of the FDIRCTRL.*PERFECT_MATCH* and FDIRCTRL.*FILTERMODE* should be modified only when no filters are set (at init time).

7.1.3.6.12 Query Filter Flow

Software might query specific filter settings and bucket length using the Query command.

- Program the filter *HASH* and *SIGNATURE_SW_INDEX* in the FDIRHASH register, and set the *CMD* field in the FDIRCMD register to 11b (Query Command). A single 64-bit access can be used for this step.
- As a result, the X550 provides the query result in the FDIRHASH, FDIRCMD and FDIRLEN registers (described in the sections as follows).
- Hardware indicates query completion by clearing the FDIRCMD.*CMD* field. The following table lists the query result.

Table 7-9. Query Filter Flow

Query Outcome	FDIRHASH -> Bucket Valid	FDIRCMD -> Filter Valid	FDIRLEN -> Bucket Length	FDIRCMD -> Filter ID Fields	FDIRCMD -> Filter Action
Empty Bucket	0	0	N/A	N/A	N/A
Valid Bucket, Matched Filter Not Found	1	0	Bucket linked list length	N/A	N/A
Found Signature Filter	1	1	Filter index within the linked list	0	Filter's parameters[1]
Found Perfect Match Filter	1	1	Filter index within the linked list	Filter's parameters	Filter's parameters

1. The pool parameter is not returned as part of the filter action in signature mode.

7.1.3.6.13 Signature Filter Registers

The signature flow director filter is programmed by setting the FDIRHASH and FDIRCMD registers. These registers are located in consecutive 8-byte aligned addresses. Software should use a 64-bit register to set these two registers in a single atomic operation. Table 7-10 lists the recommended setting.

Table 7-10. Signature Match Filter Parameters

Filter Bucket Parameters — FDIRHASH	
Hash	15-bit hash function used to define a bucket of filters. This parameter is part of the flow director filter ID that can be reported in the Rx descriptor. It is shared for signature and perfect match filters.
Valid	Should be set to 1b. It is shared for signature and perfect match filters.
Flow ID — FDIRHASH	
Signature	15-bit hash function used as the flow matching field. This parameter is also part of the flow director filter ID that can be reported in the Rx descriptor.
FDIRCMD — Programming Command and Filter action — See Section 8.2.2.15.8 for all fields descriptions.	

7.1.3.6.14 Perfect Match Filter Registers

Perfect match filters are programmed by the following registers: FDIRSIPv6[n], FDIRVLAN, FDIRPORT, FDIRIPDA, FDIRIPSA, FDIRHASH, FDIRCMD. Setting the FDIRCMD register, generates the actual programming of the filter. Therefore, write access to this register must be the last cycle after all other registers contain a valid content. Table 7-11 lists the recommended setting.

Note: Software filter programming must be an atomic operation. In a multi-core environment, software must ensure that all registers are programmed in a sequence with no possible interference by other cores.

Table 7-11. Perfect Match Filter Parameters

Filter Bucket Parameters and Software Index — FDIRHASH			
Hash	See Section 7.1.3.6.13.		
Valid	See Section 7.1.3.6.13.		
Software-Index	16-bit index provided by software at filter programming used by software to identify the matched flow. This parameter is also part of the flow director filter ID that can be reported in the Rx descriptor.		
FDIRCMD — Programming Command and Filter Action Set			
	IP Mode *FILTERMODE* = 000b	**MAC, VLAN Mode *FILTERMODE* = 001b**	**Cloud Modes *FILTERMODE* = 010b**
IPV6DMATCH	Should be set for IPv6 filters only and only if the destination address should be compared to one of the IPAT filters.	Should be cleared	
L4TYPE	Defines if the filter is for TCP, UDP or SCTP flows.	Should be cleared FDIRM.*L4P* should be set	
IPV6	Defines if the filter is for IPv4 or IPv6 addresses.	Should be cleared FDIRM.*L3P* should be set FDIRM.*DIPV6* should be set	

Table 7-11. Perfect Match Filter Parameters [continued]

TUNNEL_FILTER	Defines if the filter is a VXLAN and NVGRE tunneled packet filter. If set, the parameters of the L3 and L4 header relates to the tunneled (inner) header.	Should be cleared.	
POOL	Pool to which the packet is directed (relevant only in virtualization modes).		
Flow ID — Perfect Match Flow ID Parameters are Listed in the Following Registers and Fields			
	IP Mode *FILTERMODE* = 000b	**MAC, VLAN Mode *FILTERMODE* = 001b**	**Cloud Mode *FILTERMODE* = 010b**
FDIRSIPv6[0…2].*IP6SA*	Three LS DWord of the source IPv6. Meaningful for IPv6 flows depending on the FDIRIP6M.SIPM setting.	{FDIRSIPv6_0[31:0], FDIRSIPv6_1[15:0]} holds the destination MAC Address value FDIRSIPv6_1[31:16], and FDIRSIPv6_2[31:0] are reserved and should be set to 0x0. The FDIRIP6M.*SIPM* field should be set to 0x0xFC0F	FDIRSIPv6_2[31:0] holds the 32 bits of the *TNI/VNI* field. {FDIRSIPv6_0[31:0], FDIRSIPv6_1[15:0]} holds the destination MAC Address value FDIRSIPv6_1[31] holds the cloud mode (0 = NVGRE, 1 = VXLAN). FDIRSIPv6_1[30:16] are reserved and should be set to 0x0. To mask one of the fields the matching bits in FDIRIP6M.*SIPM* should be set: • Bits [15:12] control the masking of the *TNI/VNI* field. All the 32 bits of the key can be used for filtering. • Bit [11] controls the masking of the cloud mode and should be cleared for normal operation • Bits [3:0] and Bit [10] should be set. • Bits [9:4] control the masking of the inner MAC Address.
FDIRVLAN.*VLAN*	VLAN fields are meaningful depending on the FDIRM.VLANID and FDIRM.*VLANP* setting. The VLAN field should be stored in Little endian format.		Inner VLAN fields are meaningful depending on the FDIRM.*VLANID* and FDIRM.*VLANP* setting.The VLAN field should be stored in Little endian format.
FDIRVLAN.*FLEX*	Flexible 2-byte field at offset FDIRCTRL.*FLEX_OFFSET*. Meaningful depending on FDIRM.*FLEX* setting.		
FDIRPORT.*Source*	L4 source port. Meaningful for TCP, UDP and SCTP packets depending on the FDIRTCPM.*SportM*, FDIRUDPM.*SportM* and FDIRSCTPM.*SportM* setting.	Reserved - Should be zero. FDIRM.*L4P* should be set. FDIRTCPM.*SportM*, FDIRUDPM.*SportM* and FDIRSCTPM.*SportM* should be set.	
FDIRPORT.*Destination*	L4 destination port. Meaningful for TCP, UDP and SCTP packets depending on the FDIRTCPM.*DportM*, FDIRUDPM.*DportM* and FDIRSCTPM.*DPORTM* setting.	Reserved - Should be zero. FDIRM.*L4P* should be set. FDIRTCPM.*DportM*, FDIRUDPM.*DportM* and FDIRSCTPM.*DportM* should be set.	

Table 7-11. Perfect Match Filter Parameters [continued]

FDIRIPDA.*IP4DA*	IPv4 destination address. Meaningful depending on the FDIRDIP4M.*IPM* setting.	Reserved - Should be zero. All bits in FDIRDIP4M.*IPM* should be set.
FDIRIPSA.*IP4SA*	IPv4 source address or LS DWord of the source IPv6 address. Meaningful for IPv4 flows depending on the FDIRSIP4M.*IPM* setting and for IPv6 flows depending on the FDIRIP6M.*SIPM* setting.	Reserved - Should be zero. All bits in FDIRDSIP4M.*IPM* should be set. FDIRIP6M.*SIPM[3:0]* should be set.

7.1.3.6.15 Multiple CPU Cores Considerations

Perfect match filters programming and any query cycles require access to multiple registers. To avoid races between multiple cores, software might need to use one of the following programming methods:

- Use a software-based semaphore between the multiple cores for gaining control over the relevant CSR registers for complete programming or query cycles.
- Manage all programming and queries of the flow director filters by a single core.

Programming signature filters requires only the FDIRHASH and FDIRCMD registers. These two registers are located in 8-byte aligned adjacent addresses. Software could use an 8-byte register for the programming of these registers in a single atomic operation, which avoids the need for any semaphore between multiple cores.

7.1.3.6.16 Flow Director Hash Function

The X550 supports programmable 16-bit hash functions based on two 32-bit keys, one for the lookup table identifying a bucket of filters and another one for the signature (*FDIRHKEY* and *FDIRSKEY*). The hash function is described in the sections that follows. In some cases, a smaller hash value than 16 bits is required. In such cases, the LS bits of the hash value are used.

```
For (i=0 to 350) {if (Ext_K[i]) then Hash[15: 0] = Hash[15: 0] XOR Ext_S[15+i: i]}
```

While using the following notations:

'XOR'	Bitwise XOR of two equal length strings
If (xxx)	Equals ‘true’ if xxx = ‘1’ and equals ‘false’ if xxx = ‘0’
S[335:0]	The input bit string of the flow director tuples: 42 bytes listed in Table 7-12 AND-logic with the filters masks.
Ext_S[n]	S[14:0] | S[335:0] | S[335:321] // concatenated
K[31:0]	The hash key as defined by the FDIRHKEY or FDIRSKEY registers.
Tmp_K[11*32-1:0]	(Temp Key) equals K[31:0] | K[31:0]... // concatenated Key 11 times
Ext_K[350:0]	(Extended Key) equals T_K[351:1]

The input bit stream for the hash calculation is listed in Table 7-12 while byte 0 is the MSByte (first on the wire) of the VLAN, byte 2 is the MSByte of the source IP (IPv6 case) and so on.

Table 7-12. Input Bit Stream for Hash Calculation

Bytes/Mode	Field		
	IP	MAC, VLAN	Cloud Mode
Bytes 0…1	VLAN tag (always the outer VLAN)		Inner VLAN Tag
Bytes 2…5	Source IP (16 bytes for IPv6; source IP for IPv4 | 12 bytes of zero's)	Zero	
Bytes 6...11		MAC	Inner MAC
Bytes 12...13		Zero	8 bits of zeros | bit of tunnel_type | 7 bits of zeros
Bytes 14...17		Zero	VNI/TNI
Bytes 18…33	Signature Mode: Destination IP (16 bytes for IPv6; destination IP for IPv4 | 12 bytes of zero's) Exact Mode: Destination IP (7 zero bits |IP6AT match indication | 120 zero bits; destination IP for IPv4 | 12 bytes of zero's)[1]		
34...37	L4 source port number | L4 destination port number		
38...39	Flexible bytes		
40	00b | pool number (as defined by FDIRCMD.Pool)		
41	[7:5] 000b [4] FDIRCMD.TUNNEL_FILTER (should be zero in MAC, VLAN mode). [3] 0b [2] IPv6/IPv4 type (FDIRCMD.IPV6) [1:0] L4 type (FDIRCMD.L4TYPE)		

1. In VXLAN and NVGRE packets, the IP Addresses used are of the tunneled header in other packets the outer IP header is used.

7.1.3.7 RSS

RSS is a mechanism to distribute received packets into several descriptor queues. Software can then assign each queue to a different processor, therefore sharing the load of packet processing among several processors.

As described in Section 7.1, the X550 uses RSS as one ingredient in its packet assignment policy (the others are the various filters, DCB and virtualization). The RSS output is an RSS index. The X550 global assignment uses these bits (or only some of the LSBs) as part of the queue number.

Figure 7-9 and Figure 7-10 show the process of computing an RSS output:

1. The receive packet is parsed into the header fields used by the hash operation (such as IP Addresses, TCP port, etc.)
2. A hash calculation is performed. The X550 supports a single hash function, as defined by Microsoft* (MSFT) RSS. The X550 therefore does not indicate to the device driver which hash function is used. The 32-bit result is fed into the *RSS Hash* field in the packet receive descriptor.
3. The X550 supports two modes of RSS defined by the MRQC.*MULTIPLE_RSS* bit. When set to 0b a single RSS key and redirection table are supported. In this mode (usually used when virtualization is not enabled) the nine LSBs of the hash result are used as an index into a 512-entry redirection table. Each entry provides up to 6-bit RSS output index (Figure 7-9).

   For SRIOV or VMDQ enabled modes (set by MRQC.*MULTIPLE_RSS* set to 1b), The X550 supports up to 64 (one per pool) RSS keys and redirection tables (both can be controlled and programmed at the VF space). In this mode the 6 LSBs of the hash result are used as an index into a 64-entry redirection table. Each entry provides up to 2-bit RSS output index (Figure 7-10).

When RSS is enabled, the X550 provides software with the following information as required by Microsoft* RSS and provided for device driver assist:

- A DWord result of the MSFT RSS hash function application to the packet header, to be used by the stack for flow classification, is written into the receive packet descriptor. A 4-bit RSS *Type* field conveys the hash function used for the specific packet.
- Packets to which the RSS function cannot be applied (for example non IP packets) return an *RSS Hash* result of zero, an *RSS Type* of zero and an RSS output index of zero.

7.1.3.7.1 Enabling RSS

- RSS is enabled in the MRQC register.
- RSS enabling cannot be done dynamically and must be preceded by a software reset.
- RSS status field in the descriptor write-back is enabled when the RXCSUM.*PCSD* bit is set (fragment checksum is disabled). RSS is therefore mutually exclusive with UDP fragmentation checksum offload.
- Support for RSS is not provided when legacy receive descriptor format is used.

7.1.3.7.2 Disabling RSS

- Disabling RSS on the fly is not allowed, and the X550 must be reset after RSS is disabled.
- When RSS is disabled, packets are assigned an RSS output index = zero.

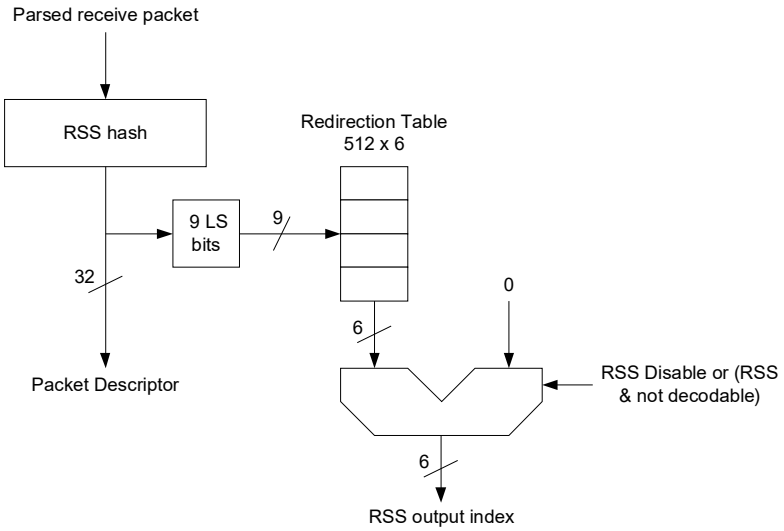


Figure 7-9. RSS Block Diagram (*MULTIPLE_RSS* = 0b)

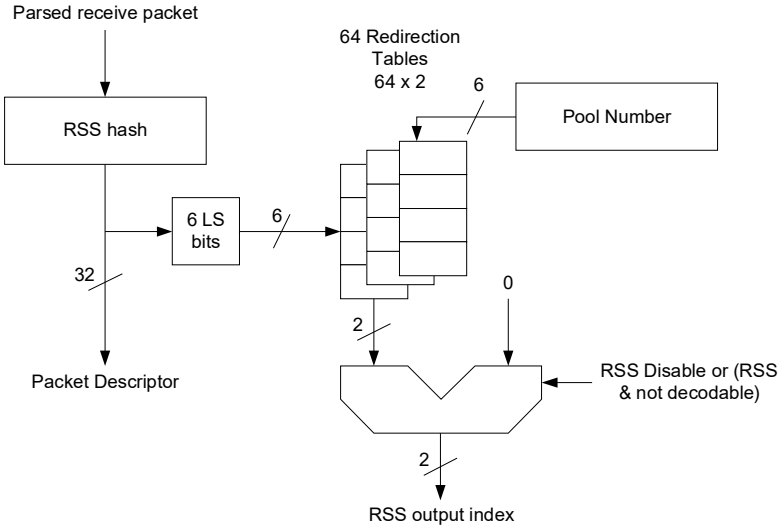


Figure 7-10. RSS Block Diagram (*MULTIPLE_RSS* = 1b)

7.1.3.7.3 RSS Hash Function

This section provides a verification suite used to validate that the hash function is computed according to MSFT nomenclature.

The X550's hash function follows the MSFT definition. A single hash function is defined with several variations for the following cases:

- **TcpIPv4** — The X550 parses the packet to identify an IPv4 packet containing a TCP segment per the criteria described below. If the packet is not an IPv4 packet containing a TCP segment, RSS is not done for the packet.
- **IPv4** — The X550 parses the packet to identify an IPv4 packet. If the packet is not an IPv4 packet, RSS is not done for the packet.
- **TcpIPv6** — The X550 parses the packet to identify an IPv6 packet containing a TCP segment per the criteria described below. If the packet is not an IPv6 packet containing a TCP segment, RSS is not done for the packet.
- **IPv6** — The X550 parses the packet to identify an IPv6 packet. If the packet is not an IPv6 packet, RSS is not done for the packet.

Tunneled IP to IP packets are considered for the RSS functionality as IP packets. The RSS logic ignores the L4 header while using the outer (first) IP header for the RSS hash.

For NVGRE and VXLAN packets the inner header (IP and TCP/UDP) is used for RSS computations.

The following additional cases are not part of the MSFT RSS specification but are supported RSS modes:

- **UdpIPV4** — The X550 parses the packet to identify a packet with UDP over IPv4.
- **UdpIPV6** — The X550 parses the packet to identify a packet with UDP over IPv6.

A packet is identified as containing a TCP segment if all of the following conditions are met:

- The transport layer protocol is TCP (not UDP, ICMP, IGMP, etc.).
- The TCP segment can be parsed (such as IPv4 options or IPv6 extensions can be parsed, packet not encrypted, etc.).
- The packet is not fragmented (even if the fragment contains a complete L4 header).

Note: IPv6 extended headers are parsed by the X550, enabling TCP layer header recognition. Still the IPv6 extended header fields are not taken into account for the queue classification by RSS filter. This rule do not apply for security headers and fragmentation header. Packets with fragmentation header miss this filter. Packets with security extended headers are parsed only up to these headers and therefore can match only filters that do not require fields from the L4 protocol.

Bits[31:16] of the Multiple Receive Queues Command (MRQC), for single RSS and VFMRQC for multiple RSS, registers enable each of the above hash function variations (several might be set at a given time). If several functions are enabled at the same time, priority is defined as follows (skip functions that are not enabled):

- IPv4 packet:
  - Try using the TcpIPv4 function
  - Try using UdpIPv4 function
  - Try using the IPv4 function
- IPv6 packet:
  - Try using the TcpIPv6 function.
  - Try using UdpIPv6 function.
  - Try using the IPv6 function

The following combinations are currently supported:

- Any combination of IPv4, TcpIPv4, and UdpIPv4.

  And/or:

- Any combination of either IPv6, TcpIPv6, and UdpIPv6.

When a packet cannot be parsed by the previous rules, it is assigned an RSS output index = zero. The 32-bit tag (normally a result of the hash function) equals zero.

The 32-bit result of the hash computation is written into the packet descriptor and also provides an index into the redirection table.

The following notation is used to describe the following hash functions:

- Ordering is little endian in both bytes and bits. For example, the IP Address 161.142.100.80 translates into 0xa18e6450 in the signature.
- A "^" denotes bit-wise XOR operation of same-width vectors.
- @x-y denotes bytes x through y (including both of them) of the incoming packet, where byte 0 is the first byte of the IP header. In other words, we consider all byte-offsets as offsets into a packet where the framing layer header has been stripped out. Therefore, the source IPv4 address is referred to as @12-15, while the destination v4 address is referred to as @16-19.
- @x-y, @v-w denotes concatenation of bytes x-y, followed by bytes v-w, preserving the order in which they occurred in the packet.

All hash function variations (IPv4 and IPv6) follow the same general structure. Specific details for each variation are described in the following section. The hash uses a random secret key of length 320 bits (40 bytes); the key is stored in the RSS Random Key Register RSSRK for single RSS and VFRSSRK[63:0] for multiple RSS.

The algorithm works by examining each bit of the hash input from left to right. Our nomenclature defines left and right for a byte-array as follows: Given an array K with k bytes, our nomenclature assumes that the array is laid out as follows:

- K[0] K[1] K[2] … K[k-1]

K[0] is the left-most byte, and the MSB of K[0] is the left-most bit. K[k-1] is the right-most byte, and the LSB of K[k-1] is the right-most bit.

ComputeHash(input[], N)

```
For hash-input input[] of length N bytes (8N bits) and a random secret key K of 320 bits
Result = 0;
For each bit b in input[] {
if (b == 1) then Result ^= (left-most 32 bits of K);
shift K left 1 bit position;
}
return Result;
```

7.1.3.7.3.1 Pseudo-code Examples

The following four pseudo-code examples are intended to help clarify exactly how the hash is to be performed in four cases, IPv4 with and without ability to parse the TCP header, and IPv6 with and without a TCP header.

Hash for IPv4 with TCP:

Concatenate SourceAddress, DestinationAddress, SourcePort, DestinationPort into one single byte-array, preserving the order in which they occurred in the packet: Input[12] = @12-15, @16-19, @20-21, @22-23.

```
Result = ComputeHash(Input, 12);
```

Hash for IPv4 with UDP:

Concatenate SourceAddress, DestinationAddress, SourcePort, DestinationPort into one single byte-array, preserving the order in which they occurred in the packet: Input[12] = @12-15, @16-19, @20-21, @22-23.

```
Result = ComputeHash(Input, 12);
```

Hash for IPv4 without TCP:

Concatenate SourceAddress and DestinationAddress into one single byte-array

```
Input[8] = @12-15, @16-19
Result = ComputeHash(Input, 8)
```

Hash for IPv6 with TCP:

Similar to previous:

```
Input[36] = @8-23, @24-39, @40-41, @42-43
Result = ComputeHash(Input, 36)
```

Hash for IPv6 with UDP:

Similar to previous:

```
Input[36] = @8-23, @24-39, @40-41, @42-43
Result = ComputeHash(Input, 36)
```

Hash for IPv6 without TCPL:

```
Input[32] = @8-23, @24-39
Result = ComputeHash(Input, 32)
```

7.1.3.7.4 Redirection Tables

The redirection table used for single RSS mode (RETA - Section 8.2.2.8.18 and ERETA - Section 8.2.2.8.24) is a 512-entry structure, indexed by the nine LSBs of the hash function output.

The redirection tables used for multiple RSS mode (VFRETA - Section 8.3.2.3.12) are 64-entry structures indexed by the six LSBs of the hash function output. The table to use is defined by the pool to which the packet is sent.

System software might update the redirection tables during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

7.1.3.7.5 RSS Verification Suite

Assume that the random key byte-stream is:

```
0x6d, 0x5a, 0x56, 0xda, 0x25, 0x5b, 0x0e, 0xc2,
0x41, 0x67, 0x25, 0x3d, 0x43, 0xa3, 0x8f, 0xb0,
0xd0, 0xca, 0x2b, 0xcb, 0xae, 0x7b, 0x30, 0xb4,
0x77, 0xcb, 0x2d, 0xa3, 0x80, 0x30, 0xf2, 0x0c,
0x6a, 0x42, 0xb7, 0x3b, 0xbe, 0xac, 0x01, 0xfa
```

Table 7-13. IPv4

Destination Address/Port	Source Address/Port	IPv4 only	IPv4 with TCP
161.142.100.80:1766	66.9.149.187:2794	0x323e8fc2	0x51ccc178
65.69.140.83:4739	199.92.111.2:14230	0xd718262a	0xc626b0ea
12.22.207.184:38024	24.19.198.95:12898	0xd2d0a5de	0x5c2b394a
209.142.163.6:2217	38.27.205.30:48228	0x82989176	0xafc7327f
202.188.127.2:1303	153.39.163.191:44251	0x5d1809c5	0x10e828a2

Note: The IPv6 address tuples are only for verification purposes, and may not make sense as a tuple.

Table 7-14. IPv6

Destination Address/Port	Source Address/Port	IPv6 only	IPv6 with TCP
3ffe:2501:200:3::1 (1766)	3ffe:2501:200:1fff::7 (2794)	0x2cc18cd5	0x40207d3d
ff02::1 (4739)	3ffe:501:8::260:97ff:fe40:efab (14230)	0x0f0c461c	0xdde51bbf
fe80::200:f8ff:fe21:67cf (38024)	3ffe:1900:4545:3:200:f8ff:fe21:67cf (44251)	0x4b61e985	0x02d1feef

7.1.4 Receive Data Storage in System Memory

The X550 posts receive packets into data buffers in system memory.

The following controls are provided for the data buffers:

- The SRRCTL[n].*BSIZEPACKET* field defines the size of the data buffer pointed by each descriptor. The maximum packet size that can be posted to a queue can be limited using the RXDCTL[n].*RLPML* field. This filter enables software to use smaller buffers than the size defined by the SRRCTL[n].*BSIZEPACKET*.

  Note: The packet size compared to RXDCTL[n].*RLPML* does not include any parts stripped by the device like CRC VLAN or other tags but include additional Timestamp appended to the packet.

- The SRRCTL[n].*BSIZEHEADER* field defines the size of the header buffer pointed by each descriptor (advanced descriptors only).
- Each queue is provided with a separate SRRCTL register.

Receive memory buffer addresses are word (2 x byte) aligned (both data and headers).

The internal receive buffers are described in Section 7.6.3.1.

7.1.5 Receive Descriptors

7.1.5.1 Legacy Receive Descriptor Format

A receive descriptor is a data structure that contains the receive data buffer address and fields for hardware to store packet information. Upon receipt of a packet for this device, hardware stores the packet data into the indicated buffer and writes the length, status and errors to the receive descriptor. If SRRCTL[n].*DESCTYPE* = zero, the X550 uses the Legacy Rx descriptor as listed in Table 7-15. The shaded areas indicate fields that are modified by hardware upon packet reception (so-called descriptor write-back).

Legacy descriptors should not be used when advanced features are enabled: SCTP, Virtualization, DCB, IPsec, FCoE, Time stamp in Packet, tunnel packets checksum or RSC. Packets that match these cases might be dropped from queues that use legacy receive descriptors.

Refer to Table 7-15 and the field descriptions that follow.

Table 7-15. Legacy Receive Descriptor (RDESC) Layout

	63 48	47 40	39 32	31 16	15 0
0	Buffer Address [63:0]				
8	VLAN Tag	Errors	Status	Fragment Checksum	Length

Buffer Address (64-bit offset 0, 1st line)

Physical address in host memory of the received packet buffer.

Length Field (16-bit offset 0, 2nd line)

The length indicated in this field covers the data written to a receive buffer including CRC bytes (if any). Software must read multiple descriptors to determine the complete length for packets that span multiple receive buffers.

Fragment Checksum (16-bit offset 16, 2nd line)

This field is used to provide the fragment checksum value. This field is equal to the unadjusted 16-bit ones complement of the packet. Checksum calculation starts at the L4 layer (after the IP header) until the end of the packet excluding the CRC bytes. To use the fragment checksum assist to offload L4 checksum verification, software might need to back out some of the bytes in the packet. For more details see Section 7.1.6.5.

The fragment checksum is always reported in the descriptor with the EOP bit set.

Status Field (8-bit offset 32, 2nd line)

Status information indicates whether the descriptor has been used and whether the referenced buffer is the last one for the packet. Error status information is listed in Table 7-17.

Table 7-16. Receive Status (RDESC.*STATUS*) Layout

7	6	5	4	3	2	1	0
PIF	IPCS	L4CS	UDPCS	VP	Reserved	EOP	DD

End of Packet (EOP) and Descriptor Done (DD)

Refer to the following table:

DD	EOP	Description
0	0	Software setting of the descriptor when it hands it to the hardware.
0	1	Reserved (invalid option).
1	0	A completion status indication for non-last descriptor of a packet that spans across multiple descriptors. It means that the hardware is done with the descriptor and its buffers while only the length is valid on this descriptor.
1	1	A completion status indication of the entire packet. Software might take ownership of its descriptors while all fields in the descriptor are valid.

VP (VLAN Packet)

When set, the *VP* field indicates that the incoming packet's type is a VLAN (802.1q, matching the VLNCTRL.*VET*). If the RXDCTL.*VME* bit is set as well, an active *VP* field also means that the VLAN has been stripped from the packet to the receive descriptor. For a further description of 802.1q VLANs, see Section 7.4.

IPCS (IPv4 Checksum), L4CS (L4 Checksum), UDPCS (UDP Checksum)

These bits are described in the following table.

Note: Switched packets from a local VM that do not use the Tx IP checksum offload by hardware have the IPCS equal to zero; switched packets from a local VM that do not use the Tx L4 checksum offload by hardware have the *L4CS* and *UDPCS* equal to zero.

L4CS	UDPCS	IPCS	Functionality
0	0	0	Hardware does not provide checksum offload.
0	0	1	Hardware provides IPv4 checksum offload. Pass/fail indication is provided in the *Error* field – *IPE*.
1	0	1/0	Hardware provides IPv4 checksum offload if IPCS is active along with TCP checksum offload. Pass/fail indication is provided in the *Error* field – *IPE* and *TCPE*.
1	1	1/0	Hardware provides IPv4 checksum offload if IPCS is active along with UDP checksum offload. Pass/fail indication is provided in the *Error* field – *IPE* and *TCPE*.

IPv6 packets do not have the *IPCS* bit set, but might have the *L4CS* bit and *UDPCS* bit set if the X550 recognizes the transport header.

PIF (Non Unicast Address)

The *PIF* bit is set on packets with a non-unicast destination Ethernet MAC Address — multicast and broadcast.

Error Field (8-bit offset 40, 2nd line)

Table 7-17 and the following text describes the possible errors reported by the hardware.

Table 7-17. Receive Errors (RDESC.*ERRORS*) Layout

7	6	5	4	3	2	1	0
IPE	TCPE	Reserved	Reserved	Reserved	Reserved	Reserved	RXE

IPE (IPv4 Checksum Error)

The IP checksum error is valid only when the *IPCS* bit in the *Status* field is set (indicating that the hardware validated the IP checksum). This bit is meaningful only on the last descriptor of a packet while the *EOP* bit is set as well. Packets with IP error are posted to host memory regardless of the store bad packet setting (FCTRL.*SBP*).

TCPE (TCP/UDP Checksum Error)

The TCP/UDP checksum error is valid only when the *L4CS* bit in the *Status* field is set (indicating that the hardware validated the L4 checksum). This bit is meaningful only on the last descriptor of a packet while the *EOP* bit is set as well. Packets with a TCP/UDP error are posted to host memory regardless of the store bad packet setting (FCTRL.*SBP*).

IPv4/UDP packets that carry a null UDP checksum field are reported with *L4CS* = 0 and *TCPE* = 0 (valid packet with no checksum).

IPv6/UDP packets that carry a null UDP checksum field are reported with *L4CS* = 1 and *TPCE* = 1 (UDP checksum is mandatory over IPv6).

RXE

The RXE error bit is an indication for any MAC error. It is a logic OR function of the following errors:

- CRC or symbol error might be a result of receiving a /V/ symbol on the TBI interface, /FE/ symbol on the GMII/XGMII interface, RX_ER assertion on GMII interface, bad EOP or loss of sync during packet reception.
- Undersize frames shorter than 64 bytes.
- Oversize frames larger than the *MFS* definition in the MAXFRS register.
- Length error in 802.3 packet format. *Length* field is not checked in presence of a VLAN or E-Tag.

Packets with an RXE error are posted to host memory only when store bad packet bit (FCTRL.*SBP*) is set.

VLAN Tag Field (16-bit offset 48, 2nd line)

If the RXDCTL.*VME* is set and the received packet type is 802.1q (as defined by VLNCTRL.*VET*), the VLAN header is stripped from the packet data storage. In this case the 16 bits of the VLAN tag, priority tag and DEI from the received packet are posted to the *VLAN Tag* field in the receive descriptor. Otherwise, the *VLAN Tag* field contains 0x0000.

Table 7-18. VLAN Tag Field Layout (for 802.1q Packet)

15 13	12	11 0
PRI	DEI	VLAN

Priority and DEI are part of 802.1Q specifications. The VLAN field is provided in network order.

7.1.5.2 Advanced Receive Descriptors

7.1.5.2.1 Advanced Receive Descriptors — Read Format

Table 7-19 lists the advanced receive descriptor programming by the software. The SRRCTL[n].*DESCTYPE* should be set to a value other than 000b when using the advanced descriptor format.

Table 7-19. Descriptor Read Format

	63 1	0
0	Packet Buffer Address [63:1]	
8	Header Buffer Address [63:1]	DD

Packet Buffer Address (64)

The physical address in host memory of the packet buffer.

Header Buffer Address (64)

The physical address in host memory of the header buffer with the lowest bit being Descriptor Done (DD). When a packet spans in multiple descriptors, only the header buffer of the first descriptor is used. In subsequent descriptors, only the data buffer is used.

During the programming phase, software must set the *DD* bit to zero (see the description of the *DD* bit in this section). This means that header buffer addresses are always word aligned.

Note: The X550 does not support null descriptors meaning packet or header addresses are zero.

7.1.5.2.2 Advanced Receive Descriptors — Write-Back Format

When the X550 writes back the descriptors, it uses the format listed in Table 7-20. The advanced descriptor writeback format is used when SRRCTL[n]. DESCTYPE is set to a value other than 000b.

Table 7-20. Descriptor Write-Back Format

	63 48	47 32	31	30 21	20 17	16 4	3 0
0	RSS Hash / Fragment Checksum / FCoE_PARAM / Flow Director Filters ID		SPH	HDR_LEN	RSCCNT	Packet Type	RSS Type
8	VLAN Tag	PKT_LEN	Extended Error		Extended Status / NEXTP		
	63 48	47 32	31 20		19 0		

RSS Type (4-bit offset 0, 1st line)

The X550 must identify the packet type and then choose the appropriate RSS hash function to be used on the packet. The RSS type reports the packet type that was used for the RSS hash function.

RSS Type	Description
0x0	No hash computation done for this packet
0x1	HASH_TCP_IPv4
0x2	HASH_IPv4
0x3	HASH_TCP_IPv6
0x4	Reserved
0x5	HASH_IPv6
0x6	Reserved
0x7	HASH_UDP_IPv4
0x8	HASH_UDP_IPv6
0x9 – 0xE	Reserved
0xF	Packet reports flow director filters status (set if flow director match, even if packet is forwarded by another filter).

Packet Type (13-bit at offset 4, 1st line)

The *Packet Type* field reports the packet type identified by the hardware as follows. Note that some of the fields in the receive descriptor are valid for specific packet types.

Bit Index	Bit 11 = 0	Bit 11 = 1 (L2 packet matching one of the *ETQF* filters)
0	IPV4 — IPv4 header present	EtherType — *ETQF* register index that matches the packet. Special types are defined for 802.1X, 1588 and FCoE.
1	IPV4O — IPv4 with options	
2	IPV6 — IPv6 header present	
3	IPV6E- IPv6 with extensions	Reserved.
4	TCP — TCP header present	

Bit Index	Bit 11 = 0	Bit 11 = 1 (L2 packet matching one of the *ETQF* filters)
5	UDP — UDP header present	Reserved.
6	SCTP — SCTP header	
7	If bit 12 = 0 - Reserved If bit 12 = 1: 0b = NVGRE 1b = XLAN	
8	IPsec ESP – IPsec encapsulation	
9	IPsec AH – IPsec encapsulation	
10	Reserved	Reserved.
11	0b = non L2 packet	1b = L2 packet
12	NVGRE or VXLAN Tunnel Packet. If this bit is set, bit 7 indicates the tunnel type.	Reserved.

Note: UDP, TCP and IPv6 indications are not set in any IPv4 fragmented packet.

In virtualization mode, packets might be received from other local VMs. The X550 does not check the L5 header for these packets and does not report NFS header. If such packets carry IP tunneling (IPv4 — IPv6), they are reported as IPV4E. The packets received from local VM are indicated by the *LB* bit in the status field. To be identified, the *CC* bit in the transmit descriptor must be set and if it is a tunnel packet, the TUNNEL.*OUTERIPCS* must be set also. If bit 12 (Tunnel Packet) is set, all the L3/L4 indications (IPV4, IPV4O, IPV6, IPV6E, TCP, UDP, SCTP) reflects the inner header status.

RSC Packet Count- RSCCNT (4-bit offset 17, 1st line)

The *RSCCNT* field is valid only for RSC descriptors while in non-RSC it equals zero. *RSCCNT* minus one indicates the number of coalesced packets that start in this descriptor. *RSCCNT* might count up to 14 packets. Once 14 packets are coalesced in a single buffer, RSC is closed enabling accurate coalesced packet count. If the *RSCCNTBP* bit in RDRXCTL is set, coalescing might proceed beyond the 14 packets per buffer while *RSCCNT* stops incrementing beyond 0xF.

Note: Software can identify RSC descriptors by checking the *RSCCNT* field for non-zero value.

HDR_LEN (10-bit offset 21, 1st line)

The *HDR_LEN* reflects the size of the packet header in byte units (if the header is decoded by the hardware). This field is meaningful only in the first descriptor of a packet and should be ignored in any subsequent descriptors. Header split is explained in Section 7.1.6 while the packet types for this functionality are enabled by the PSRTYPE[n] registers.

Split Header — SPH (1-bit offset 31, 1st line)

When set to 1b, indicates that the hardware has found the length of the header. If set to 0b, the header buffer may be used only in split always mode. If the received header size is greater or equal to 1024 bytes, the *SPH* bit is not set and header split functionality is not supported. The *SPH* bit is meaningful in all the descriptors of a packet. See additional details on *SPH*, *PKT_LEN* and *HDR_LEN* as a function of split modes in Table 7-25.

Packet Type	Header Length (Includes All Fields Up to the Field Specified)	Header Split
Un-recognized EtherType only with/without SNAP and with/without VLAN, or packets that match the L2 filters (MTQF) other than FCoE with/without VLAN.	VLAN header(s) if present. Else, *EtherType* field	No
FCoE packet without ESP option header.	FC header including FC options	N/A[1]
FCoE packet with ESP option header.	FC header excluding FC options	N/A[1]
Pv4 only or fragmented IPv4 with any payload including IPv4-IPv6 tunneling.	IPv4 header	Enabled
Non-fragmented IPv4, TCP / UDP / SCTP.	L4 header	Enabled
IPv4-IPv6, only or fragmented IPv4-IPv6 at IPv6 header with any payload.	IPv6 header (up to the fragment extension header if exist)	Enabled
IPv4-IPv6,TCP / UDP / SCTP.	L4 header	Enabled
VXLAN or NVGRE.	Cloud Header/L2 Header	Enabled[2]

1. Header split is not permitted in queues that might receive FCoE packets.
2. If cloud split modes used (PSRTYPE15 or PSRTYPE16 are set). If not, according to internal header.

RSS Hash or FCOE_PARAM or Flow Director Filters ID (32-bit offset 32, 1st line) / Fragment Checksum (16-bit offset 48, 1st line)

This field has multiplexed functionality according to the received packet type (reported on the *Packet Type* field in this descriptor) and device setting.

FCoE_PARAM

For FCoE packets that matches a valid DDP context, this field holds the *PARAM* field in the DDP context after processing the received packet. If the *Relative Offset Present* bit in the F_CTL was set in the data frames, the *PARAM* field indicates the size in bytes of the entire exchange inclusive the frame reported by this descriptor. This field is valid for FCoE packets for which the *FCSTAT* field is non zero.

Fragment Checksum

For fragmented UDP/IP packets, this field holds the UDP fragment checksum (described in Section 7.1.6.5) if both the RXCSUM.*PCSD* bit is cleared and RXCSUM.*IPPCSE* bit is set. This field is meaningful only for UDP packets where the *UDPV* bit in the *Extended Status* word is set. When Fragment Checksum is reported, bits 47:32 are invalid.

The checksum does not include any padding or timestamp added by the device.

RSS Hash / Flow Director Filters ID

For non-FCoE packets, if the RXCSUM.*PCSD* bit is set, this field holds the RSS hash value or flow director filters ID.

- If the FDIRCTRL.*REPORT_STATUS* bit is set, the flow director filters ID is reported;
- If the RSS Type is non zero, the RSS hash is reported.
- Otherwise, the value is not valid.

Table 7-21. Checksum Enable/Disable

RXCSUM.*PCSD*	0 (Checksum Enable)	1 (Checksum Disable)
	Fragment Checksum and IP Identification are reported in the Rx Descriptor.	RSS Hash value is reported in the Rx Descriptor.

RSS Hash

The RSS hash value is required for RSS functionality as described in Section 7.1.3.7. Note that the RSS hash is meaningful only for 'RSS Type' in the range 0x1 to 0x8.

Flow Director Filters ID

The flow director filters ID is reported only when the received packet matches a flow directory filter (see Section 7.1.3.6). The flow director filter ID field has a different structure for signature-based filters and perfect match filters as follows:

Filter Type	31	30 29	28 16	15 13	12 0
Hash-based Flow Director Filter ID	Rsv	Bucket Hash		Signature	
Perfect Match Flow Director Filter ID	Rsv		Hash	Rsv	SW-Index

Bucket Hash — A hash value that identifies a flow director bucket. When the Flow Director table is smaller than 32 K filters the bucket hash is smaller than 15 bits. In this case the upper bit(s) are set to zero.

Signature — A hash value used to identify flow within a bucket.

SW-Index — The SW-Index that is taken from the filter context, programmed by software. It is meaningful only when the *FLM* bit in the Extended Status is set as well.

Rsv — Reserved.

Extended Status / NEXTP (20-bit offset 0, 2nd line)

Status information indicates whether the descriptor has been used and whether the referenced buffer is the last one for the packet. Table 7-22 lists the *Extended Status* word in the last descriptor of a packet (*EOP* bit is set). Table 7-23 lists the *Extended Status* word in any descriptor but the last one of a packet (*EOP* bit is cleared).

Table 7-22. Receive Status (RDESC.*STATUS*) Layout of *Last* Descriptor

19	18	17	16	15	14	13	12	11	10
BMC	LB	SECP	TS	TSIP	Rsv			Rsv	UDPV
VEXT	OUTERIPCS	PIF	IPCS	L4I	UDPCS	VP	FLM	EOP	DD
			FCEOFs	FCSTAT					
9	8	7	6	5	4	3	2	1	0

Table 7-23. Receive Status (RDESC.*STATUS*) Layout of *Non-Last* Descriptor

19 4	3:2	1	0
Next Descriptor Pointer — NEXTP	Rsv	EOP = 0b	DD

Rsv (14:11)

Reserved at zero.

FLM(2)

Flow director filter match indication is set for packets that match these filters.

VP(3), PIF (7)

These bits are described in the legacy descriptor format in Section 7.1.5. The VP bit is not set even if a VLAN tag is present if the PFQDE.*HIDE_VLAN* bit is set for the queue.

EOP (1) and DD (0)

End of Packet and *Done* bits are listed in the following table:

DD	EOP	Description
0	X	Software setting of the descriptor when it hands it to hardware.
1	0	A completion status indication for a non last descriptor of a packet (or multiple packets in the case of RSC) that spans across multiple descriptors. In a single packet case the *DD* bit indicates that the hardware is done with the descriptor and its buffers. In the case of RSC, the *DD* bit indicates that the hardware is done with the descriptor but might still use its buffers (for the coalesced header) until the last descriptor of the RSC completes. Only the *Length* fields are valid on this descriptor. In the RSC case, the next descriptor pointer and RSCCNT are valid as well.
1	1	A completion status indication of the entire packet (or the multiple packets in the case of RSC) and software might take ownership of its descriptors. All fields in the descriptor are valid (reported by the hardware).

UDPCS (4), L4I (5) / FCSTAT (5:4)

This field has multiplexed functionality for FCoE and non-FCoE packets. Hardware identifies FCoE packets in the filter unit and indicates it in the *Packet Type* field in the Rx descriptor. For non-FCoE packets this field is UDPCS and L4I. The UDPCS (UDP checksum) is set (together with L4CS bit) when hardware provides UDP checksum offload. The L4I (L4 Integrity) is set when hardware provides any L4 offload as: UDP checksum, TCP checksum or SCTP CRC offload. For FCoE packets, this field represents the FCSTAT (FCoE Status) as follows:

FCSTAT	Meaning
00	No match to any active FC context
01	FCoE frame matches an active FC context with no DDP. The entire frame is posted to the receive buffer indicated by this descriptor.
10	FCP_RSP frame received that invalidates an FC read context or last data packet in a sequence with sequence initiative set that invalidates an FC write context.
11	FCoE frame matches an active FC context and found liable for DDP by the filter unit. The packet's data was posted directly to the user buffers if no errors were found by the DMA unit as reported in the *FCERR* field. If any error is found by the DMA unit the entire packet is posted to the legacy queues.

IPCS(6), FCEOFs (6)

This bit has multiplexed functionality for FCoE and non-FCoE packets. The hardware identifies FCoE packets in the filter unit and indicates it in the *Packet Type* field in the Rx descriptor.

For non-FCoE packets it is IPCS as described in Legacy Rx descriptor (in Section 7.1.5).

For FCoE packets, this bit and the *FCEOFe* bit in the *Extended Error* field indicates the received EOF code as follows:

FCEOFe	FCEOFs	Description and Digested Meaning and Device Behavior
0	0	EOFn. Nominal operation, DDP is enabled.
0	1	EOFt. Nominal operation (end of sequence), DDP is enabled.
1	0	Unexpected EOFn-EOFt or SOFi-SOFn. No DDP while filter context is updated by the packet.
1	1	EOFa, EOFni or un-recognized EOF / SOF. No DDP while filter context is invalidated.

OUTERIPCS(8)

Indicates that a checksum was done on the outer IP header of an NVGRE or VXLAN packet.

VEXT (9)

Outer-VLAN is found on a double VLAN packet. This bit is valid only when CTRL_EXT.*EXTENDED_VLAN* is set. See more details in Section 7.4.5.

UDPV (10)

The *UDP Checksum Valid* bit indicates that the incoming packet contains a valid (non-zero value) checksum field in an incoming fragmented (non-tunneled) UDP IPv4 packet. It means that the *Fragment Checksum* field in the receive descriptor contains the UDP checksum as described in Section 7.1.6.5. When this field is cleared in the first fragment that contains the UDP header, it means that the packet does not contain a valid UDP checksum and the checksum field in the Rx descriptor should be ignored. This field is always cleared in incoming fragments that do not contain the UDP header.

TSIP (15)

Timestamp in packet. The *Timestamp In Packet* bit is set to indicate that the received packet arrival time was captured by the hardware and the timestamp was placed in the receive buffer. For more details see Section 7.7 and Section 7.1.6.2.

TS (16)

The *Time Stamp* bit is set when the device recognized a time sync packet. In such a case the hardware captures its arrival time and stores it in the Time Stamp register. For more details see Section 7.7.

SECP (17)

Security Processing bit indicates that the hardware identified the security encapsulation and processed it as configured.

IPsec processing — This bit is set only if a matched SA was found. Note that hardware does not process packets with an IPv4 option or IPv6 extension header and the *SECP* bit is not set. This bit is not set for IPv4 packets shorter than 70 bytes, IPv6 ESP packets shorter than 90 bytes, or IPv6 AH packets shorter than 94 bytes (all excluding CRC). Note that these packet sizes are never expected and set the length error indication in the *SECERR* field.

LB (18)

This bit provides a loopback status indication which means that this packet is sent by a local VM (VM to VM switch indication).

BMC (19)

Packet received from BMC. The *BMC* bit is set to indicate the packet was sent by the local BMC. Bit is cleared if packet arrives from the network. For more details see Section 11.4.2.2.

NEXTP (19:4)

Large receive might be composed of multiple packets and packets might span in multiple buffers (descriptors). These buffers are not guaranteed to be consecutive while the *NEXTP* field is a pointer to the next descriptor that belongs to the same RSC. The *NEXTP* field is defined in descriptor unit (the same as the head and tail registers). The *NEXTP* field is valid for any descriptor of a large receive (the *EOP* bit is not set) except the last one. It is valid even in consecutive descriptors of the same packet. In the last descriptor (on which the *EOP* bit is set), *NEXTP* is not indicated but rather all other status fields previously described in this section.

Extended Error (12-bit offset 21, 2nd line)

Table 7-24 and the following text describe the possible errors reported by hardware.

Table 7-24. Receive Errors (RDESC.*ERRORS*) Layout

11	10	9	8:7	6	5:4	3	2:0
IPE FCEOFe	L4E	RXE	SECERR	OUTERIPER	Rsv	HBO	FCERR / FDIRERR

FCERR (2:0)

Defines error cases for FCoE packets. Note that hardware indicates FCoE packet recognition on the *Packet Type* field in the Rx descriptor. Packets with FCERR are posted to host memory regardless of the store bad packet setting in the Filter Control register. This field must be ignored when FCSTAT is 00b.

FCERR Code	Meaning
000	No exception errors found
001	Bad FC CRC. Hardware does not check any other FC fields in the packet.
010	One of the following error indications found by the filter unit (hardware auto-invalidates a matched DDP filter context if exists): 1. Received non-zero abort sequence condition in the *F_CTL* field in FC read packet. 2. Received EOFa or EOFni or any un-recognized EOF or SOF flags.
011	The DMA unit gets FCoE packets that match a DDP context while it missed the packet that was marked as first by the filter unit. Filter context parameters might be updated while DMA context parameters are left intact (see error code 101b).
100	One of the following cases: 1. Unsupported FCoE version. *FCSTAT* equals to 00b. 2. Out of order reception (SEQ_CNT does not match expected value) of a packet that matches an active DDP context. The filter unit might set the *FCSTAT* to 01b, or 10b.
101	No DMA resources due to one of the following cases listed while the hardware auto-invalidates the DDP DMA context. Software should invalidate the filter context before it can reuse it. 1. Last buffer exhausted (no space in the user buffers). 2. Legacy receive queue is not enabled or no legacy receive descriptor. 3. Some cases of a missed packet as described in *FCERR* code 011b and 110b.
110	Filter context valid and DMA context invalid. Indicates that some packet(s) were lost by the DMA context due to lack of legacy receive descriptors or were missed by the Rx packet buffer. 1. This error code might also be received in some cases of a missed packet as described in FCERR code 011b.
111	Reserved

FDIRERR (2:0)

This field is relevant for non-FCoE packets when the flow director filters are enabled.

FDIRErr(0) - Length — If the flow director filter matches the *Length* bit, this indicates that the distance of the matched filter from the hash table exceeds the FDIRCTRL.Max-Length. If there is no matched filter, the *Length* bit is set if the flow director linked list of the matched hash value exceeds the FDIRCTRL.*Max-Length*.

FDIRErr(1) - Drop — The *Drop* bit indicates that a received packet matched a flow director filter with a drop action. In the case of perfect mode filtering, it is expected to find the drop indication only when the linked list in the flow director bucket exceeds the permitted Max-Length. In this case, the packet is not dropped. Instead, it is posted to the Rx queue (indicated in the filter context) for software handling of the Max-Length exception. In the case of hash mode filtering, it is expected that the drop queue is always a valid queue so all packets that match the drop filter are visible to software.

FDIRErr(2) - Coll — A matched flow director filter with a collision indication was found. The collision indicates that software attempted to step over this filter with a different action that was already programmed.

HBO (3)

The *Header Buffer Overflow* bit is set if the packet header (calculated by hardware) is bigger than the header buffer (defined by SSRCTL.*BSIZEHEADER*). *HBO* reporting might be used by software to allocate bigger buffers for the headers. It is meaningful only if the *SPH* bit in the receive descriptor is set as well. The *HDR_LEN* field is valid even when the *HBO* bit is set. Packets with HBO error are posted to host memory regardless of the store bad packet setting (FCTRL.SBP). Packet DMA to its buffers when the *HBO* bit is set, depends on the device settings as follows:

SRRCTL.DESCTYPE	DMA Functionality
Header Split (010b)	The header is posted with the rest of the packet data to the packet buffer.
Always Split Mode (101b)	The header buffer is used as part of the data buffers and contains the first SRRCTL.*BSIZEHEADER* bytes of the packet.

Rsv (5:4)

Reserved at zero.

OUTERIPER (6)

Indicates an error was found in the checksum of an outer IP header of a VXLAN or NVGRE packet or that the UDP checksum of the outer UDP header in a VXLAN packet was not zero.

SECERR (8:7)

Security error indication for IPsec. This field is meaningful only if the *SECP* bit in the extended status is set.

SECERR	IPsec Error Reporting
00	No errors found while an active SA found or no security processing.
01	Invalid IPsec Protocol: IPsec protocol field (*ESP* or *AH*) in the received IP header does not match expected one stored in the SA table.

SECERR	IPsec Error Reporting
10	Length error: ESP packet is not 4-bytes aligned or AH/ESP header is truncated (for example, a 28-byte IPv4 packet with IPv4 header + ESP header that contains only SPI and SN) or AH *Length* field in the AH header is different than 0x07 for IPv4 or 0x08 for IPv6 or the entire packet size excluding CRC is shorter than 70 bytes for IPv4 or 90 bytes for IPv6 ESP or 94 bytes for IPv6 AH.
11	Authentication failed: Bad signature.

RXE (9)

RXE is described in the legacy descriptor format in Section 7.1.5.

L4E (10)

L4 integrity error is valid only when the *L4I* bit in the *Status* field is set. It is active if L4 processing fails (TCP checksum or UDP checksum or SCTP CRC). Packets with L4 integrity error are posted to host memory regardless of the store bad packet setting (FCTRL.*SBP*). In case of VXLAN or NVGRE packets, this relates to the internal L4 header.

FCEOFe(11) / IPE(11)

This bit has multiplexed functionality. FCoE packets are indicated as such in the *Packet Type* field in the Rx descriptor.

Non-FCoE Packet	FCoE Packet
IPE (IPv4 checksum error) is described in Section 7.1.5. In case of VXLAN or NVGRE packets, this relates to the internal IP header.	FC EOF Exception (**FCEOFe**). This bit indicates unexpected EOF or SOF flags. The specific error is defined by the *FCEOF* bit in the extended status previously described.

PKT_LEN (16-bit offset 32, 2nd line)

PKT_LEN holds the number of bytes posted to the packet buffer. The length covers the data written to a receive buffer including posted CRC bytes (if any). Software must read multiple descriptors to determine the complete length for packets that span multiple receive buffers. If SRRCTL.*DESCTYPE* = 2 (advanced descriptor header splitting) and the buffer is not split because the header is bigger than the allocated header buffer, this field reflects the size of the data written to the data buffer (header + data).

When short packets are padded, the *PKT_LEN* does not include the padding, so that the software device driver can detect the location of a potential time stamp in the packet.

When a time stamp is added (*TSIP* bit is set), the *PKT_LEN* includes the timestamp size (+8).

VLAN Tag (16-bit offset 48, 2nd line)

This field is described in the legacy descriptor format in Section 7.1.5. The VLAN tag is set to 0x0000 even if a VLAN tag is present if the PFQDE.*HIDE_VLAN* bit is set for the queue.

7.1.5.3 Receive Descriptor Fetching

The X550 implements a fetch-by-demand mechanism for descriptor fetch. Descriptors are not fetched in advance, but rather fetched after a packet is received. Such a strategy eliminates the need to store descriptors on-die for each and every descriptor queue in anticipation for packet arrival.

7.1.5.4 Receive Descriptor Write-Back

The X550 writes back the receive descriptor immediately following the packet write into system memory. It is therefore possible for a single descriptor to be written at a time into memory. However, if aggregation occurs during descriptor fetch (see Section 7.1.5.3), the descriptors fetched in the aggregated operation are written back in a single write-back operation. In Receive Coalescing (RSC), all the descriptors except the last one are written back when they are completed. This does not have to be on packet boundaries but rather when the next descriptor of the same RSC is fetched. See Section 7.9.5.1 for more on RSC.

Note: Software can determine if a packet has been received only by checking the receive descriptor *DD* bit in memory. Checking through *DD* bits (and not by checking the receive head pointer in RDH/RDL registers) eliminates a potential race condition: all descriptor data is posted internally prior to incrementing the head register and a read of the head register could potentially pass the descriptor waiting inside the X550.

7.1.5.5 Receive Descriptor Queue Structure

Figure 7-11 shows the structure of each of the receive descriptor rings. Note that each ring uses a contiguous memory space.

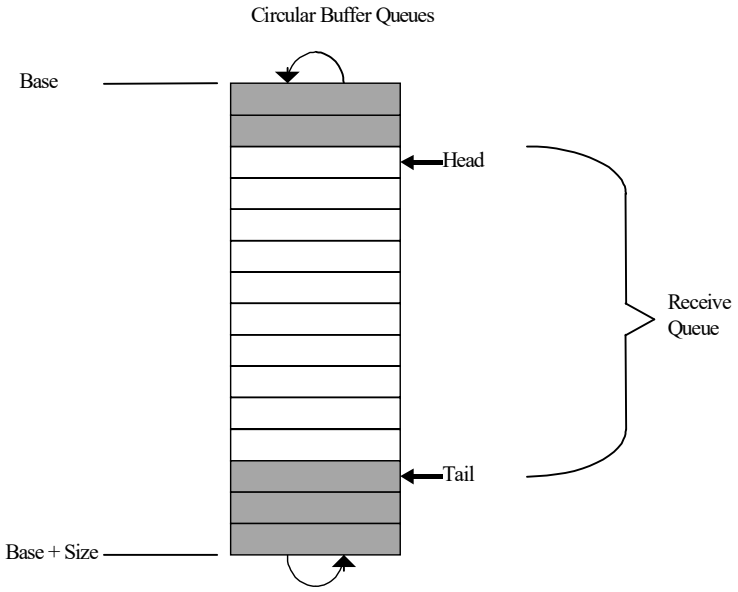


Figure 7-11. Receive Descriptor Ring Structure

Software inserts receive descriptors by advancing the tail pointer(s) to refer to the address of the entry just beyond the last valid descriptor. This is accomplished by writing the descriptor tail register(s) with the offset of the entry beyond the last valid descriptor. The X550 adjusts its internal tail pointer(s) accordingly. As packets arrive, they are stored in memory and the internal head pointer(s) is increased by the X550.

When RSC is not enabled, the visible (external) head pointer(s) reflect the internal ones. On any receive queue that enables RSC, updating the external head pointer might be delayed until interrupt assertion. When the head pointer(s) is equal to the tail pointer(s), the queue(s) is empty. The X550 stops storing packets in system memory until software advances the tail pointer(s), making more receive buffers available.

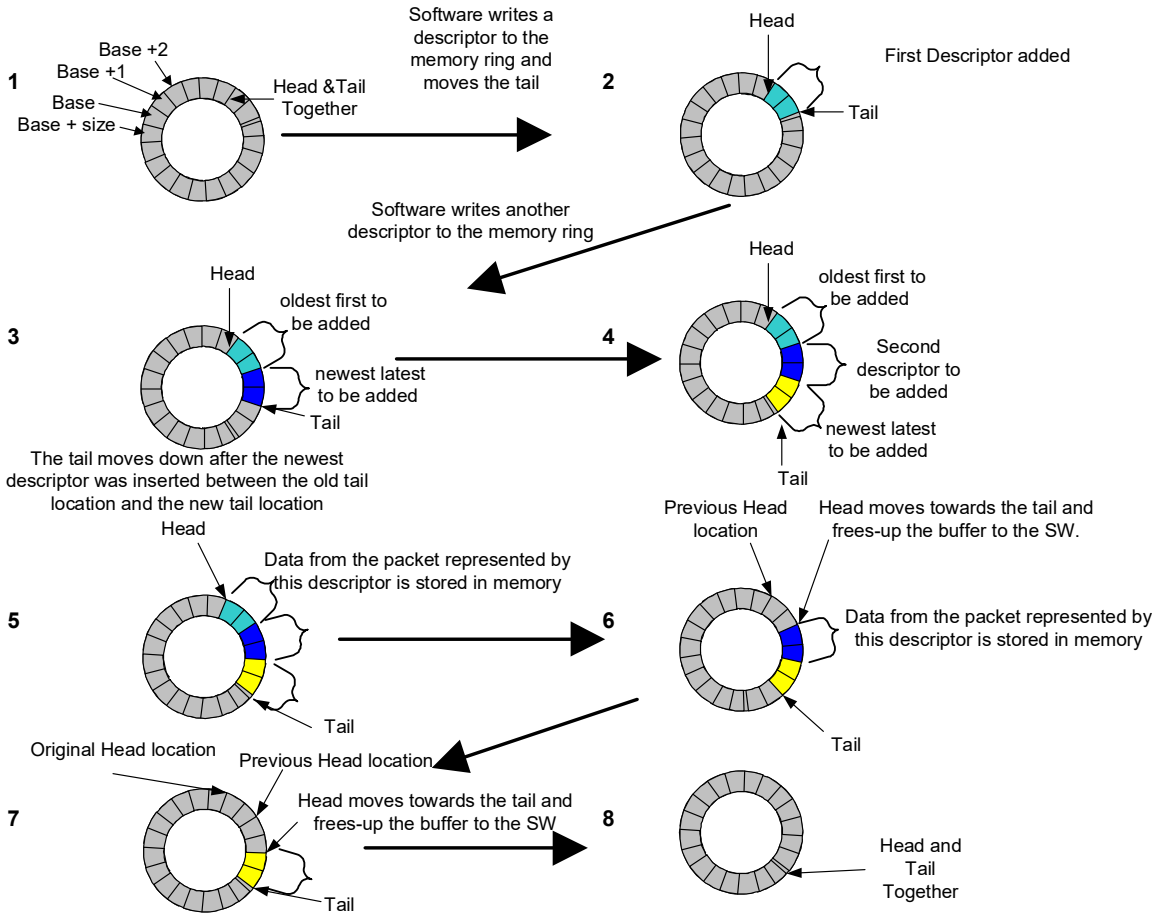


Figure 7-12. Descriptors and Memory Rings

The X550 writes back used descriptors just prior to advancing the head pointer(s). Head and tail pointers wrap back to base when the number of descriptors corresponding to the size of the descriptor ring have been processed.

The receive descriptor head and tail pointers reference to 16-byte blocks of memory. Shaded boxes in Figure 7-12 represent descriptors that have stored incoming packets but have not yet been recognized by software. Software can determine if a receive buffer is valid by reading descriptors in memory rather than by I/O reads. Any descriptor with a *DD* bit set has been used by the hardware, and is ready to be processed by software.

Note: The head pointer points to the next descriptor that is to be written back. At the completion of the descriptor write-back operation, this pointer is increased by the number of descriptors written back. Hardware owns all descriptors between [head... tail]. Any descriptor not in this range is owned by software.

The receive descriptor rings are described by the following registers:

- **Receive Descriptor Base Address registers (RDBA)** — This register indicates the start of the descriptor ring buffer; this 64-bit address is aligned on a 128-byte boundary and is stored in two consecutive 32-bit registers. Hardware ignores the lower 7 bits.
- **Receive Descriptor Length registers (RDLEN)** — This register determines the number of bytes allocated to the circular buffer. This value must be a multiple of 128 (the maximum cache line size). Since each descriptor is 16 bytes in length, the total number of receive descriptors is always a multiple of 8.
- **Receive Descriptor Head registers (RDH)** — This register holds a value that is an offset from the base, and indicates the in-progress descriptor. There can be up to 64K-8 descriptors in the circular buffer. Hardware maintains a shadow copy that includes those descriptors completed but not yet stored in memory.

  Software can determine if a packet has been received by either of two methods: reading the *DD* bit in the receive descriptor field or by performing a Programmed I/O read of the Receive Descriptor Head register. Checking the descriptor *DD* bit in memory eliminates a potential race condition. All descriptor data is written to the I/O bus prior to incrementing the head register, but a read of the head register could pass the data write in systems performing I/O write buffering. Updates to receive descriptors use the same I/O write path and follow all data writes. Consequently, they are not subject to the race.
- **Receive Descriptor Tail registers (RDT)** — This register holds a value that is an offset from the base, and identifies the location beyond the last descriptor hardware can process. This is the location where software writes the first new descriptor.

If software statically allocates buffers, and uses a memory read to check for completed descriptors, it simply has to zero the status byte in the descriptor to make it ready for re-use by hardware. This is not a hardware requirement, but is necessary for performing an in-memory scan. This is relevant only to legacy descriptors.

All the registers controlling the descriptor rings behavior should be set before receive is enabled, apart from the tail registers which are used during the regular flow of data.

7.1.5.5.1 Low Receive Descriptors Threshold

As previously described, the size of the receive queues is measured by the number of receive descriptors. During run time, software processes descriptors and upon completion of descriptors, increments the Receive Descriptor Tail registers. At the same time, the hardware may post new received packets incrementing the Receive Descriptor Head registers for each used descriptor.

The number of usable (free) descriptors for the hardware is the distance between the Tail and Head registers. When the tail reaches the head, there are no free descriptors and further packets might be either dropped or block the receive FIFO. To avoid this situation, the X550 might generate a low latency interrupt (associated to the relevant Rx queue) once there are less equal free descriptors than specified by a low level threshold. The threshold is defined in 64 descriptors granularity per queue in the SRRCTL[n].*RDMTS* field.

7.1.6 Receive Offloads

7.1.6.1 Header Splitting

7.1.6.1.1 Purpose

This feature consists of splitting a packet header to a different memory space. This helps the host to fetch headers only for processing: headers are posted through a regular snoop transaction to be processed by the host CPU. It is recommended to perform this transaction with TPH enabled (see Section 7.5).

The X550's support for header split is controlled by the *DESCTYPE* field of the Split Receive Control registers (SRRCTL). The following modes exist in both split and non-split modes:

- 000b: Legacy mode — Legacy descriptors are used, headers and payloads are not split.
- 001b: Advanced mode, no split — Advanced descriptors are in use, header and payload are not split.
- 010b: Advanced mode, header split — Advanced descriptors are in use, header and payload are split to different buffers.
- 101b: Advanced mode, split always — Advanced descriptors are in use, header and payload are split to different buffers. If no split is done, the first part of the packet is stored in the header buffer.

The X550 uses packet splitting when the SRRCTL[n].*DESCTYPE* is greater than one.

7.1.6.1.2 Description

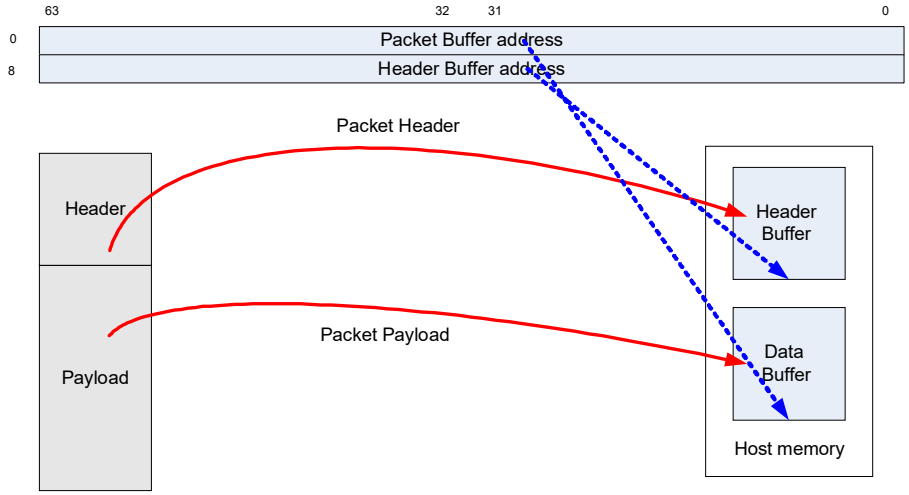


Figure 7-13. Header Splitting Diagram

The physical address of each buffer is written in the *Buffer Addresses* fields:

- The packet buffer address includes the address of the buffer assigned to the packet data.
- The header buffer address includes the address of the buffer that contains the header information. The receive DMA module stores the header portion of the received packets into this buffer.

The sizes of these buffers are statically defined in the SRRCTL[n] registers:

- The *BSIZEPACKET* field defines the size of the buffer for the received packet.
- The *BSIZEHEADER* field defines the size of the buffer for the received header. If header split is enabled, this field must be configured to a non-zero value. The X550 only writes the header portion into the header buffer. The header size is determined by the options enabled in the PSRTYPE registers.

When header split is selected, the packet is split only on selected types of packets. A bit exists for each option in PSRTYPE[n] registers, so several options can be used in conjunction. If one or more bits are set, the splitting is performed for the corresponding packet type. See Section 8.2.2.8.17 for details on the possible header types supported. In virtualization mode, a separate PSRTYPE register is provided per pool up to the number of pools enabled. In non-virtualization mode, only PSRTYPE[0] is used.

Rules regarding header split:

- Packets that have headers bigger than 1023 bytes are not split.
- The header of a fragmented IPv6 packet is defined until the *Fragment* extension header.
- Header split must not be used in a queue used for a FCoE large receive.
- An IP in IP packet (such as any combination of IPv4 and IPv6 tunneling) is not split. Not relevant for NVGRE and VXLAN packets
- Packet header cannot span across buffers, therefore, the size of the header buffer must be larger than any expected header size. In case of header split mode (SRRCTL.*DESCTYPE* = 010b), a packet with a header larger than the header buffer is not split.
- If an IPsec header is present in the receive packet, the following rules apply:
  - IPsec packets handled in the X550 always include IPsec header in a split done at IP boundary.
  - IPsec packets handled in software must never do header split.

Table 7-25 lists the behavior of the X550 in the different modes.

Table 7-25. Behavior in Header Split Modes

DESCTYPE	Condition	SPH	HBO[1]	PKT_LEN[2]	HDR_LEN[2]	Header and Payload DMA[3]
Split	1. Header can't be decoded	0	0	Min (packet length, buffer size)	0x0	Header + Payload (+ TimeStamp) (+ Padding) -> Packet Buffer
	2. Header <= BSIZEHEADER and Payload > 0	1	0	Min (payload length, buffer size)	Header size	Header -> Header Buffer Payload (+ TimeStamp) (+ Padding) -> Packet Buffer
	3. Header <= BSIZEHEADER and Payload = 0 (Header only packet)	1	0	0	Header size	Header (+ TimeStamp) (+ Padding) -> Header Buffer[4]
	4. Header > BSIZEHEADER	1	1	Min (packet length, buffer size)	Header size[5]	Header + Payload (+ TimeStamp) -> Packet Buffer

Table 7-25. Behavior in Header Split Modes [continued]

DESCTYPE	Condition	SPH	HBO[1]	PKT_LEN[2]	HDR_LEN[2]	Header and Payload DMA[3]
Split – always use header buffer	1. Header can't be decoded and packet length <= BSIZEHEADER	0	0	0x0	Packet length	Header + Payload (+ TimeStamp) (+ Padding) -> Header Buffer
	2. Header cannot be decoded and packet length > BSIZEHEADER	0	0	Min (packet length – BSIZEHEADER, data buffer size)	BSIZEHEADER	Header + Payload (+ TimeStamp) -> Header + Packet Buffers[5]
	3. Header <= BSIZEHEADER	1	0	Min (payload length, data buffer size)	Header Size	Header -> Header Buffer Payload (+ TimeStamp) (+ Padding) -> Packet Buffer
	4. Header > BSIZEHEADER	1	1	Min (packet length – BSIZEHEADER, data buffer size)	Header Size[6]	Header + Payload (+ TimeStamp) -> Header + Packet Buffer

1. HBO is set to 1b if the header size is bigger than *BSIZEHEADER* and zero otherwise.
2. PKT_LEN and HDR_LEN includes optionally added time stamp if *TSIP* bit is set and does not include padding added by the device.
3. Partial means up to *BSIZEHEADER*.
4. Even if there is no payload at all and there is a timestamp in the packet - the timestamp is considered as an 8 bytes payload and is written to the packet buffer.
5. If the packet spans more than one descriptor, only the header buffer of the first descriptor is used.
6. HDR_LEN does not reflect the actual data size stored in the header buffer. It reflects the header size determined by the parser.

7.1.6.2 Receive Packet Timestamp in Buffer

The X550 supports adding an optional tailored header appended to the end of the packet in the receive buffer. The tailored header includes a 64-bit timestamp composed of the packet reception time measured in the SYSTIMEL (Low DW) and SYSTIMEH (High DW) registers (See Section 7.7.3.2 for further information on SYSTIMEL/H operation). The timestamp starts right at the end of the packet (after the last byte). It is sent as {SYSTIMEH, SYSTEML} - most significant byte first (closest to packet).

PKT_LEN and *HDR_LEN* includes optionally added time stamp.

When the TSAUXC.*DISABLE_SYSTIME* bit is cleared and the TSYNCRXCTL.*TSIP_UP_EN* is set for the UP in the packet (or TSYNCRXCTL.*TSIP_UT_EN* for un-tagged packets), packets received that meet the TSYNCRXCTL.*TYPE* are time-stamped. A packet that was time stamped is reported as follows:

- Place a 64-bit timestamp, indicating the time a packet was received by the MAC, appended at the end of the received packet within the receive buffer.
- Set the *TSIP* bit in the RDESC.*STATUS* field of the last receive descriptor.

Notes: Even if TSYNCRXCTL.*TYPE*=100b (timestamp all packets), FCoE DDP packets do not have a timestamp.

When packets are coalesced, the timestamp reflects the reception time of the last coalesced fragment, unless the packet is a pure ACK packet, in which case, the timestamp is of the first packet.

7.1.6.3 Receive Checksum Offloading

The X550 supports the offloading of three receive checksum calculations: the fragment checksum, the IPv4 header checksum, and the TCP/UDP checksum.

For supported packet/frame types, the entire checksum calculation can be offloaded to the X550. The X550 calculates the IPv4 checksum and indicates a pass/fail indication to software via the *IPv4 Checksum Error* bit (RDESC.*IPE*) in the *ERROR* field of the receive descriptor. Similarly, the X550 calculates the TCP or UDP checksum and indicates a pass/fail condition to software via the *TCP/UDP Checksum Error* bit (RDESC.*TCPE*). For NVGRE or VXLAN packets, the *IPE* and *TCPE* bits relates to the inner IP/TCP header. The outer IPv4 checksum is also checked and a pass/fail indication is indicated to software via the *Outer IPv4 Checksum Error* bit (*OUTERIPER*). These error bits are valid when the respective status bits indicate the checksum was calculated for the packet (RDESC.*IPCS*, RDESC.*L4CS*, and RDESC.*OUTERIPCS* respectively).

Similarly, if RFCTL.*IPV6_DIS* and RFCTL.*IP6XSUM_DIS* are cleared to zero, the X550 calculates the TCP or UDP checksum for IPv6 packets. It then indicates a pass/fail condition in the *TCP/UDP Checksum Error* bit (RDESC.*TCPE*).

Note: Only Ethernet II frames are supported (no checksum support for SNAP packets).

Table 7-26. Supported Receive Checksum Capabilities

Packet Type	Hardware IP Checksum Calculation	Hardware TCP/UDP Checksum Calculation
IP header's protocol field contains a protocol # other than TCP or UDP.	Yes	No
IPv4 + TCP/UDP packets.	Yes	Yes
IPv6 + TCP/UDP packets.	No (N/A)	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes).	Yes	Yes
IPv6 packet with next header options: • Hop-by-hop options. • Destinations options (without home address). • Destinations options (with home address). • Routing (with segment left 0). • Routing (with segment left > 0). • Fragment.	 No (N/A) No (N/A) No (N/A) No (N/A) No (N/A) No (N/A)	 Yes Yes No Yes No No
Packet has TCP or UDP options.	Yes	Yes
IPv4 tunnels: • IPv4 packet in an IPv4 tunnel. • IPv6 packet in an IPv4 tunnel.	 Yes (outer IPv4 only) Yes (IPv4)	 No No
IPv6 tunnels: • IPv4 packet in an IPv6 tunnel. • IPv6 packet in an IPv6 tunnel.	 No No (N/A)	 No No
Packet is an IPv4 fragment.	Yes	UDP checksum assist
Packet is greater than 1522 bytes.	Yes	Yes
Packet has 802.3ac tag.	Yes	Yes

Table 7-26. Supported Receive Checksum Capabilities [continued]

Packet Type	Hardware IP Checksum Calculation	Hardware TCP/UDP Checksum Calculation
NVGRE: • Inner IPv4 packet • Inner IPv6 packet	Yes (Inner and outer) Yes (outer)	Yes Yes
VXLAN: • Inner IPv4 packet • Inner IPv6 packet	Yes (Inner and outer) Yes (outer)	Yes (inner only)[1] Yes (inner only)

1. The outer UDP header of VXLAN packets do not use a checksum.

7.1.6.4 SCTP Receive Offload

If a receive packet is identified as SCTP, the X550 checks the CRC32 checksum of this packet and identifies this packet as SCTP. Software is notified of the CRC check via the *L4I* and *L4E* bits in the *Extended Status* field and *Extended Error* field in the Rx descriptor. The detection of an SCTP packet is indicated via the *SCTP* bit in the *Packet Type* field of the Rx descriptor. SCTP CRC uses the CRC32c polynomial as follows (0x11EDC6F41):

$$X_{32}+X_{28}+X_{27}+X_{26}+X_{25}+X_{23}+X_{22}+X_{20}+X_{19}+X_{18}+X_{14}+X_{13}+X_{11}+X_{10}+X_{9}+X_{8}+X_{6}+X_{0}$$

The checker assumes the following SCTP packet format.

Table 7-27. SCTP Header

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Source Port																Destination Port															
Verification Tag																															
CRC Checksum (CRC32c)																															
Chunks 1..n																															

7.1.6.5 Receive UDP Fragmentation Checksum

The X550 might provide a receive fragmented UDP checksum offload for IPv4 non-tunneled packets. The RXCSUM.*PCSD* bit should be cleared and the RXCSUM.*IPPCSE* bit should be set to enable this mode.

The following table lists the outcome descriptor fields for the following incoming packets types.

Incoming Packet Type	Fragment Checksum	UDPV	UDPCS / L4CS
Non-IP packet	0	0	0
IPv6 packet Non fragmented IPv4 packet	0	0	Depends on transport UDP: 1 / 1 TCP: 0 / 1
Fragmented IPv4 with protocol = UDP, first fragment (UDP protocol present)	The unadjusted 1's complement checksum of the IP payload if Checksum in packet header is different than zero and zero otherwise.	1 if the UDP header checksum is valid (not 0)	0 / 0
Fragmented IPv4, when not first fragment	The unadjusted 1's complement checksum of the IP payload	1	0 / 0

Incoming Packet Type	Fragment Checksum	UDPV	UDPCS / L4CS
Fragmented IPv4 with protocol = UDP, first fragment (UDP protocol present) within a VXLAN or NVGRE packet.	The unadjusted 1's complement checksum of the inner IP payload	1 if the UDP header checksum is valid (not 0)	1 / 0
Fragmented IPv4 when not first fragment within a VXLAN or NVGRE packet.	The unadjusted 1's complement checksum of the inner IP payload	0	1 / 0

Note: When the driver computes the 16-bit ones complement sum on the incoming packets of the UDP fragments, it should expect a value of 0xFFFF.

7.1.7 Receive Statistics

7.1.7.1 General Rules

- All Statistics registers are cleared on read. In addition, they stick at 0xFF...F when the maximum value is reached.
- For the receive statistics it should be noted that a packet is indicated as received if it passes the device filters and is placed into the packet buffer memory. A packet does not have to be DMA'ed to host memory to be counted as received.
- Due to divergent paths between interrupt-generation and logging of relevant statistics counts, it might be possible to generate an interrupt to the system for a noteworthy event prior to the associated statistics count actually being increased. This is extremely unlikely due to expected delays associated with the system interrupt-collection and ISR delay, but might be an explanation for interrupt statistics values that do not quite make sense. Hardware guarantees that any event noteworthy of inclusion in a statistics count is reflected in the appropriate count within 1 μs; a small time-delay prior to reading the statistics might be required to avoid a potential mismatch between and interrupt and its cause.
- If RSC is enabled, statistics are collected before RSC is applied to the packets.
- All byte (octet) counters composed of 2 registers can be fetched by two consecutive 32-bit accesses while reading the Low 32-bit register first or a single 64-bit access.
- All receive statistic counters count the packets and bytes before coalescing by the RSC logic or FCoE DDP logic.
- All receive statistic counters in the Filter unit (listed below) might count packets that might be dropped by the packet buffer or receive DMA. Same comment is valid for the byte counters associated with these packet counters: PRC64, PRC127, PRC255, PRC511, PRC1023, PRC1522, BPRC, MPRC, GPRC, RXNFGPC, RUC, ROC.

7.1.7.2 Receive Statistics Hierarchy

The following diagram describes the relations between the packet flow and the different statistic counters.

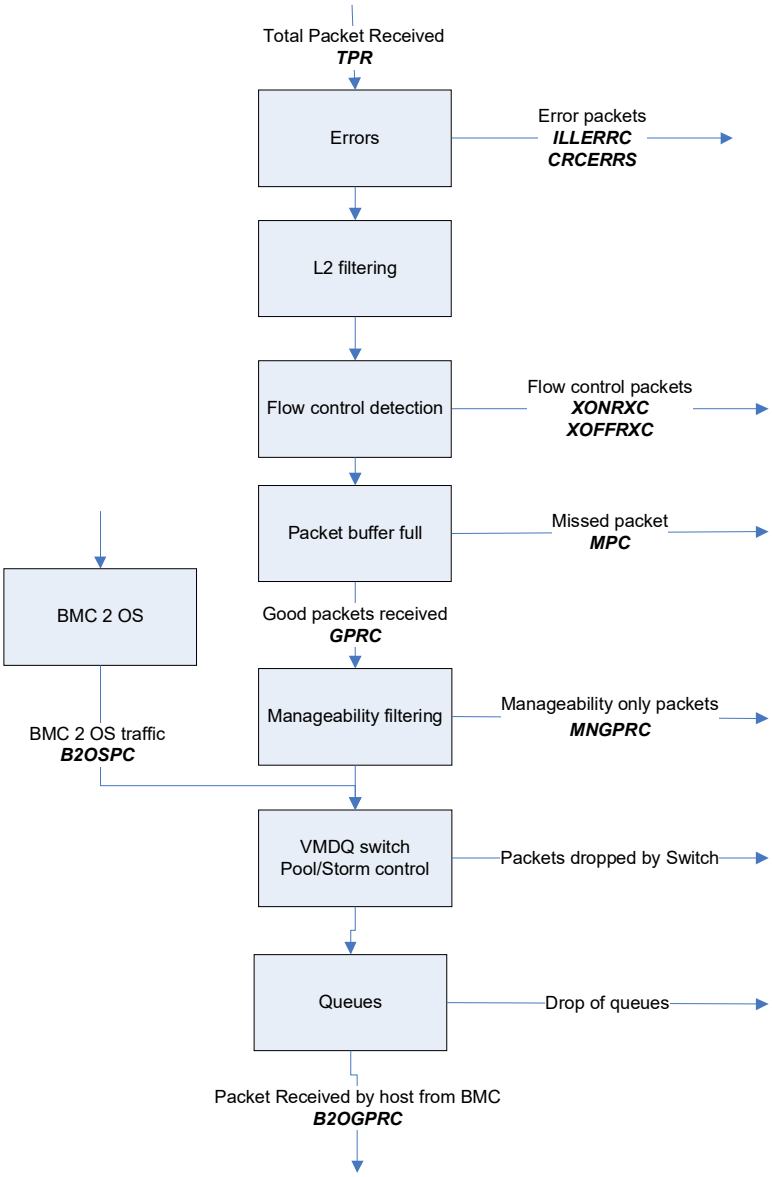


Figure 7-14. Receive Flow Statistics

7.2 Transmit Functionality

7.2.1 Packet Transmission

Transmit packets are made up of data buffers in host memory that are indicated to hardware by pointer and length pairs. These pointer and length pairs are named as transmit descriptors that are stored in host memory as well.

Software prepares memory structures for transmission by assembling a list of descriptors. It then indicates this list to hardware for updating the on-chip transmit tail pointer. Hardware transmits the packet only after it has completely fetched all packet data from host memory and deposited it into the on-chip transmit FIFO. This store and forward scheme enables hardware-based offloads such as TCP or UDP checksum computation, and many other ones detailed in this document while avoiding any potential PCIe under-runs.

7.2.1.1 Transmit Storage in System Memory

A packet (or multiple packets in transmit segmentation) can be composed of one or multiple buffers. Each buffer is indicated by a descriptor. Descriptors of a single packet are consecutive, while the first one points to the first buffer and the last one points to the last buffer (see Figure 7-15). The following rules must be kept:

- Address alignment of the data buffers can be on any byte boundary.
- Data buffers of any transmitted packet must include at least the 12 bytes of the source and destination Ethernet MAC Addresses as well as the 2 bytes of the *Type/Len* field.
- A packet (or multiple packets in transmit segmentation) can span any number of buffers (and their descriptors) up to a limit of 40 minus *WTHRESH* minus 2 (see Section 7.2.3.3 for Tx Ring details and Section 7.2.3.5.1 for *WTHRESH* details). For best performance it is recommended to minimize the number of buffers as possible.

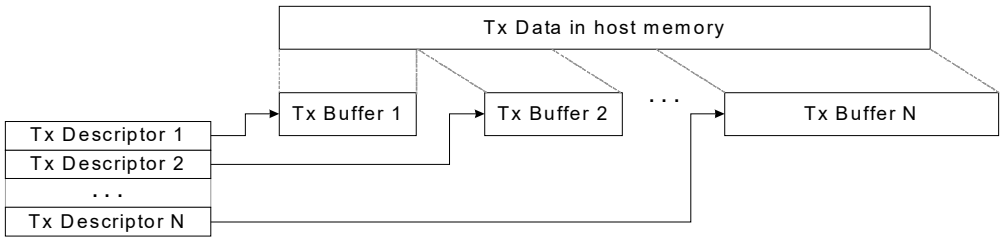


Figure 7-15. Tx Packet in Host Memory

7.2.1.2 Transmit Path in the X550

The transmit path in the X550 consists of the following stages:

- Descriptor plane
  - The X550 maintains a set of 128 on-die descriptor queues. Each queue is associated with a single descriptor ring in system memory. See Section 7.2.3.3 for more details on the Tx descriptor rings. Each on-die descriptor queue contains up to 40 descriptors to achieve the desired performance.
  - A fetch mechanism loads Tx descriptors from the descriptor rings in system memory to the respective descriptor queues in the X550. A descriptor fetch arbiter determines the order in which descriptors are fetched into the various on-die descriptor queues. See Section 7.2.3.4 for more details on the fetch mechanism.
  - An arbitration scheme determines the order in which descriptors are processed and requests are generated for data reads. These requests load packet data from system memory into a set of Tx packet buffers. The arbitration mechanism varies with configuration and is described in Section 7.7.7.
  - Once a packet has been fetched into a packet buffer, status is (optionally) written back into system memory. See Section 7.2.3.5 for more details.
- Packet plane (data plane)
  - Packet data is stored in up to eight packet buffers. The number and size of packet buffers vary with the mode of operation and is described in Section 7.2.1.2.2.
  - If more than a single packet buffer is enabled, an arbitration scheme determines the order in which packets are taken out of the packet buffers and sent to the MAC for transmission.The arbitration mechanism is described in Section 7.7.7.

7.2.1.2.1 Tx Queues Assignment

The X550 supports a total of 128 queues per LAN port. Each Tx queue is associated with a packet buffer and the association varies with the operational mode. The following mechanisms impact the association of the Tx queues. These are described briefly in this section, and in full details in separate sections:

- **Virtualization (VT)** — In a virtualized environment, DMA resources are shared between more than one software entity (operating system and/or device driver). This is done through allocation of transmit descriptor queues to virtual partitions (VMM, IOVM, VMs, or VFs). Allocation of queues to virtual partitions is done in sets of queues of the same size, called queue pools, or **pools**. A pool is associated with a single virtual partition. Different queues in a pool can be associated with different packet buffers. For example, in a DCB system, each of the queues in a pool might belong to a different TC and therefore to a different packet buffer. The PFVFTE register contains a bit per VF. When the bit is set to 0b, packet transmission from the VF is disabled. When set to 1b, packet transmission from the VF is enabled.
- **DCB** — DCB provides QoS through priority queues, priority flow control, and congestion management. Queues are classified into one of several (up to eight) Traffic Classes (TCs). Each TC is associated with a single unique packet buffer.
- **Transmit fanout** — A single descriptor queue might be enough for a given functionality. For example, in a VT system, a single Tx queue can be allocated per VM. However, it is often the case that the data rate achieved through a single buffer is limited. This is especially true with 10 GbE, and traffic needs to be divided into several Tx queues to reach the desired data rate. Therefore, multiple queues might be provided for the same functionality.

Table 7-28 lists the queuing schemes. Scheme selection is done via the MTQC register.

Table 7-28. Tx Queuing Schemes

VT	DCB	Queues Allocation	Packet Buffers Allocation
No	No	A single set of 64 queues is assigned to a single packet buffer. Queues 64…127 should not be used.	A single packet buffer for all traffic
No	Yes	Eight TCs mode – allocation of 32-32-16-16-8-8-8-8 queues for TC0-TC1-…- TC7, respectively. Four TCs mode — allocation of 64-32-16-16 queues for TC0-TC1-…- TC3, respectively.	A separate packet buffer is allocated to each TC (total of four or eight).
Yes	No	32 pools x 4 queues, or 64 pools x 2 queues	A single packet buffer for all traffic.
Yes	Yes	16 pools x 8 TCs, or 32 pools x 4 TCs	A separate packet buffer is allocated to each TC (total of four or eight).

Note: Software can use any number of queues per each TC or per each pool within the allocated ranges previously described by disabling any unused queue.

Note: Programming MTQC must be done only during the init phase, while software must also set RTTDCS.*ARBDIS* before configuring MTQC and then clear RTTDCS.*ARBDIS* afterwards.

Table 7-29. Queues, Pools, and TCs programming

Device Setting (MTQC)			Device Functionality	
DCB_ena	VT_Ena	NUM_TC_OR_Q	Total Number of Tx Queues	TC & Pools
0	0	00	64	-
<> 0	<> 0	00	Non valid configuration	
0	0	<> 00	Non valid configuration	
1	0	01	N/A	
1	0	10	128	TC0 — TC3
1	0	11	128	TC0 — TC7
0	1	01	128	64 pools
0	1	10	128	32 pools
0	1	11	N/A	
1	1	01	N/A	
1	1	10	128	TC0 — TC3 & 32 pools
1	1	11	128	TC0 — TC7 & 16 pools

Allocating descriptor queues to VFs uses a consistent indexing over the different Tx queuing schemes. The most significant bits of the queue index represent the VF index, and the least significant bits are either the TC index or are used by software to dispatch traffic according to a Transmit Side Scaling (TSS) algorithm — similar to RSS in the Rx path.

The Tx queue numbers associated with the TCs are listed in the following tables: Table 7-30 and Table 7-31.

Table 7-30. Tx Queues Indexing When VT-on

VT Mode	Allocation of Queue Index Bits According to						
	6	5	4	3	2	1	0
64 VFs + TSS	VF or pool (63..0)						TSS
32 VFs + TSS *or* 4 TCs	VF or pool (31..0)					TSS / TC	
16 VFs + 8 TCs	VF (15..0)				TC		

Table 7-31. Tx Queues Indexing When VT-off and DCB-on

TC Mode	TCn	# of Qs	Queues Attached to TCn
4 TCs	TC0	64	0xxxxxx
	TC1	32	10xxxxx
	TC2	16	110xxxx
	TC3	16	111xxxx
8 TCs	TC0	32	00xxxxx
	TC1	32	01xxxxx
	TC2	16	100xxxx
	TC3	16	101xxxx
	TC4	8	1100xxx
	TC5	8	1101xxx
	TC6	8	1110xxx
	TC7	8	1111xxx

Note: "x" refers to both 0 or 1, and is used by software to dispatch Tx flows via TSS algorithm.

7.2.1.2.2 Tx Packet Buffers

As previously described, the following modes exist for the X550 packet buffers:

- A single 160 KB packet buffer that serves all Tx descriptor queues, leading to one single (or no) TC enabled, TC0
- Four 40 KB packet buffers, one per enabled TC, leading to four TCs, TC0 to TC3
- Eight 20 KB packet buffers, one per enabled TC, leading to eight TCs, TC0 to TC7

The size of the Tx packet buffer(s) is programmed via the TXPBSIZE registers, one register per TC. Null-sized packet buffer corresponds to a disabled TC.

Note: Setting the packet buffers' size leads to a different partition of a shared internal memory and must be done during boot, prior to communicating, and followed by a software reset.

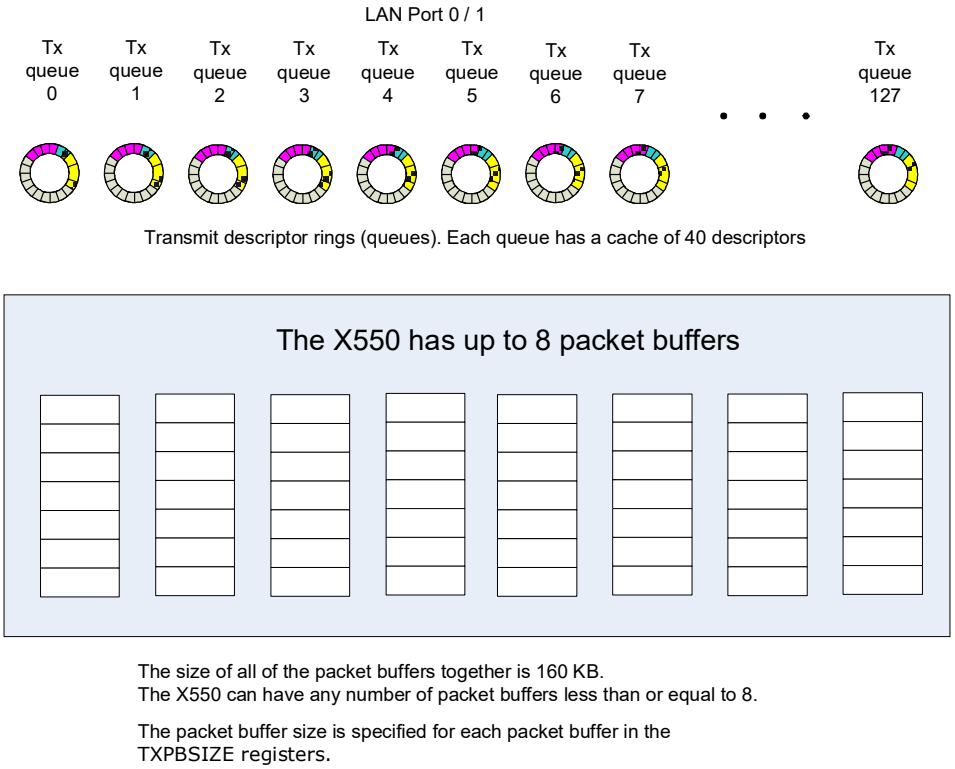


Figure 7-16. Tx Packet Buffers

7.2.1.2.3 Tx Arbitration Schemes

There are basically four Tx arbitration schemes, one per each combination of the DCB and Virtualization (VT) enabled/disabled modes. They are configured via the MTQC.*MTQE* register field.

7.2.1.2.3.1 DCB-on/VT-on

When both DCB and virtualization are enabled, queues are allocated to the packet buffers in a fixed manner, the same number of queues per each TC. Two DCB modes are supported, four TCs or eight TCs mode, according to coherent configuration made in registers TXPBSIZE and MTQC.

Descriptor Plane Arbiters and Schedulers:

- **Rate-Scheduler** — Once a frame has been fetched out from a rate-limited queue, the next time another frame could be fetched from that queue is regulated by the rate-scheduler. In the meantime, the queue is considered as if it was empty (such as switched-off) for the subsequent arbitration layers.
- **VM Weighted Round Robin Arbiter** — Descriptors are fetched out from queues attached to the same TC in a frame-by-frame weighted round-robin manner, while taking into account any limitation as previously described. Weights or credits allocated to each queue are configured via the RTTDT1C register. Bandwidth unused by one queue is reallocated to the other queues within the TC, proportionally to their relative bandwidth shares. TC bandwidth limitation is distributed across all the queues attached to the TC, proportionally to their relative bandwidth shares. Details on

weighted round-robin arbiter between the queues can be found in Section 7.6.2.3. It is assumed traffic is dispatched across the queues attached to a same TC in a straightforward manner, according to the VF to which it belongs.

- **TC Weighted Strict Priority Arbiter** — Descriptors are fetched out from queues attached to different TCs in a frame-by-frame weighted strict-priority manner. Weights or credits allocated to each TC are configured via RTTDT2C registers. Bandwidth unused by one TC is reallocated to the others, proportionally to their relative bandwidth shares. Link bandwidth limitation is distributed across all the TCs, proportionally to their relative bandwidth shares. Details on weighted strict-priority arbiter between the TCs can be found at Section 7.6.2.3. It is assumed (each) driver dispatches traffic across the TCs according to the 802.1p *User Priority* field inserted by the operating system and according to a user priority-to-TC Tx mapping table.

Packet Plane Arbiters:

- **TC Weighted Strict Priority Arbiter** — Packets are fetched out from the different packet buffers in a frame-by-frame weighted strict-priority manner. Weights or credits allocated to each TC (such as to each packet buffer) are configured via RTTPT2C registers, with the same allocation done at the descriptor plane. Bandwidth unused by one TC and link bandwidth limitation is distributed over the TCs as in the descriptor plane. Details on weighted strict-priority arbiter between the TCs can be found in Section 7.6.2.3.
- **Priority Flow Control** packets are inserted with strict priority over any other packets.
- **Manageability** packets are inserted with strict priority over data packets from the same TC, with respect to the bandwidth allocated to the concerned TC. TCs that belong to manageability packets are controlled by MNGTXMAP.*MAP*.

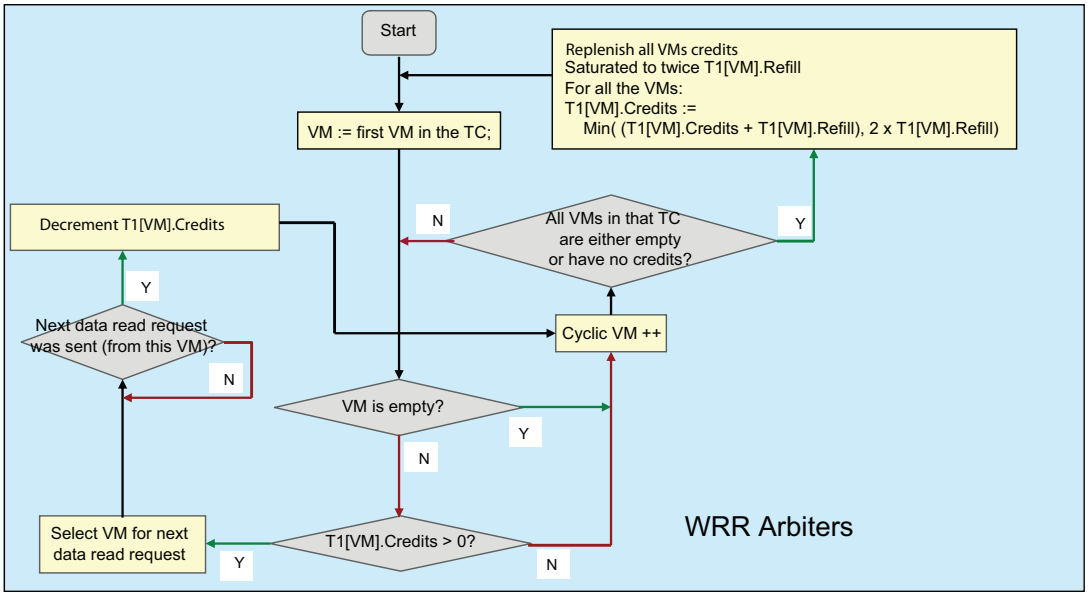


Figure 7-17. Transmit Architecture DCB-on/VT-on — Eight TCs Mode

Note: Replicating TC arbiters before and after the packet buffers is required to provide arbitration whether PCI bandwidth is smaller or greater than the link bandwidth, respectively.

7.2.1.2.3.2 DCB-on/VT-off

When DCB is enabled and virtualization disabled, queues are allocated to the packet buffers in a fixed manner according to the number of TCs. Two DCB modes are supported, four TCs or eight TCs mode, according to coherent configuration made in registers TXPBSIZE and MTQC. In Figure 7-18, eight TCs mode is shown.

- The unique difference with the DCB-on/VT-on arbitration scheme previously described is that the VM weighted round-robin arbiters are degenerated into simple frame-by-frame round-robin arbiters across the queues attached to the same TC. It is assumed driver dispatches traffic across the queues attached to a same TC according to hashing performed on MAC destination addresses. This is aimed to minimize crosstalk between rate-limited and non-rate-limited flows.

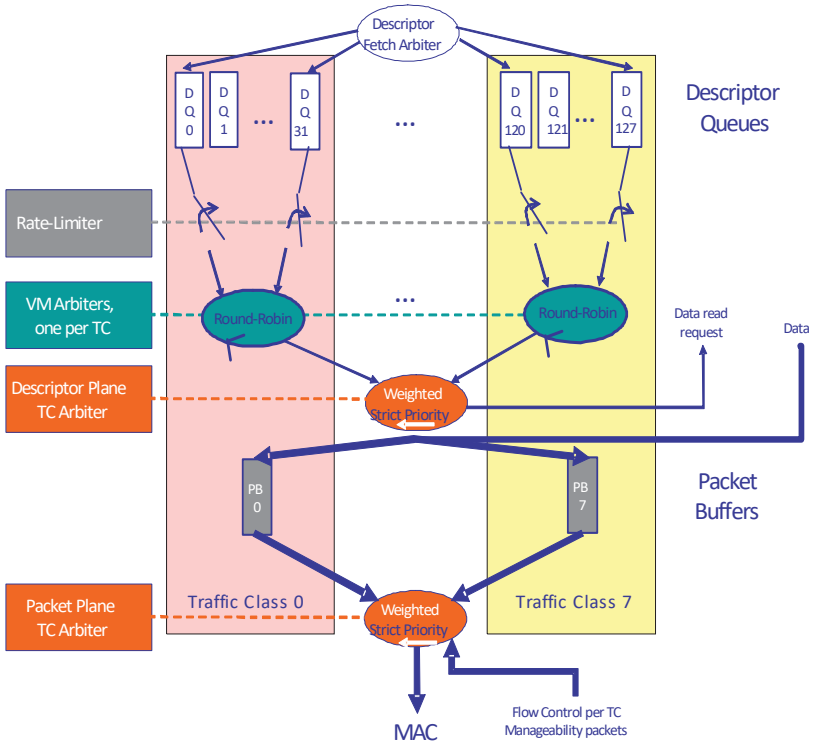


Figure 7-18. Transmit Architecture DCB-on/VT-off — Eight TCs Mode

7.2.1.2.3.3 DCB-off/VT-on

When DCB is disabled and virtualization enabled, all the 128 queues are allocated to a single packet buffer PB(0). Queues are grouped into 32 or 64 pools of 4 or 2 queues, respectively. The number of queue pools corresponds to the number of VFs exposed. Queues are attached to pools according to consecutive indexes

— For the 32 pools case, queues 0, 1, 2, 3 are attached to VF0, queues 4, 5, 6, 7 are attached to VF1, and so forth up to VF31.

— For the 64 pools case, queues 0 and 1 are attached to VF0, queues 2 and 3 are attached to VF1, and so forth up to VF63.

Descriptor Plane Arbiters:

- **Descriptor Queues Round Robin Arbiter** — Descriptors are fetched out from the internal descriptor queues attached to the same pool in a frame-by-frame round-robin manner. It is assumed driver dispatches traffic across the queues of a same pool according to some Transmit Side Scaling (TSS) algorithm similarly to what is done by hardware in the Rx path with RSS.
- **VM Weighted Round Robin Arbiter** — Descriptors are fetched out from queues attached to different pools in a frame-by-frame weighted round-robin manner. Weights or credits allocated to a pool are those allocated for the lowest queue of the pool via the RTTDT1C register. Bandwidth unused by one pool is reallocated to the others proportionally to their relative bandwidth shares. Link bandwidth limitation is distributed across all the pools, proportionally to their relative bandwidth shares. Details on weighted round-robin arbiter between the pools can be found in Section 7.6.2.3.

Packet Plane Arbiter:

- **Manageability** packets are inserted with strict priority over data packets.

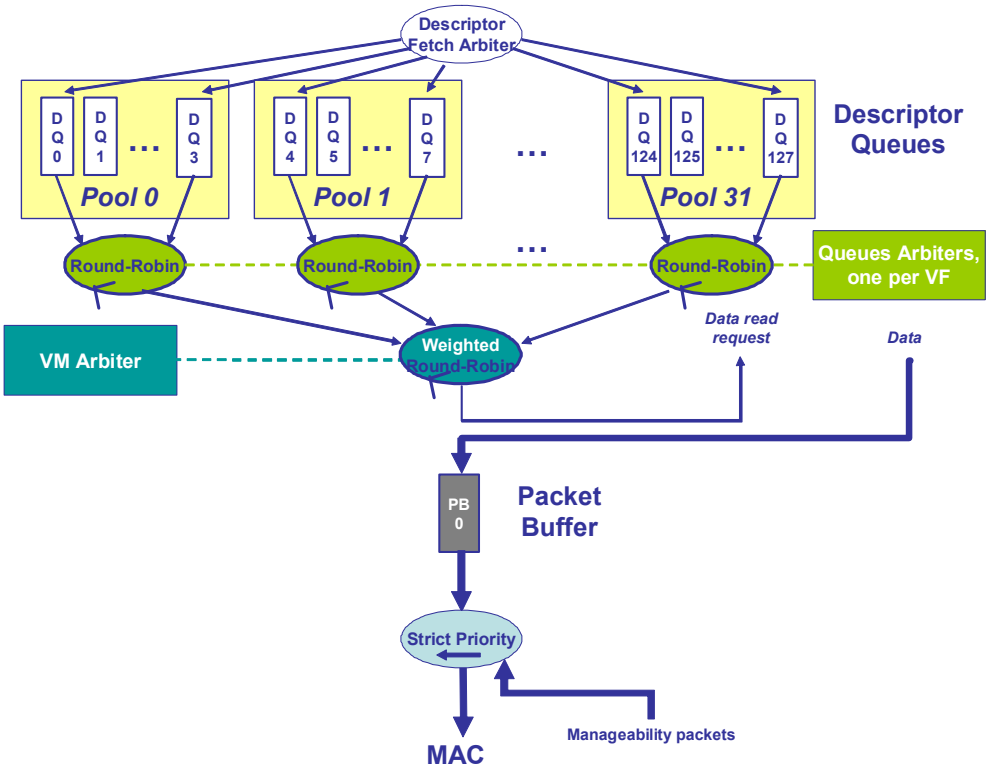


Figure 7-19. Transmit Architecture DCB-off/VT-on — 32 VFs

7.2.1.2.3.4 DCB-off/VT-off

When both DCB and virtualization features are disabled, a single set of up to 64 queues is allocated to a single packet buffer PB(0).

Descriptor Plane Arbiter:

- **Descriptor Queues Round Robin Arbiter** — Descriptors are fetched out from the internal descriptor queues in a frame-by-frame round-robin manner. It is assumed driver dispatches traffic across the queues according to some TSS algorithm similarly to what is done by hardware in the Rx path with RSS.

Packet Plane Arbiter:

- **Manageability** packets are inserted with strict priority over data packets.

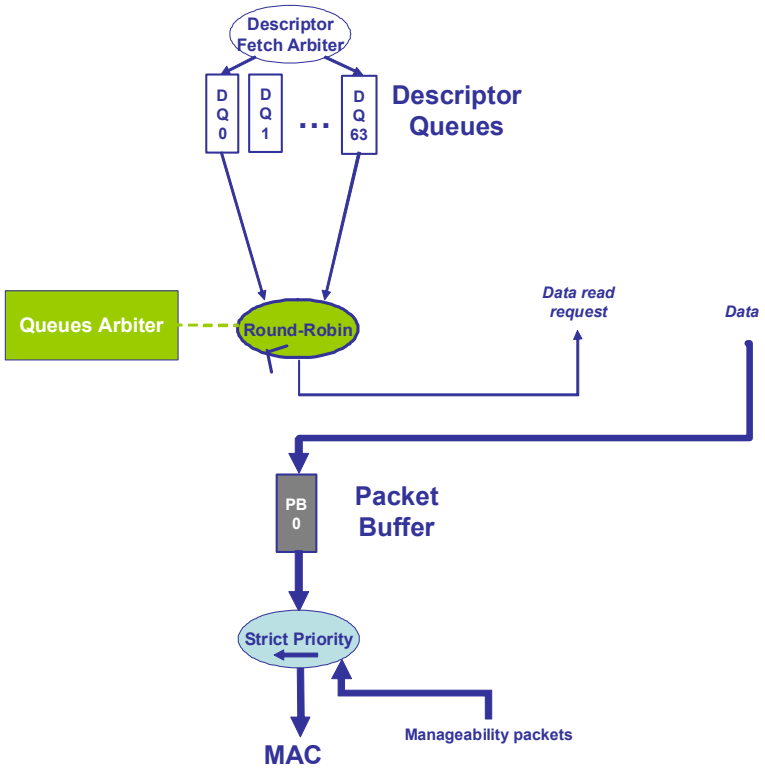


Figure 7-20. Transmit Architecture DCB-off/VT-off

7.2.2 Transmit Contexts

The X550 provides hardware checksum offload and TCP segmentation facilities. These features enable TCP and UDP packet types to be handled more efficiently by performing additional work in hardware, thus reducing the software overhead associated with preparing these packets for transmission. Part of the parameters used to control these features are handled through contexts.

A context refers to a set of parameters providing a particular offload functionality. These parameters are loaded by unique descriptors named transmit context descriptors. A transmit context descriptor is identified by the *DTYP* field (described later in this section) equals to 0x2.

The X550 supports two contexts for each of its 128 transmit queues. The *IDX* bit contains an index to one of these two contexts. Each advanced data descriptor that uses any of the advanced offloading features must refer to a context by the *IDX* field.

Contexts can be initialized with a transmit context descriptor and then used for a series of related transmit data descriptors. Software can use these contexts as long lived ones, while one of the two contexts is used for checksum offload and the other one for transmit segmentation detailed in the following sections. The contexts should be modified when new offload parameters are required.

7.2.3 Transmit Descriptors

7.2.3.1 Introduction

The X550 supports legacy descriptors and advanced descriptors.

Legacy descriptors are intended to support legacy drivers, to enable fast platform power up and to facilitate debug. The legacy descriptors are recognized as such based on *DEXT* bit (see the sections that follow). Legacy descriptors are not supported together with DCB, virtualization, and IPsec. These modes are recognized by a dedicated enable bit for each.

In addition, the X550 supports two types of advanced transmit descriptors:

1. Advanced transmit context descriptor, DTYP = 0010b
2. Advanced transmit data descriptor, DTYP = 0011b

Note: DTYP = 0000b and 0001b are reserved values.

The transmit data descriptor (both legacy and advanced) points to a block of packet data to be transmitted. The advanced transmit context descriptor does not point to packet data. It contains control/context information that is loaded into on-chip registers that affect the processing of packets for transmission. The following sections describe the descriptor formats.

7.2.3.2 Transmit Descriptors Formats

7.2.3.2.1 Notations

This section defines the structure of descriptors that contain fields carried over the network. At the moment, the only relevant field is the *VLAN Tag* field.

The rule for VLAN tag is to use network ordering (also called big endian). It appears in the following manner in the descriptor:

Table 7-32. VLAN Tag

Byte address N + 1 -> first byte on the wire Bit 7 first on the wire <- Bit 0			Byte address N -> second byte on the wire Bit 7 -> last on the wire Bit 0
PRI (3 bits)	DEI	VID (4 bits)	VID (8 bits)

7.2.3.2.2 Legacy Transmit Descriptor Format

To select legacy mode operation, bit 29 (*TDESC.DEXT*) should be set to 0b. In this case, the descriptor format is defined as listed in Table 7-33. Address and length must be supplied by software on all descriptors. Bits in the command byte are optional, as are the *CSO*, and *CSS* fields.

Table 7-33. Transmit Descriptor (TDESC) Layout — Legacy Mode

	63 48	47 40	39 36	35 32	31 24	23 16	15 0
0	Buffer Address [63:0]						
8	VLAN	CSS	Rsvd	STA	CMD	CSO	Length

Table 7-34. Transmit Descriptor Write-Back Format — Legacy Mode

	63 48	47 40	39 36	35 32	31 24	23 16	15 0
0	Reserved				Reserved		
8	VLAN	CSS	Rsvd	STA	CMD	CSO	Length

Buffer Address (64) and Length (16)

The buffer address is a byte-aligned address. Length (*TDESC.LENGTH*) specifies the length in bytes to be fetched from the buffer address provided. The maximum length associated with a single descriptor is 15.5 KB while the total frame size must meet the maximum supported frame size. There is no limitation for the minimum buffer size.

Note: Descriptors with zero length (null descriptors) transfer no data. Null descriptors might appear only between packets and must have their *EOP* bits set.

Checksum Offset and Start — CSO (8) and CSS (8)

A *Checksum Offset* (*TDESC.CSO*) field indicates where, relative to the start of the packet, to insert a TCP checksum if this mode is enabled. A *Checksum Start* (*TDESC.CSS*) field indicates where to begin computing the checksum. Note that *CSO* and *CSS* are meaningful only in the first descriptor of a packet.

Both *CSO* and *CSS* are in units of bytes. These must both be in the range of data provided to the device in the descriptor. This means for short packets that are padded by software, *CSO* and *CSS* must be in the range of the un-padded data length, not the eventual padded length (64 bytes).

CSO must be set to the location of TCP or UDP checksum in the packet. CSS must be set to the beginning of the IP header or the L4 (TCP/UDP) header. Checksum calculation is not done if *CSO* or *CSS* are out of range. This occurs if (*CSS* > length) OR (*CSO* > length - 1).

For the 802.1Q header, the offset values depend on the VLAN insertion enable bit — the *VLE* bit. If they are not set (VLAN tagging included in the packet buffers), the offset values should include the VLAN tagging. If these bits are set (VLAN tagging is taken from the packet descriptor), the offset values should exclude the VLAN tagging.

Hardware does not add the 802.1q EtherType or the VLAN field following the 802.1Q EtherType to the checksum. So for VLAN packets, software can compute the values to back out only on the encapsulated packet rather than on the added fields.

Note: UDP checksum calculation is not supported by the legacy descriptor because the legacy descriptor does not support the translation of a checksum result of 0x0000 to 0xFFFF needed to differentiate between an UDP packet with a checksum of zero and an UDP packet without checksum.

Because the *CSO* field is eight bits wide, it puts a limit on the location of the checksum to 255 bytes from the beginning of the packet.

Note: *CSO* must be larger than *CSS*.

Software must compute an offsetting entry to back out the bytes of the header that should not be included in the TCP checksum and store it in the position where the hardware computed checksum is to be inserted.

Hardware adds the checksum at the byte offset indicated by the *CSO* field. Checksum calculations are for the entire packet starting at the byte indicated by the *CSS* field. The byte offset is counted from the first byte of the packet fetched from host memory.

Command Byte — CMD (8)

The CMD byte stores the applicable command and has the fields listed in Table 7-35.

Table 7-35. Transmit Command (TDESC.CMD) Layout

7	6	5	4	3	2	1	0
RSV	VLE	DEXT	RSV	RS	IC	IFCS	EOP

RSV (bit 7)

Reserved.

VLE (bit 6) - VLAN Packet Enable

When set to 1b, VLE indicates that the packet is a VLAN packet and hardware adds the VLAN header to the Tx packet. The VLAN EtherType is taken from DMATXCTL.*VT* and the 802.1q VLAN tag is taken from the *VLAN* field in the Tx descriptor. See Section 7.4.5 for details about double VLAN.

Table 7-36. VLAN Tag Insertion Decision Table for VLAN Mode Enabled

VLE	Action
0	Send generic Ethernet packet.
1	Send 802.1Q packet; the *Ethernet Type* field comes from the *VET* field of the VLNCTRL register and the VLAN data comes from the *VLAN* field of the Tx descriptor.

Note: This table is relevant only if PFVMVIR.*VLANA* = 00b (use descriptor command) for the queue.

DEXT (bit 5) - Descriptor Extension

Zero for legacy mode.

RSV (bit 4)

Reserved.

RS (bit 3) - Report Status

RS signals hardware to report the DMA completion status indication as well as triggering ITR. Hardware indicates a DMA completion by setting the *DD* bit in the transmit descriptor when TDWBAL[n].*HEAD_WB_EN* = 0b or by Head Write-back if *HEAD_WB_EN* = 1b (see Section 7.2.3.5.2). The *RS* bit is permitted only on descriptors that has the *EOP* bit set (last descriptor of a packet).

Note: Software should not set the *RS* bit when TXDCTL.*WTHRESH* is greater than zero. Instead, the hardware reports the DMA completion according to the *WTHRESH* rules (explained in Section 7.2.3.5.1). This note is relevant only for descriptor write back while in head write-back mode. WTRESH must also be set to zero.

When TXDCTL.*WTHRESH* = zero, software must set the *RS* bit on the last descriptor of every packet.

There are some exceptions for descriptor completion indication in head write-back mode. For more details see Section 7.2.3.5.2.

IC (bit 2) - Insert Checksum

Hardware inserts a checksum at the offset indicated by the *CSO* field if the *Insert Checksum* bit (*IC*) is set.

IFCS (bit 1) - Insert FCS

When set, hardware appends the MAC FCS at the end of the packet. When cleared, software should calculate the FCS for proper CRC check. There are several cases in which software must set IFCS as follows:

- Transmitting a short packet while padding is enabled by the HLREG0.*TXPADEN* bit.
- Checksum offload is enabled by the *IC* bit in the *TDESC.CMD*.
- VLAN header insertion enabled by the *VLE* bit in the *TDESC.CMD* or by the PFVMVIR.*VLANA* fields.
- E-tag insertion enabled by the PFVMVIR.*TAGA* fields.
- TSO or TCP/IP checksum offload using a context descriptor.
- IPsec offload is requested.

Note: TSO and IPsec offload are relevant only to advanced Tx descriptors.

EOP (bit 0) - End of Packet

A packet can be composed of multiple buffers (each of them indicated by its own descriptor). When EOP is set, it indicates the last descriptor making up the packet.

Note: *VLE*, *IFCS*, and *IC* fields should be set in the first descriptor of a packet. The *RS* bit can be set only on the last descriptor of a packet. The *DEXT* bit must be set to zero for all descriptors. The *EOF* bit is meaningful in all descriptors.

STA (4) — Transmitted Status

DD (bit 0) — Descriptor Done Status

This bit provides a status indication that the DMA of the buffer has completed. Software might re-use descriptors with the *DD* bit set and any other descriptors processed by the hardware before this one. The other bits in the *STA* field are reserved.

Rsvd — Reserved (4)

VLAN (16)

The *VLAN* field is used to provide the 802.1q/802.1ac tagging information. The *VLAN* field is qualified on the first descriptor of each packet when the *VLE* bit is set to 1b. The *VLAN* field is provided in network order and is meaningful in the first descriptor of a packet. See Section 7.2.3.2.1 for more details.

Table 7-37. VLAN Field (TDESC.VLAN) Layout

15 13	12	11 0
PRI	DEI	VLAN

7.2.3.2.3 Advanced Transmit Context Descriptor

	63 56	55 48	47 42	41 32	31 16	15 9	8 0
0	TUNNELLEN	OUTERIPLEN	FCoEF	IPsec SA Index	VLAN	MACLEN	IPLEN/ HEADLEN

8	MSS	L4LEN	RSV	IDX	Reserved must be 0x3F	DEXT	RSV	DTYP	TUCMD	IPsec ESP_LEN
	63 48	47 40	39 37	36	35 30	29	28 24	23 20	19 9	8 0

IPLEN/HEADLEN (9)

IPLEN (for IP packets)

This field holds the value of the IP header length for the IP checksum offload feature. If an offload is requested, *IPLEN* must be greater than or equal to 20, and less than or equal to 511. For IP tunnel packets (IPv4-IPv6) *IPLEN* must be defined as the length of the two IP headers. The hardware is able to offload the L4 checksum calculation while software should provide the IPv4 checksum. For IPsec packets, it is the sum of IP header length plus IPsec header length.

HEADLEN (for FCoE packets)

This field indicates the size (in bytes) of the FCoE frame header. The frame header includes the MAC header, optional VLAN and FCoE header(s) as shown in Figure 7-46. *HEADLEN* is meaningful only if transmit FCoE offload is enabled by setting the *FCoE* bit in the *TUCMD* field. *HEADLEN* that matches Figure 7-46 equals 56 for packets without FC extended headers.

MACLEN (7)

For non-FCoE packets: This field indicates the length of the MAC header. When an offload is requested, one of the *TSE* bits (in the advanced transmit data descriptor) or *IXSM* bit or *TXSM* bit are set, *MACLEN* must be larger than or equal to 14, and less than or equal to 127. This field should include only the part of the L2 header supplied by the driver and not the parts added by hardware. The following table lists the value of *MACLEN* in the different cases.

Regular VLAN	Extended tag	MACLEN
By hardware or none	None	14
By hardware or none	VLAN	18

Regular VLAN	Extended tag	MACLEN
By hardware or none	E-tag	22
By software	None	18
By software	VLAN	22
By software	E-tag	26

For FCoE packets: This field is a byte offset to the last DWord of the FCoE header (supplied by the driver) that includes the SOF flag. The FC frame header starts four bytes after the *MACLEN* as shown in Figure 7-46. The *MACLEN* that matches Figure 7-46 equals 28.

VLAN (16)

This field contains the 802.1Q VLAN tag to be inserted in the packet during transmission. This VLAN tag is inserted when a packet using this context has its *DCMD.VLE* bit is set. This field should include the entire 16-bit *VLAN* field including DEI and priority fields as listed in Table 7-37.

Note: The *VLAN* field is provided in network order. See Section 7.2.3.2.1.

IPsec SA IDX (10)

If an IPsec offload is requested for the packet (*IPSEC* bit is set in the advanced Tx data descriptor), indicates the index in the SA table where the IPsec *KEY* and *SALT* are stored for that flow.

FCoEF (6)

EOF (bits 1:0)

End of frame delimiter index (see Section 7.10.2.4 for details).

ORIE (bit 4)

Orientation relative to the last frame in an FC sequence (see Section 7.10.2.4 for details).

SOF (bit 2)

Start of frame delimiter index (see Section 7.10.2.3 for details).

ORIS (bit 5)

Orientation relative to the first frame in an FC sequence (see Section 7.10.2.3 for details).

PARINC (bit 3)

When this bit is set, hardware relates to the *PARAM* field in the FC header as relative offset. In this case, hardware increments the *PARAM* field in TSO by an MSS value on each transmitted packet of the TSO. Software should set the *PARINC* bit when it sets the *Relative Offset Present* bit in the F_CTL.

OUTERIPLEN (8)

This field holds the value of the outer IP header length.

TUNNELLEN (8)

This field holds the length value of the tunnel headers including the inner L2 header.

IPS_ESP_LEN(9)

Size of the ESP trailer and ESP ICV appended by software. Meaningful only if the *IPSEC_TYPE* bit is set in the *TUCMD* field and to single send packets for which the *IPSEC* bit is set in their advanced Tx data descriptor.

TUCMD (11)

RSV (bit 10-7)

Reserved.

FCoE (bit 6)

This bit defines the context descriptor and the associated data descriptors as FCoE frame type. See Section 7.10.2 for a description of the offload provided by the hardware while transmitting a single frame and TSO.

Reserved (bit 5:4)

Reserved.

L4T (bit 3:2)

L4 Packet TYPE (00: UDP; 01: TCP; 10: SCTP; 11: RSV). Should be set to 11b for FCoE packets.

IPV4(bit 1)

IP Packet Type: When 1b, IPv4; when 0b, IPv6.

DTYP (4)

This field is always 0010b for this type of descriptor.

RSV(1)

Reserved.

DEXT (1)

Descriptor Extension (one for advanced mode).

Reserved (6)

Reserved.

IDX (1)

The context descriptor is posted to a context table in hardware. There are two context tables per queue. The IDX is the index of the context tables.

Note: Because the X550 supports only two context descriptors per queue, the two MS bits are reserved and should be set to 0b.

RSV(1)

Reserved.

L4LEN(8)

This field holds the layer 4 header length. If *TSE* is set, this field must be greater than or equal to 8 and less than or equal to 64. Otherwise, this field is ignored. Note that for UDP segmentation the L4 header size equals 8 and for TCP segmentation (with no TCP options) it equals 20.

MSS (16)

This field controls the Maximum Segment Size. This specifies the maximum protocol payload segment sent per frame, not including any header. MSS is ignored when *DCMD.TSE* is not set.

7.2.3.2.3.1 TCP/UDP Segmentation

The total length of each frame (or segment) excluding Ethernet CRC as follows. Note that the last packet of a TCP segmentation might be shorter.

MACLEN + 4(if VLE set) + [4, 8, 14, or 16] + OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + MSS + [PADLEN + 18] (if ESP packet)

PADLEN ranges from zero to three in Tx and is the content of the *ESP Padding Length* field that is computed when offloading ESP in cipher blocks of 16 bytes (AES-128) with respect to the following alignment formula:

[L4LEN + MSS + PADLEN + 2] modulo(4) = 0

For a single send the IPS_ESP_LEN equals to PADLEN + 18.

Note: The headers lengths must meet the following: MACLEN + IPLEN + L4LEN <= 512

7.2.3.2.3.2 FCoE Segmentation

The total length of each frame (or segment) excluding Ethernet CRC is equal to:

MACLEN + 4(if VLE set) + [4, 8, 14, or 16] + 8 (FC CRC + EOF)

Note: For FCoE packets, the maximum segment size defines the FC payload size in all packets but the last one, which can be smaller.

The context descriptor requires valid data only in the fields used by the specific offload options. Table 7-38 lists the required valid fields according to the different offload options.

Table 7-38. Valid Fields by Offload Option

	Context Fields ->	FCoE	FCoEF	VLAN	MACLEN	OUTERIPLEN/TUNNELLEN	IPLEN/HEADLEN	L4LEN		IPV4	L4T	Encryption	IPSECTYPE	SAIDX	ESP_LEN	MSS	QCNTLEN	CC (data descriptor)
Required Offload	**VLAN insertion**			yes													yes	
	IPv4 XSUM	n/a	n/a		yes		yes			1							yes	0
	L4 XSUM	n/a	n/a		yes		yes				yes						yes	0
	TCP/UDP Seg	n/a	n/a		yes		yes	yes		yes	yes					yes	yes	0
	FCoE CRC	yes	yes		yes		yes	n/a		n/a	Yes (11)	n/a	n/a	n/a	n/a		yes	1
	FCoE Seg	yes	yes		yes		yes	n/a		n/a	Yes (11)	n/a	n/a	n/a	n/a	yes	yes	1
	IPsec ESP	n/a	n/a		yes		yes			yes		yes	yes	yes	yes		yes	1
	IPsec AH	n/a	n/a		yes		yes			yes		yes	yes	yes	yes		yes	1
	Tunnel XSUM	n/a	n/a		yes	yes	yes				yes							1
	Tunnel Seg	n/a	n/a		yes	yes	yes	yes		yes	yes					yes	yes	1
	Tx switch	n/a	n/a		yes		yes	yes		yes	yes	n/a	n/a	n/a	n/a		yes	1

Note: All fields that are not used in the context descriptor must be set to zero.

7.2.3.2.4 Advanced Transmit Data Descriptor

Table 7-39. Advanced Transmit Data Descriptor Read Format

	63 46	45 40	39	38 36	35 32	31 24	23 20	19 18	17 16	15 0
0	Address[63:0]									
8	PAYLEN	POPTS	CC	IDX	STA	DCMD	DTYP	MAC	TUNNEL	DTALEN

Table 7-40. Advanced Transmit Data Descriptor Write-back Format

	63 36	35 32	31 0
0	RSV		
8	RSV	STA	RSV

Address (64)

This field holds the physical address of a data buffer in host memory, which contains a portion of a transmit packet. This field is meaningful in all descriptors.

DTALEN (16)

This field holds the length in bytes of data buffer at the address pointed to by this specific descriptor. This field is meaningful in all descriptors. The maximum length is 16 KB with no limitations on the minimum size. Refer to the comment on descriptors with zero length described in the sections that follow.

TUNNEL (2)

Tunnel Type (bit 0):

0 = VXLAN
1 = NVGRE

OUTERIPCS (bit 1) - Outer IP checksum enable

Indicates this is a tunnel packet and that the *OUTERIPLEN/TUNNELLEN* parameters in the context descriptor are valid.

MAC (2)

This field is meaningful on the first descriptor of the packet(s).

Reserved (bit 0)

Reserved.

1588 (bit 1)

IEEE1588 time stamp packet.

DTYP (4)

0011b for advanced data descriptor. DTYP should be valid in all descriptors of the packet(s).

DCMD (8)

TSE (bit 7) - Transmit Segmentation Enable

This bit indicates a TCP or FCoE segmentation request. When *TSE* is set in the first descriptor of a TCP or FCoE packet, hardware must use the corresponding context descriptor to perform segmentation.

Note: It is recommended that HLREG0.*TXPADEN* be enabled when TSE is used since the last frame can be shorter than 60 bytes — resulting in a bad frame.

VLE (bit 6) - VLAN Packet Enable

This bit indicates that the packet is a VLAN packet (hardware must add the VLAN EtherType and an 802.1q VLAN tag to the packet).

DEXT (bit 5) - Descriptor Extension

This bit must be one to indicate advanced descriptor format (as opposed to legacy).

Rsv (bit 4)

Reserved.

RS (bit 3) - Report Status

See the description in the legacy transmit descriptor in Section 7.2.3.2.2.

Rsv (bit 2)

Reserved.

IFCS (bit 1) - Insert FCS

When this bit is set, the hardware appends the MAC FCS at the end of the packet. When cleared, software should calculate the FCS for proper CRC check. There are several cases in which software must set IFCS as follows:

- Transmitting a short packet while padding is enabled by the HLREG0.*TXPADEN* bit.
- Checksum offload is enabled by the either *IC*, *TXSM* or *IXSM* bits in the *TDESC.DCMD.*
- VLAN header insertion enabled by the *VLE* bit in the *TDESC.DCMD*.
- FC CRC (FCoE) offload is enabled by the *FCoE* bit in the transmit context descriptor.
- TCP or FCoE segmentation offload enabled by the *TSE* bit in the *TDESC.DCMD*.

EOP (bit 0) - End of Packet

A packet might be composed of multiple buffers (each of them is indicated by its own descriptor). When *EOP* is set, it indicates the last descriptor making up the packet. In transmit segmentation (explained later on in this section) the *EOP* flag indicates the last descriptor of the last packet of the segmented transmission.

Note: *TSE*, *VLE*, and *IFCS* fields should be set in the first descriptor of the packet(s). The *RS* bit can be set only on the last descriptor of the packet. The *EOP* bit is valid in all descriptors. The *DEXT* bit must be set to 1b for all descriptors.

Descriptors with zero length, transfer no data. If the *RS* bit in the command byte is set, the *DD* field in the status word is not written when hardware processes them.

STA (4)

Rsv (bit 3:1)

Reserved.

DD (bit 0) - Descriptor Done

The *DD* bit provides a status indication that the DMA of the buffer has completed. Software might re-use descriptors with the *DD* bit set, and any other descriptors processed by hardware before this one. In TSO, the buffers that include the TSO header are used multiple times during transmission and special considerations should be made as described in Section 7.2.4.2.2.

IDX (3)

This field holds the index into the hardware context table to indicate which of the two per-queue contexts should be used for this request. If no offload is required and the *CC* bit is cleared, this field is not relevant and no context needs to be initiated before the packet is sent. See Table 7-38 for details of which packets requires a context reference. This field is relevant only on the first descriptor of the packet(s).

CC (1)

Check Context bit — When set, a Tx context descriptor indicated by IDX index should be used for this packet(s). The *CC* bit should be set in the following cases:

1. Non-zero *QCNTLEN* field is required (defined in the context descriptor).
2. Any FCoE offload is required.
3. Tx switching is enabled (including anti-spoof checks).

POPTS (6)

Rsv (bit 5:2)

Reserved.

TXSM (bit 1) - Insert TCP/UDP Checksum

When set to 1b, the L4 checksum must be inserted. In this case, TUCMD.LP4 indicates whether the checksum is TCP or UDP or SCTP. When DCMD.TSE is set, TXSM must be set to 1b. If this bit is set, the packet should at least contain a TCP header.

IXSM (bit 0) - Insert IP Checksum

This field indicates that IP checksum must be inserted. In IPv6 mode, it must be reset to 0b. If DCMD.TSE and TUCMD.IPV4 are set, IXSM must be set to 1b. If this bit is set, the packet should at least contain an IP header.

This field is relevant only on the first descriptor of the packet(s).

PAYLEN (18)

PAYLEN indicates the size (in byte units) of the data buffer(s) in host memory for transmission.

In a single-send packet, *PAYLEN* defines the entire packet size fetched from host memory. It does not include the fields that hardware adds such as: optional VLAN tagging, the FCoE trailer containing the FC CRC and EOF (For FCoE packets) and Ethernet CRC or Ethernet padding.

When IPsec offload is enabled, the *PAYLEN* field does not include the ESP trailer added by hardware.

In TSO (regardless if it is transmitted on a single or multiple packets), the PAYLEN defines the protocol payload size fetched from host memory.

- In TCP or UDP segmentation offload, *PAYLEN* defines the TCP/UDP payload size.
- In FCoE TSO offload, the *PAYLEN* field defines the FC payload size. It includes the FC option headers (if present) and the FC data payload but excludes the FCoE trailer containing the FC CRC and EOF.

This field is relevant only on the first descriptor of the packet(s). The minimum transmitted packet size excluding VLAN padding and CRC bytes is 17 and the *PAYLEN* size should meet this limitation. On a single-packet send, the maximum size of the *PAYLEN* is dictated by the maximum allowed packet size which is 15.5 KB. On TSO, the maximum *PAYLEN* can be up to 2^{18}-1.

Note: When a packet spreads over multiple descriptors, all of the descriptor fields are valid only on the first descriptor of the packet, except for *RS* and *EOP* bits, which are set on the last descriptor of the packet.

7.2.3.3 Transmit Descriptor Ring

The transmit descriptor ring structure (shown in Figure 7-21) uses a contiguous memory space. A set of four registers (described later in this section) maintain the transmit descriptor ring in the host memory. Hardware maintains internal circular queues of 64 descriptors per queue to hold the descriptors that were fetched from the software ring.

Descriptors handled to hardware should not be manipulated by software until hardware completes its processing. It is indicated by advancing the head pointer beyond these descriptors.

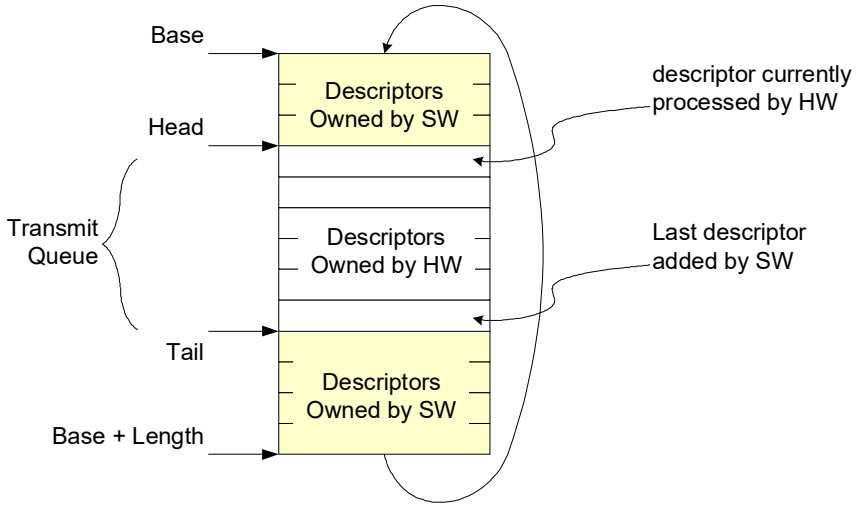


Figure 7-21. Transmit Descriptor Ring Structure

The transmit descriptor ring is defined by the following registers:

- **Transmit Descriptor Base Address register (TDBA 0-127)** — This register indicates the start address of the descriptor ring buffer in the host memory; this 64-bit address is aligned on a 128-byte boundary and is stored in two consecutive 32-bit registers. Hardware ignores the lower 7 bits.
- **Transmit Descriptor Length register (TDLEN 0-127)** — This register determines the number of bytes allocated to the circular buffer. This value must be 0 modulo 128.
- **Transmit Descriptor Head register (TDH 0-127)** — This register holds a value that is an offset from the base and indicates the in-progress descriptor. There can be up to 64 K minus 8 descriptors in the circular buffer. The transmit queue consists of the descriptors between the head and tail pointers. Transmission starts with the descriptor pointer by the head registers. When the DMA engine processes a descriptor, it might optionally write back the completed descriptor and then advance the head pointer. It then processes the next descriptor up to the point that the head pointer reaches the tail. Head equals tail means that the transmit queue in host memory is empty. Reading this register indicates the hardware progress to the software. All descriptors behind the head pointer and in front of tail register are owned by the software. The other descriptors are owned by the hardware and should not be modified by the software.

- **Transmit Descriptor Tail register (TDT 0-127)** — This register holds a value, which is an offset from the base, and indicates the location beyond the last descriptor hardware can process. Software adds new descriptors to the ring by writing descriptors in the circular buffer pointed by the tail pointer. The new descriptor(s) are indicated to hardware by updating the tail pointer one descriptor above the last added descriptor. Note that a single packet or TSO might be composed of multiple descriptors. The transmit tail pointer should never point to the middle of a packet or TSO, which might cause undesired software/hardware races.

Software might detect which packets have already been processed by hardware using the following:

- Read the TDH head register to determine which packets (those logically before the head) have been transferred to the on-chip FIFO or transmitted. This method is not recommended as races between the internal update of the head register and the actual write back of descriptors can occur.
- When head write back is enabled (TDWBAL[n].*HEAD_WB_EN* = 1b) software might read the image of the head pointer in host memory at the address defined by TDWBAH[n]/TDWBAL[n] pair. Hardware updates the head image in host memory by completed descriptors as described in Section 7.2.3.5.2.
- When head write back is not enabled (TDWBAL[n].*HEAD_WB_EN* = 0b), software might track the *DD* bits in the descriptor ring. Descriptor write back is controlled by the *RS* bit and the *WTHRESH* setting as well as interrupt assertion.
- Issue an interrupt. An interrupt condition is generated each time a packet was transmitted or received and a descriptor was write back or transmit queue goes empty (EICR.*RTxQ*[0-19]). This interrupt can either be enabled or masked.

All of the registers controlling the descriptor rings behavior should be set before transmit is enabled.

7.2.3.4 Transmit Descriptor Fetching

The X550 fetches new descriptors as required for packet transmission depending on its on-die descriptor buffer state:

- **Fetch** — The on-chip descriptor buffer is empty or contains less descriptors than a complete packet.
  - A fetch starts as soon as any descriptors are made available (host writes to the tail pointer).
  - A request is issued for any available descriptors up to the size of the on-die buffer.
  - Once the sum of on-die descriptors and requested descriptors is more than required for a single packet, the buffer transitions to the pre-fetch state.
  - If several on-chip descriptor queues are empty simultaneously, queues are served in round robin arbitration except the queues indicated as strict priority, which are served first.
- **Pre-Fetch** — The on-chip descriptor buffer becomes almost empty while there are enough descriptors in the host memory.
  - The on-chip descriptor buffer is defined as almost empty if it contains less descriptors than the threshold defined by TXDCTL[n].*PTHRESH*.
  - The transmit descriptor contains enough descriptors if it includes more ready descriptors than the threshold defined by TXDCTL[n].*HTHRESH*.
  - In pre-fetch mode, descriptors are fetched only after there are no other DMA activity of greater priority as: transmit descriptor fetch; status write-backs or packet data transfers).
  - A request is issued for any available descriptors up to the capacity of the on-die buffer.

— If several on-chip descriptor queues are in this situation simultaneously, queues are served in round robin arbitration except the queues indicated as strict priority which are served first.

- **Idle** — Requests are not issued. This is the state reached when none of the previous states apply.

Note: Software must update the Tail register on packet boundaries. That is, the last valid descriptor might not be a context descriptor and must have the *EOP* bit set.

7.2.3.4.1 Transmit Descriptor Fetch and Write-back Settings

This section describes the settings of transmit descriptor thresholds. It relates to fetch thresholds described above as well as the write-back threshold (*WTHRESH*) when operating in descriptor write-back mode, which is described in Section 7.2.3.5.1.

- Transmit descriptor fetch setting is programmed in the TXDCTL[n] register per queue. The default settings of *PTHRESH*, *HTHRESH* and *WTHRESH* are zero's.
- To reduce transmission latency, it is recommended to set the *PTHRESH* value as high as possible while the *HTHRESH* and *WTHRESH* as low as possible (down to zero).
- To minimize PCIe overhead the *PTHRESH* should be set as low as possible while *HTHRESH* and *WTHRESH* should be set as high as possible.
- The sum of *PTHRESH* plus *WTHRESH* must not be greater than the on-chip descriptor buffer size.
- Some practical rules:
  - CPU cache line optimization — Assume 'N' equals the CPU cache line divided by 16 (descriptor size). Then, to align descriptors pre-fetch to CPU cache line (in most cases), it is advised to set *PTHRESH* to the on-chip descriptor buffer size minus 'N' and *HTHRESH* to 'N'. To align descriptor write back to the CPU cache line it is advised to set *WTHRESH* to either 'N' or even 2 times 'N'. Note that partial cache line writes might significantly degrade performance. Therefore, it is highly recommended to follow this advice.
  - Minimizing PCIe overhead — As an example, setting *PTHRESH* to the on-chip descriptor buffer size minus 16 and *HTHRESH* to 16 minimizes the PCIe request and header overhead to ~20% of the bandwidth required for the descriptor fetch.
  - Minimizing transmission latency from tail update — Setting *PTHRESH* to the on-chip descriptor buffer size minus 'N' ('N' previously defined) while *HTHRESH* and *WTHRESH* to zero.
  - Threshold settings in DCB mode — Note that only values of *PTHRESH* equals on-chip descriptor buffer size minus 8 and *HTHRESH* equals 4 were thoroughly tested.

Note: As previously described, device setting is a trade-off between overhead (translated to performance) and latencies. It is expected that some level of optimization is done at software driver development phase. Customers who want better performance might need to adjust the threshold values according to the previous guidelines while optimizing to specific platform and targets.

7.2.3.5 Transmit Write Back

The X550 periodically updates software on its progress in processing transmit buffers. Two methods are described for doing so:

- Updating by writing back into the Tx descriptor.
- Update by writing to the head pointer in system memory.

7.2.3.5.1 Tx Descriptor Write Back

When the TXDCTL[n].*WTHRESH* equals zero, descriptors are written back for those descriptors with the *RS* bit set. When the TXDCTL[n].*WTHRESH* value is greater than zero, descriptors are accumulated until the number of accumulated descriptors equals the TXDCTL[n].*WTHRESH* value, then these descriptors are written back. Accumulated descriptor write back enables better use of the PCIe bus and memory bandwidth.

Any descriptor write back includes the full 16 bytes of the descriptor.

Descriptors are written back in one of three cases:

- TXDCTL[n].*WTHRESH* = 0 and a descriptor that has *RS* set is ready to be written back.
- TXDCTL[n].*WTHRESH* > 0 and *TXDCTL[n].WTHRESH* descriptors have accumulated.
- TXDCTL[n].*WTHRESH* > 0 and the corresponding *EITR* counter has reached zero. The timer expiration flushes any accumulated descriptors and sets an interrupt event (*TXDW*).

An additional mode in which transmit descriptors are not written back at all and the head pointer of the descriptor ring is written instead is described in the following section.

7.2.3.5.2 Tx Head Pointer Write Back

In legacy hardware, transmit requests are completed by writing the *DD* bit to the transmit descriptor ring. This causes cache trash since both the driver and hardware are writing to the descriptor ring in host memory. Instead of writing the *DD* bits to signal that a transmit request is complete, hardware can write the contents of the descriptor queue head to host memory. The driver reads that memory location to determine which transmit requests are complete. To improve the performance of this feature, the driver needs to program TPH registers to configure which CPU processes each Tx queue.

The head pointer is reflected in a memory location that is allocated by software for each queue.

Rules for head pointer write back:

- Head write back occurs if TDWBAL[n].*HEAD_WB_EN* is set for this queue, and the *RS* bit is set in the Tx descriptor, following its corresponding data upload into packet buffer.
  - If the head write-back feature is enabled, software must set *WTHRESH* to 0x0 while only descriptors with the *RS* bit set, generate header write back.
  - Note that the head pointer write back does not hold transmission. Instead, if packets with the *RS* bit are transmitted fast enough, it might happen that the header pointer write back is not updated for each and every packet. In addition, it might happen that the head pointer write back might be updated up to descriptors that do not have the *RS* bit set. In such cases, hardware might report a completion of a descriptor that might not be the last descriptor in a TSO or even the last descriptor in a single packet.

The driver has control of this feature per queue through the TDWBAL and TDWBAH registers.

The low register's LSB hold the control bits.

- The *HEAD_WB_EN* bit enables activation of tail write back. In this case, no descriptor write back is executed.
- The 30 upper bits of this register hold the lowest 32 bits of the head write-back address, assuming that the two last bits are zero.

The high register holds the high part of the 64-bit address.

Note: Hardware writes a full DWord when writing this value, so software should reserve enough space for each head value and make sure the TDBAL value is DWord-aligned.

7.2.4 TCP and UDP Segmentation

Hardware TCP segmentation is one of the offloading options supported by the Windows* and Linux* TCP/IP stack. This is often referred to as Large Send offloading or TSO. This feature enables the TCP/IP stack to pass to the network device driver a message to be transmitted that is bigger than the Maximum Transmission Unit (MTU) of the medium. It is then the responsibility of the device driver and hardware to divide the TCP message into MTU size frames that have appropriate layer 2 (Ethernet), 3 (IP), and 4 (TCP) headers. These headers must include sequence number, checksum fields, options and flag values as required. Note that some of these values (such as the checksum values) are unique for each packet of the TCP message, and other fields such as the source IP Address is constant for all packets associated with the TCP message.

Similar to TCP segmentation, the X550 also provides a capability to offload UDP segmentation. Note that current UDP segmentation offload is not supported by any standard operating system.

Note: CRC appending (HLREG0.*TXCRCEN*) must be enabled in TCP / UDP segmentation mode because CRC is inserted by hardware.

Padding (HLREG0.*TXPADEN*) must be enabled in TCP / UDP segmentation mode, since the last frame might be shorter than 60 bytes — resulting in a bad frame if *TXPADEN* is disabled.

The offloading of these mechanisms to the device driver and the X550 saves significant CPU cycles. The device driver shares the additional tasks to support these options with the X550.

7.2.4.1 Assumptions and Restrictions

The following assumptions apply to the TCP / UDP segmentation implementation in the X550:

- To limit the internal cache dimensions, software is required to spread the header onto a maximum four descriptors, while still allowed to mix header and data in the last header buffer. This limitation stands for up to Layer 4 header included, and for IPv4 or IPv6 independently.
- The maximum size of a single TSO can be as large as defined by the *PAYLEN* field in the Tx data descriptor (such as up to 256 KB).
- The *RS* bit operation is not changed. Interrupts are set after data in the buffers pointed to by individual descriptors is transferred (DMA'ed) to hardware.
- SNAP packets are not supported for segmentation.
- IP in IP tunneled packets are not supported for offloading under TSO operation. VXLAN and NVGRE tunneled packets can be segmented.
- Software must enable the Ethernet CRC offload in the HLREG0.*TXCRCEN* register since CRC must be inserted by hardware after the checksum has been calculated.
- Software must initialize the appropriate checksum fields in the packet's header.

7.2.4.2 Transmission Process

The transmission process involves the following:

- The protocol stack receives from an application a block of data that is to be transmitted.
- The protocol stack calculates the number of packets required to transmit this block based on the MTU size of the media and required packet headers.

- The stack interfaces with the device driver and passes the block down with the appropriate header information: Ethernet, IP and TCP / UDP headers.
- The stack interfaces with the device driver and commands the driver to send the individual packet. The device driver sets up the interface to the hardware (via descriptors) for the TCP / UDP segmentation.
- The hardware transfers (DMA's) the packet data and performs the Ethernet packet segmentation and transmission based on offset and payload length parameters in the TCP/IP or UDP/IP context descriptor including:
  - Packet encapsulation
  - Header generation and field updates including IPv4/IPv6 and TCP/UDP checksum generation.
- The driver returns ownership of the block of data to the NOS when the hardware has completed the DMA transfer of the entire data block.

7.2.4.2.1 TCP and UDP Segmentation Data Fetch Control

To perform TCP / UDP segmentation in the X550, the DMA must be able to fit at least one packet of the segmented payload into available space in the on-chip packet buffer. The DMA does various comparisons between the remaining payload and the packet buffer available space, fetching additional payload and sending additional packets as space permits.

The X550 enables interleaving between different TSO requests at an Ethernet packet level. In other words, the X550 might fetch part of a TSO from a queue, equivalent to one or more Ethernet packets, then transition to another queue and fetch the equivalent of one or more packets (TSO or not), then move to another queue (or the first queue), etc. The X550 decides on the order of data fetched based on its QoS requirements (such as bandwidth allocation and priority).

To enable interleaving between descriptor queues at the Ethernet frame resolution inside TSO requests, the frame header pointed by the so called header descriptors are re-read from system memory for every TSO segment (once per packet), storing in an internal cache only the header's descriptors instead of the header's content.

7.2.4.2.2 TCP and UDP Segmentation Write-back Modes

TCP / UDP segmentation mode uses the buffers that contain the header of the packet multiple times (once for each transmitted segment). Software should guarantee that the header buffers are available throughout the entire TSO transmission. Therefore, software should not re-use any descriptors of the TSO header during the TSO transmission.

7.2.4.3 TCP and UDP Segmentation Performance

Performance improvements for a hardware implementation of TCP / UDP segmentation offload include:

- The stack does not need to partition the block to fit the MTU size, saving CPU cycles.
- The stack only computes one Ethernet, IP, and TCP / UDP header per segment, saving CPU cycles.
- The stack interfaces with the device driver only once per block transfer, instead of once per frame.
- Larger PCI bursts are used, which improves bus efficiency (such as lowering transaction overhead).
- Interrupts are easily reduced to one per TCP / UDP message instead of one per packet.
- Fewer I/O accesses are required to command the hardware.

7.2.4.4 Packet Format

A TCP / UDP message can be as large as 256 KB and is generally fragmented across multiple pages in host memory. The X550 partitions the data packet into standard Ethernet frames prior to transmission. The X550 supports calculating the Ethernet, IP, TCP, and UDP headers, including checksum, on a frame-by-frame basis. For tunneled packets (NVGRE, VXLAN), The X550 also supports updating the outer IPv4 header.

Table 7-41. TCP/IP and UDP/IP Packet Format Sent by Host

Pseudo Header				Data
Ethernet	Optional tunnel header	IPv4/IPv6	TCP/UDP	DATA (full TCP/UDP message)

Table 7-42. Packets Format Sent by Device

Pseudo Header (updated)	Data (first MSS)	FCS	...	Pseudo Header (updated)	Data (Next MSS)	FCS	...

Frame formats supported by the X550 include:

- Ethernet 802.3
- IEEE 802.1Q VLAN (Ethernet 802.3ac)
- Ethernet Type 2
- NVGRE (see frame format in Section A.2.5.1)
- VXLAN (see frame format in Section A.2.5.2)
- IPv4 headers with options
- IPv6 headers with extensions
- TCP with options
- UDP with options

VLAN tag insertion can be handled by hardware.

Note: UDP (unlike TCP) is not a reliable protocol and fragmentation is not supported at the UDP level. UDP messages that are larger than the MTU size of the given network medium are normally fragmented at the IP layer. This is different from TCP, where large TCP messages can be fragmented at either the IP or TCP layers depending on the software implementation.

The X550 has the ability to segment UDP traffic (in addition to TCP traffic); however, because UDP packets are generally fragmented at the IP layer, the X550's segmentation capability might not be used in practice for UDP.

7.2.4.5 TCP and UDP Segmentation Indication

Software indicates a TCP/UDP segmentation transmission context to the hardware by setting up a TCP/IP or UDP/IP context transmit descriptor (see Section 7.2.3). The purpose of this descriptor is to provide information to the hardware to be used during the TCP / UDP segmentation offload process.

Setting the *TSE* bit in the *DCMD* field to one (in the data descriptor) indicates that this descriptor refers to the segmentation context (as opposed to the normal checksum offloading context). This causes the checksum offloading, packet length, header length, and maximum segment size parameters to be loaded from the descriptor into the device.

The TCP/UDP segmentation prototype header is taken from the packet data itself. Software must identity the type of packet that is being sent (IPv4/IPv6, TCP/UDP, other), calculate appropriate checksum off loading values for the desired checksums, and then calculate the length of the header that is pre-appended. The header can be up to 240 bytes in length.

Once the TCP/UDP segmentation context has been set, the next descriptor provides the initial data to transfer. This first descriptor(s) must point to a packet of the type indicated. Furthermore, the data it points to might need to be modified by software as it serves as the prototype header for all packets within the TC /UDP segmentation context. The following sections describe the supported packet types and the various updates that are performed by hardware. This should be used as a guide to determine what must be modified in the original packet header to make it a suitable prototype header.

The following summarizes the fields considered by the driver for modification in constructing the prototype header.

IP Header

For IPv4 headers:

- Identification field should be set as appropriate for first packet of send (if not already).
- Header checksums of inner and outer IP headers should be zeroed out unless some adjustment is needed by the driver.

TCP Header

- Sequence number should be set as appropriate for first packet of send (if not already).
- PSH, and FIN flags should be set as appropriate for LAST packet of send.
- TCP checksum should be set to the partial pseudo-header checksum as follows (there is a more detailed discussion of this in Section 7.2.4.6:

Table 7-43. TCP Partial Pseudo-header Checksum for IPv4

IP Source Address		
IP Destination Address		
Zero	Layer 4 Protocol ID	Zero

Table 7-44. TCP Partial Pseudo-header Checksum for IPv6

IPv6 Source Address	
IPv6 Final Destination Address	
Zero	
Zero	Next Header

UDP Header

- Checksum should be set as in TCP header, as previously explained.

The following sections describe the updating process performed by the hardware for each frame sent using the TCP segmentation capability.

7.2.4.6 Transmit Checksum Offloading with TCP and UDP Segmentation

The X550 supports checksum offloading as a component of the TCP/UDP segmentation off-load feature and as stand-alone capability. Section 7.2.5 describes the interface for controlling the checksum off-loading feature. This section describes the feature as it relates to TCP/UDP segmentation.

The X550 supports IP and TCP header options in the checksum computation for packets that are derived from the TCP segmentation feature.

Two specific types of checksum are supported by the hardware in the context of the TCP/UDP segmentation off-load feature:

- IPv4 checksum
- TCP/UDP checksum

Each packet that is sent via the TCP/UDP segmentation off-load feature optionally includes the IPv4 checksum and/or the TCP/UDP checksum.

All checksum calculations use a 16-bit wide one's complement checksum. The checksum word is calculated on the outgoing data.

Refer to Table 7-45 for the list of supported transmit checksums per packet type.

7.2.4.7 IP/TCP/UDP Header Updating

IP/TCP and IP/UDP header is updated for each outgoing frame based on the header prototype that hardware DMA's from the first descriptor(s). The checksum fields and other header information are later updated on a frame-by-frame basis. The updating process is performed concurrently with the packet data fetch.

The following sections define what fields are modified by hardware during the TCP/UDP segmentation process by the X550.

7.2.4.7.1 TCP/IP/UDP Header for the First Frame

The hardware makes the following changes to the headers of the first packet that is derived from each TCP segmentation context.

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follows:

- IP Total Length = OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + MSS
- Calculates the Outer IP Checksum

IPv4 Header

- IP Total Length = MSS + L4LEN + IPLEN
- Calculates the IP Checksum

IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension[1]

1. IPV6_HDR_extension is calculated as IPLEN — 40 bytes.

TCP Header

- Sequence Number: The value is the sequence number of the first TCP byte in this frame.
- The flag values of the first frame are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGL.*TCP_FLG_FIRST_SEG*. The default values of the DTXTCPFLGL.*TCP_FLG_FIRST_SEG* are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculates the TCP checksum.

UDP Header

- Calculates the UDP checksum.

7.2.4.7.2 TCP/IP Header for the Subsequent Frames

The hardware makes the following changes to the headers for subsequent packets that are derived as part of a TCP segmentation context:

Number of bytes left for transmission = PAYLEN — (N * MSS)

Where N is the number of frames that have been transmitted.

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follows:

- IP Identification: increased from last value (wrap around based on 16-bit width)
- IP Total Length = OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + MSS
- Calculates the Outer IP Checksum

IPv4 Header

- IP Identification: increased from last value (wrap around)
- IP Total Length = MSS + L4LEN + IPLEN
- Calculate the IP Checksum

IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension[1]

TCP Header

- Sequence Number update: Add previous TCP payload size to the previous sequence number value. This is equivalent to adding the MSS to the previous sequence number.
- The flag values of the subsequent frames are set by logic AND function between the flag word in the pseudo header with the DTXTCPFLGL.*TCP_FLG_MID_SEG*. The default values of the DTXTCPFLGL.*TCP_FLG_MID_SEG* are set.
- Calculate the TCP checksum

UDP Header

- Calculates the UDP checksum.

7.2.4.7.3 TCP/IP Header for the Last Frame

Hardware makes the following changes to the headers for the last frame of a TCP segmentation context:

Last frame payload bytes = PAYLEN — (N * MSS)

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follows:

- IP Identification: increased from last value (wrap around based on 16-bit width)
- IP Total Length = OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + last frame payload bytes.
- Calculates the Outer IP Checksum

IPv4 Header

- IP Total length = last frame payload bytes + L4LEN + IPLEN
- IP identification: increased from last value (wrap around based on 16-bit width)
- Calculate the IP checksum

IPv6 Header

- Payload length = last frame payload bytes + L4LEN + IPV6_HDR_extension[1]

TCP Header

- Sequence number update: Add previous TCP payload size to the previous sequence number value. This is equivalent to adding the MSS to the previous sequence number.
- The flag values of the last frames are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGH.*TCP_FLG_LST_SEG*. The default values of the DTXTCPFLGH.*TCP_FLG_LST_SEG* are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculate the TCP checksum

UDP Header

- Calculates the UDP checksum.

1. IPV6_HDR_extension is calculated as IPLEN — 40 bytes.

7.2.5 Transmit Checksum Offloading in Non-Segmentation Mode

The previous section on TCP/UDP segmentation offload describes the IP/TCP/UDP checksum offloading mechanism used in conjunction with segmentation. The same underlying mechanism can also be applied as a stand-alone checksum offloading. The main difference in a single packet send is that only the checksum fields in the IP/TCP/UDP headers are calculated and updated by hardware.

Before taking advantage of the X550's enhanced checksum offload capability, a checksum context must be initialized. For a single packet send, *DCMD.TSE* should be set to zero (in the data descriptor). For additional details on contexts, refer to Section 7.2.3.3.

Enabling checksum offload, software must also enable Ethernet CRC offload by the HLREG0.*TXCRCEN* since CRC must be inserted by hardware after the checksum has been calculated.

Each checksum operates independently. Insertion of the IP and TCP/UDP checksum for each packet are enabled through the transmit data descriptor *POPTS.TXSM* and *POPTS.IXSM* fields, respectively.

7.2.5.1 IP Checksum

Three fields in the transmit context descriptor set the context of the IP checksum offloading feature:

- *TUCMD.IPV4*
- *IPLEN*
- *MACLEN*

TUCMD.IPV4=1 specifies that the packet type for this context is IPv4, and that the IP header checksum should be inserted. *TUCMD.IPV4*=0 indicates that the packet type is IPv6 (or some other protocol) and that the IP header checksum should not be inserted.

MACLEN specifies the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum, the start of the IP header. The minimal allowed value for this field is 14. Note that the maximum value for this field is 127. This is adequate for typical applications.

Note: The MACLEN+IPLEN value must be less than the total DMA length for a packet. If this is not the case, the results are unpredictable.

IPLEN specifies the IP header length. Maximum allowed value for this field is 511 bytes.

MACLEN+*IPLEN* specify where the IP checksum should stop. The sum of *MACLEN*+*IPLEN* must be smaller equals to the first 638 (127+511) bytes of the packet and obviously must be smaller or equal to the total length of a given packet. If this is not the case, the result is unpredictable.

For IP tunnel packets (IPv4-IPv6), *IPLEN* must be defined as the length of the two IP headers. Hardware is able to offload the L4 checksum calculation while software should provide the IPv4 checksum.

For NVGRE and VXLAN tunneled packet as defined by the *Tunnel* bit in the descriptor, Outer IPv4 checksum is controlled by *OUTERIPCS* bit and is determined by *MACLEN* and *OUTERIPLEN* similar to the above description. Inner IPv4 checksum is controlled by *TUCMD.IPv4* and offset determination should be adjusted by hardware such that *MACLEN* should be replaced by (*MACLEN*+*OUTIPLEN*+*TUNNELLEN*)

The 16-bit IPv4 header checksum is placed at the two bytes starting at MACLEN+10.

7.2.5.2 TCP and UDP Checksum

Three to five fields in the transmit context descriptor set the context of the TCP/UDP checksum offloading feature:

- *MACLEN*
- *OUTERIPLEN* (optional)
- *TUNNELLEN* (optional)
- *IPLEN*
- *TUCMD.L4T*

TUCMD.L4T=01b specifies that the packet type is TCP, and that the 16-bit TCP header checksum should be inserted at byte offset *MACLEN*+*IPLEN*+16. *TUCMD.L4T*=00b indicates that the packet is UDP and that the 16-bit checksum should be inserted starting at byte offset *MACLEN*+*IPLEN*+6. If *Tunnel* bit is set, the checksum is inserted at *MACLEN*+*OUTERIPLEN*+*TUNNELLEN*+*IPLEN*+6.

MACLEN+(*OUTERIPLEN*+*TUNNELLEN*)+*IPLEN* specifies the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum, the start of the UDP/TCP header. See MACLEN table in Section 7.2.3.2.3 for its relevant values.

Note: The *MACLEN*+(*OUTERIPLEN*+*TUNNELLEN*)+*IPLEN*+*L4LEN* value must be less than the total DMA length for a packet. If this is not the case, the results are unpredictable.

The TCP/UDP checksum always continues to the last byte of the DMA data.

Note: For non-TSO, software still needs to calculate a full checksum for the TCP/UDP pseudo-header. This checksum of the pseudo-header should be placed in the packet data buffer at the appropriate offset for the checksum calculation.

7.2.5.3 SCTP CRC Offloading

For SCTP packets, a CRC32 checksum offload is provided.

Three fields in the transmit context descriptor set the context of the STCP checksum offloading feature:

- MACLEN
- IPLEN
- TUCMD.L4T

TUCMD.L4T=10b specifies that the packet type is SCTP, and that the 32-bit STCP CRC should be inserted at byte offset *MACLEN*+*IPLEN*+8.

IPLEN+*MACLEN* specifies the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum, the start of the STCP header. The minimal allowed value for this sum is 26. See MACLEN table in Section 7.2.3.2.3 for its relevant values.

The SCTP CRC calculation always continues to the last byte of the DMA data.

The SCTP total L3 payload size (*PAYLEN* - *IPLEN* - *MACLEN*) should be a multiple of four bytes (SCTP padding not supported).

Note: TSO is not available for SCTP packets.

Note: The CRC field of the SCTP header must be set by driver to zero prior to requesting a CRC calculation offload.

7.2.5.4 Checksum Supported per Packet Types

Table 7-45 lists which checksums are supported per packet type.

Note: TSO is not supported for packet types for which IP checksum and TCP/UDP checksum cannot be calculated.

Table 7-45. Checksums Supported by Packet Type

Packet Type	Hardware IP Checksum Calculation	Hardware TCP/UDP/SCTP Checksum Calculation
IPv4 packets	Yes	Yes
IPv6 packets	No (n/a)	Yes
IPv6 packet with next header options: • Hop-by-hop options • Destinations options • Routing (with len 0) • Routing (with len >0) • Fragment • Home option • Security option (AH/ESP)	 No (n/a) No (n/a) No (n/a) No (n/a) No (n/a) No (n/a) No (n/a)	 Yes Yes Yes No No No Yes
IPv4 tunnels: • IPv4 packet in an IPv4 tunnel • IPv6 packet in an IPv4 tunnel	 No No	 No Yes
IPv6 tunnels: • IPv4 packet in an IPv6 tunnel • IPv6 packet in an IPv6 tunnel	 No No	 No No
Packet is an IPv4 fragment	Yes	No
Packet has 802.3ac tag	Yes	Yes
IPv4 packet has IPsec header without IP options	Yes	Yes
Packet has TCP or UDP options	Yes	Yes
IP header's protocol field contains protocol # other than TCP, UDP, or SCTP	Yes	No
NVGRE: • Inner IPv4 packet • Inner IPv6 packet	 Yes (Inner and outer) Yes (outer)	 Yes Yes
VXLAN: • Inner IPv4 packet • Inner IPv6 packet	 Yes (Inner and outer) Yes (outer)	 Yes (inner only)[1] Yes (inner only)

1. The outer UDP header of VXLAN packets do not use a checksum.

7.2.6 Transmit Statistics

7.2.6.1 General Notes

- All Statistics registers are cleared on read. In addition, they stick at 0xFF...F when the maximum value is reached.
- Due to divergent paths between interrupt-generation and logging of relevant statistics counts, it might be possible to generate an interrupt to the system for a noteworthy event prior to the associated statistics count actually being increased. This is extremely unlikely due to expected delays associated with the system interrupt-collection and ISR delay, but might be an explanation for interrupt statistics values that do not quite make sense. Hardware guarantees that any event noteworthy of inclusion in a statistics count is reflected in the appropriate count within 1 μs; a small time-delay prior to reading the statistics might be required to avoid a potential mismatch between and interrupt and its cause.
- If TSO is enabled, statistics are collected after segmentation.
- All byte (octet) counters composed of 2 registers can be fetched by two consecutive 32-bit accesses while reading the Low 32-bit register first or a single 64-bit access.

7.2.6.2 Transmit Statistics Hierarchy

Figure 7-22 describes the relations between the packet flow and the different statistic counters.

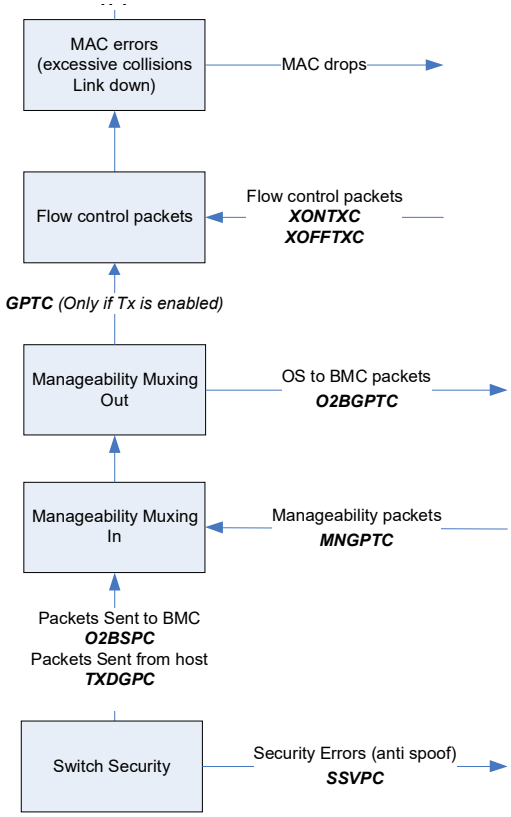


Figure 7-22. Transmit Flow Statistics

7.3 Interrupts

The X550 supports the following interrupt modes. Mapping of interrupts causes is different in each of these modes as described in this section.

- PCI legacy interrupts or MSI or MSI-X and only a single vector is allocated — Selected when GPIE.*MULTIPLE_MSIX* is set to 0b.
- MSI-X with multiple MSI-X vectors in non-IOV mode — Selected when GPIE.*MULTIPLE_MSIX* is set to 1b and GPIE.*VT_MODE* is set to 00b.
- MSI-X in IOV mode — Selected when GPIE.*MULTIPLE_MSIX* is set (as previously stated) and GPIE.*VT_MODE* DOES NOT equal 00b.

The following sections describe the interrupt registers and device functionality at all operation modes.

7.3.1 Interrupt Registers

7.3.1.1 Physical Function (PF) Registers

The PF interrupt logic consists of the registers listed in the Table 7-46 followed by their description:

Table 7-46. PF Interrupt Registers

Acronym	Complete Name
EICR	Extended Interrupt Cause register
EICS	Extended Interrupt Cause Set register (enables software to initiate interrupts)
EIMS	Extended Interrupt Mask Set/Read register
EIMC	Extended Interrupt Mask Clear register
EIAC	Extended Interrupt Auto Clear register (following interrupt assertion)
EIAM	Extended Interrupt Auto Mask register (auto set/clear of the EIMS)
EITR	Extended Interrupt Throttling register (throttling)
IVAR	Interrupt Vector Allocation Registers (described in Section 7.3.4)
IVAR_MISC	Miscellaneous Interrupt Vector Allocation Register (described in Section 7.3.4)

These registers are extended to 64 bits by an additional set of two registers. EICR has an additional two registers EICR(1)... EICR(2) and so on for the EICS, EIMS, EIMC, EIAM and EITR registers. The EIAC register is not extended to 64 bits as this extended interrupt causes are always auto cleared. Any reference to EICR... EIAM registers as well as any global interrupt settings in the GPIE register relates to their extended size of 64 bits.

The legacy EICR[15:0] mirror the content of EICR(1)[15:0]. In the same manner the lower 16 bits of EICS, EIMS, EIMC, EIAM mirror the lower 16 bits of EICS(1), EIMS(1), EIMC(1), EIAM(1). For more details on the use of these registers in the various interrupt modes (legacy, MSI, MSI-X) see Section 7.3.4.

7.3.1.2 Virtual Function (VF) Registers

The VF interrupt logic has the same set of interrupt registers while each of them has three entries for three interrupt causes. The names and functionality of these registers are the same as those of the PF with a prefix of VT as follows: VFEICR, VFEICS, VFEIMS, VFEIMC, VFEIAM, VFEITR. The interrupt causes are always auto cleared. Although each VF can generate up to three interrupts, only the first two registers are capable of interrupt throttling and are associated to VFEITR registers (see Section 7.3.4.3.2 for its proper usage). Each VF also has the mapping registers VFIVAR and VFIVAR_MISC. Note that any global interrupt setting by the GPIE register affect both interrupt settings of the PF as well as the VFs.

7.3.1.3 Extended Interrupt Cause (EICR) Registers

This register records the interrupt causes to provide software information on the interrupt source. Each time an interrupt cause happens, the corresponding interrupt bit is set in the EICR registers. An interrupt is generated each time one of the bits in these registers is set, and the corresponding interrupt is enabled via the EIMS registers. The possible interrupt causes are as follows:

- Each *RTxQ* bit represents the following events: Tx or Rx descriptor write back; Rx queue full and Rx descriptor queue minimum threshold.
- Interrupts can be throttled by ITR as configured in the EITR register.
- Mapping the Tx and Rx queues to EICR is done by the IVAR registers as described in Section 7.3.4. Each bit might represent an event on a single Tx or Rx queue or could represent multiple queues according to the IVAR setting. In the later case, software might not be able to distinguish between the interrupt causes other than checking all associated Tx and Rx queues.
- The *MULTIPLE_MSIX* = 1b setting is useful when multiple MSI-X vectors are assigned to the device. When the GPIE.*MULTIPLE_MSIX* bit is set, the *RTxQ* bits are associated with dedicated MSI-X vectors. Bit 0 is Tx/Rx interrupt associated with MSI-X vector 0 and bit 15 is Tx/Rx interrupt associated with MSI-X vector 15.

Writing a 1b to any bit in the register clears it. Writing a 0b to any bit has no effect. The EICR is also cleared on read if GPIE.*OCD* bit is cleared. When the GPIE.*OCD* bit is set, only bits 16...29 are cleared on read. The later setting is useful for MSI-X mode in which the Tx and Rx and possibly the timer interrupts do not share the same interrupt with the other causes. Bits in the register can be auto cleared depending on the EIAC register setting (see Section 7.3.1.6).

7.3.1.4 Extended Interrupt Cause Set (EICS) Register

This register enables software to initiate a hardware interrupt. Setting any bit on the EICS sets its corresponding bit in the EICR register while bits written to 0b have no impact. It then causes an interrupt assertion if enabled by the EIMS register. Setting any bit generates throttled interrupt depending on the GPIE.EIMEN setting: When the *EIMEN* bit is set, setting the EICS register causes an LLI interrupt; When the *EIMEN* bit is cleared, setting the EICS register causes an interrupt after the corresponding interrupt throttling timer expires.

Note: The *EIMEN* bit can be set high only when working in auto-mask mode (*EIAM* bit of the associated interrupt is set).

7.3.1.4.1 EICS Affect on RSC Functionality

Setting EICS bits causes interrupt assertion (if enabled). EICS settings have the same impact on RSC functionality as nominal operation:

- In ITR mode (GPIE.*EIMEN* = 0b), setting the EICS bits impact the RSC completion and interrupt assertion the same as any Rx packet. The functionality depends on the EICS setting schedule relative to the ITR intervals as described in Section 7.3.2.1.1.
- In LLI mode (GPIE.*EIMEN* = 1b), setting the *EICS* bits impact the RSC completion and interrupt assertion as follows:
  - Interrupt is asserted.
  - Concurrently, hardware triggers RSC completion in all Rx queues associated with the same interrupt.
  - Most likely these RSC(s) are completed to host memory after the interrupt is already asserted. In his case, it is guaranteed that an additional interrupt is asserted when the ITR expires.

7.3.1.5 Extended Interrupt Mask Set and Read (EIMS) Register, and Extended Interrupt Mask Clear (EIMC) Register

The Extended Interrupt Mask Set and Read (EIMS) register enables the interrupts in the EICR. When set to 1b, each bit in the EIMS register, enables its corresponding bit in the EICR. Software might enable each interrupt by setting bits in the EIMS register to 1b. Reading EIMS returns its value. Software might clear any bit in the EIMS register by setting its corresponding bit in the Extended Interrupt Mask Clear (EIMC) register. Reading the EIMC register does not return any meaningful data.

This independent mechanism of setting and clearing bits in the EIMS register saves the need for read modify write and also enables simple programming in multi-thread, multi-CPU core systems.

Note: The EICR register stores the interrupt events regardless of the state of the EIMS register.

7.3.1.6 Extended Interrupt Auto Clear Enable (EIAC) Register

Each bit in this register enables auto clearing of its corresponding bit in EICR following interrupt assertion. It is useful for Tx and Rx interrupt causes that have dedicated MSI-X vectors. When the Tx and Rx interrupt causes share an interrupt with the other or a timer interrupt, the relevant EIAC bits should not be set. Bits in the EICR register that are not enabled by auto clear, must be cleared by either writing a 1b to clear or a read to clear.

Note that there are no EIAC(1)...EIAC(2) registers. The hardware setting for interrupts 16...63 is always auto clear.

Note: Bits 29:16 should never be set to auto clear since they share the same MSI-X vector.

Writing to the EIAC register changes the setting of the entire register. In IOV mode, some of the bits in this register might affect VF functionality (VF-56...VF-63). It is recommended that software set the register in PF before VF's are enabled. Otherwise, a software semaphore might be required between the VF and the PF to avoid setting corruption.

7.3.1.7 Extended Interrupt Auto Mask Enable (EIAM) Register

Each bit in this register enables auto clearing and auto setting of its corresponding bit in the EIMS register as follows:

- Following a write of 1b to any bit in the EICS register (interrupt cause set), its corresponding bit in the EIMS register is auto set as well enabling its interrupt.
- A write to clear the EICR register clears its corresponding bits in the EIMS register masking further interrupts.
- A read to clear the EICR register, clears the *EIMS* bits (enabled by the EIAM) masking further interrupts. Note that if the GPIE.*OCD* bit is set, Tx and Rx interrupt causes are not cleared on read (bits 0:15 in the EICR). In this case, bits 0:15 in the EIMS are not cleared as well.
- In MSI-X mode the auto clear functionality can be driven by MSI-X vector assertion if GPIE.*EIAME* is set.

Note: Bits 29:16 should never be set to auto clear since they share the same MSI-X vector.

Writing to the EIAM register changes the setting of the entire register. In IOV mode, some of the bits in this register might affect VF functionality. It is recommended that software set the register in PF before VF's are enabled. Otherwise, a software semaphore might be required between the VF and the PF to avoid setting corruption.

If any of the *Auto Mask Enable* bits is set in the EIAM registers, the GPIE.*EIAME* bit must be set as well.

7.3.2 Interrupt Moderation

Interrupt rates can be tuned by the EITR register for reduced CPU utilization while minimizing CPU latency. In MSI or legacy interrupt modes, only EITR register 0 can be used. In MSI-X, non-IOV mode, the X550 includes 64 EITR registers 0...63 that are mapped to MSI-X vectors 0...63, respectively. In IOV mode, there are an additional 65 EITR registers that are mapped to the MSI-X vectors of the virtual functions. The mapping of MSI-X vectors to EITR registers are described in Section 7.3.1.3.

7.3.2.1 Time-Based Interrupt Throttling — ITR

Time-based interrupt throttling is useful to limit the maximum interrupt rate regardless of network traffic conditions. The ITR logic is targeted for Rx/Tx interrupts only. It is assumed that the software device driver does not moderate the timer, other and mail box (IOV mode) interrupts. In non-IOV mode, all 64 interrupts can be associated with ITR logic. In IOV mode, the ITR logic is shared between the PF and VFs as shown in Figure 7-23. The ITR mechanism is based on the following parameters:

- **ITR Interval** field in the EITR registers — The minimum inter-interrupt interval is specified in 2.048 µs units (at 1 Gb/s or 10 Gb/s link). When the ITR Interval equals zero, interrupt throttling is disabled and any event causes an immediate interrupt. The field is composed of nine bits enabling a range of 2.048 µs up to 1046.528 µs. These ITR interval times correspond to interrupt rates in the range of 488 K INT/sec to 955 INT/sec. When operating at 100 Mb/s link, the ITR interval is specified in 20.48 µs units.
  - Due to internal synchronization issues, the ITR interval can be shortened by up to 1 µs at 10 Gb/s or 1 Gb/s link and up to 10 µs at 100 Mb/s link when it is triggered by packet write back or interrupt enablement.

- **ITR Counter** partially exposed in the EITR registers — Down counter that is loaded by the ITR interval each time the associated interrupt is asserted.
  - The counter is decremented by one each 1.024 μs (at 1 Gb/s or 10 Gb/s link) and stops decrementing at zero. At 100 Mb/s link, the speed of the counter is decremented by one each 10.24 μs.
  - If an event happens before the counter is zero, it sets the EICR. The interrupt can be asserted only when the ITR time expires (counter is zero).
  - Else (no events during the entire ITR interval), the EICR register is not set and the interrupt is not asserted on ITR expiration. The next event sets the EICR bit and generates an immediate interrupt. See Section 7.3.2.1.1 for interrupt assertion when RSC is enabled.
  - Once the interrupt is asserted, the ITR counter is loaded by the ITR interval and the entire cycle re-starts. The next interrupt can be generated only after the ITR counter expires once again.

7.3.2.1.1 ITR Affect on RSC Functionality

Interrupt assertion is one of the causes for RSC completion (see Section 7.9.6). When RSC is enabled on specific Rx queues, the associated ITR interval with these queues must be enabled and must be larger (in time units) than RSC delay. The ITR is divided to the two time intervals that are defined by the ITR interval and RSC delay. RSC completion is triggered after the first interval completes and the interrupt is asserted when the second interval completes.

The *RSC Delay* field is defined in the GPIE registers. *RSC Delay* can have one of the following eight values: 4 μs, 8 μs, 12 μs... 32 μs.

- The first ITR interval equals ITR interval minus RSC delay. The internal ITR counter starts at ITR interval value and counts down until it reaches the RSC delay value. Therefore, the ITR interval must be set to a larger value than the RSC delay.
- The second ITR interval equals RSC delay. The internal ITR counter continues to count down until it reaches zero.
- RSC completion can take some time (usually in the range of a few micro seconds). This time is composed by completing triggering latency and completing process latency. These delays should be considered when tuning the RSC delay. The clock frequency of the RSC completion logic depends on the link speed. As a result, the completion delay can as high as ~0.8 μs at 10 Gb/s link and ~8 μs at 1 Gb/s link. The RSC completion logic might take additional ~50 ns at 10 Gb/s link and ~0.5 μs at 1 Gb/s link per RSC. In addition, there is the PCIe bus arbitration latency as well as system propagation latencies from the device up to host memory.
- Recommended RSC delay numbers are: 8 μs at 10 Gb/s link and 28 μs at 1 Gb/s link.
- RSC is not recommended when operating at 100 Mb/s link.

Following are cases of packet reception with respect to the ITR intervals:

- Packets are received and posted (including their status) to the Rx queue in the first ITR interval. In this case, RSC completion is triggered at the end of the first ITR interval and the interrupt is asserted at the second interval expiration.
- A packet (and its status) is received and posted to the Rx queue only after the first ITR interval has expired (either on the second interval or after the entire ITR interval has expired). In this case, RSC completion is triggered almost instantly (other than internal logic latencies). The interrupt is asserted at RSC delay time after the non-coalesced Rx status is queued to be posted to the host.
- Due to internal synchronization issues, the RSC delay can be shorten by up to 1 μs when it is triggered by packet write back.

7.3.2.2 Immediate Interrupt

The X550 might initiate an immediate interrupt when the receive descriptor ring is almost empty (Rx descriptors below a specific threshold). The threshold is defined by SRRCTL[n].*RDMTS* per Rx queue. This mechanism can protect against memory resources being used up during reception of a long burst of short packets.

7.3.3 TCP Timer Interrupt

7.3.3.1 Introduction

To implement TCP timers, software needs to take action periodically (every 10 ms). Today, the driver must rely on software-based timers, whose granularity can change from platform to platform. This software timer generates a software NIC interrupt, which then enables the driver to perform timer functions, avoiding cache thrash and enabling parallelism. The timer interval is system-specific.

It would be more accurate and more efficient for this periodic timer to be implemented in hardware. The driver would program a timeout value (usual value of 10 ms), and each time the timer expires, hardware sets a specific bit in the EICR register. When an interrupt occurs (due to normal interrupt moderation schemes), software reads the EICR register and discovers that it needs to process timer events.

The timeout should be programmable by the driver, and the driver should be able to disable the timer interrupt if it is not needed.

7.3.3.2 Description

A stand-alone, down-counter is implemented. An interrupt is issued each time the value of the counter is zero.

Software is responsible for setting an initial value for the timer in the *Duration* field. Kick-starting is done by writing a 1b to the *KickStart* bit.

Following kick starting, an internal counter is set to the value defined by the *Duration* field. Then the counter is decreased by one each ms. When the counter reaches zero, an interrupt is issued. The counter re-starts counting from its initial value if the *Loop* field is set.

7.3.4 Mapping of Interrupt Causes

The following sections describe legacy, MSI and MSI-X interrupt modes.

7.3.4.1 Legacy and MSI Interrupt Modes

In legacy and MSI modes, an interrupt cause is reflected by setting one of the bits in the EICR register, where each bit reflects one or more causes. All interrupt causes are mapped to a single interrupt signal: either legacy INTA/B or MSI. This section describes the mapping of interrupt causes (that is a specific Rx or Tx queue event or any other event) to bits in the EICR.

The TCP timer and all other interrupt causes are mapped directly to EICR[30:16]. Note that the IVAR_MISC register is not used in legacy and MSI modes.

Mapping the Tx and Rx queues to interrupt bits in the EICR register is programmed in the IVAR registers as shown in Figure 7-23. Each entry in the IVAR registers is composed of two fields that identify the associated bit in the EICR[15:0] register. Software might map multiple Tx and Rx queues to the same EICR bit.

- **INT_Alloc** — Defines one of the bits (0...15) in the EICR register that reflects the interrupt status indication.
- **INT_Alloc_val** — Valid bit for the this interrupt cause.

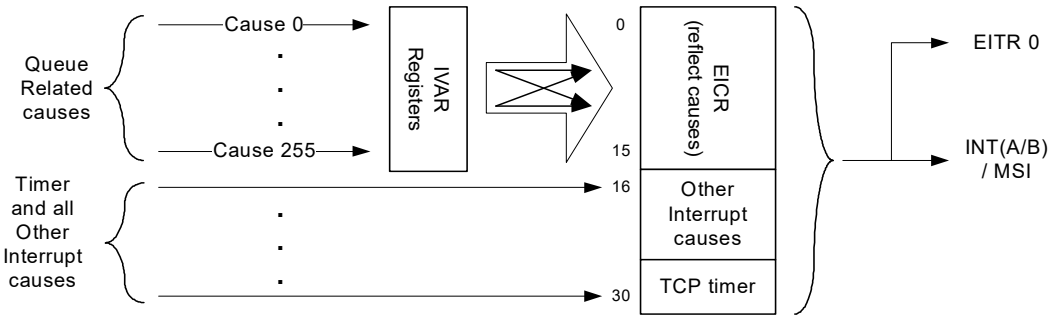


Figure 7-23. Cause Mapping in Legacy and MSI Modes

Mapping between the Tx and Rx queue to the IVAR registers is hard-wired as shown in the Figure 7-24.

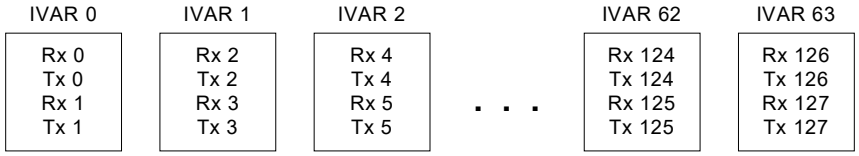


Figure 7-24. Rx and Tx Queue Mapping to IVAR Registers

7.3.4.2 MSI-X Mode in Non-IOV Mode

- MSI-X defines a separate optional extension to basic MSI functionality. Hardware indicates the number of requested MSI-X vectors in the table size in the MSI-X capability structure in the configuration space. The number of requested MSI-X vectors is loaded from NVM in the PCI_CNF2.*MSI_X_PF_N* field up to maximum of 64 MSI-X vectors. The operating system might allocate any number of MSI-X vectors to the device from a minimum of one up to the requested number of MSI-X vectors.
- Enables interrupts causes allocation to the assigned MSI-X vectors. Interrupt allocation is programmed by the IVAR registers and are described in this section.
- Each vector can use an independent address and data value as programmed directly by the operating system in the MSI-X vector table.
- Each MSI-X vector is associated to an EITR register with the same index (MSI-X 0 to EITR[0], MSI-X 1 to EITR[1],...).

For more information on MSI-X, refer to the PCI Local Bus Specification, Revision 3.0.

MSI-X vectors can be used for several purposes:

1. Dedicated MSI-X vectors per interrupt cause (avoids the need to read the interrupt cause register).
2. Load balancing by MSI-X vectors assignment to different CPUs.
3. Optimized interrupt moderation schemes per MSI-X vector using the EITR registers.

The MSI-X vectors are used for Tx and Rx interrupt causes as well as the other and timer interrupt causes. The remainder of this section describes the mapping of interrupt causes (such as a specific Rx or Tx queue event or any other event) to the interrupts registers and the MSI-X vectors.

The TCP timer and other events are reflected in EICR[30:16] the same as the legacy and MSI mode. It is then mapped to the MSI-X vectors by the IVAR_MISC register as shown in Figure 7-25. The IVAR_MISC register includes two entries for the timer interrupt and an additional entry for all the other causes. The structure of each entry is as follows:

- **INT_Alloc** — Defines the MSI-X vector (0...63) assigned to this interrupt cause.
- **INT_Alloc_val** — Valid bit for the this interrupt cause.

The Tx and Rx queues are associated to the IVAR0...IVAR63 the same as legacy and MSI mode shown in Figure 7-24. The Tx and Rx queues are mapped by the IVAR registers to EICR(1),...EICR(2) registers and MSI-X vectors 0...63 illustrated in Figure 7-25. The IVAR entries have the same structure as the IVAR_MISC register previously shown. Each bit in EICR(1...2) registers is associated to MSI-X vector 0...63 as follows:

- EICR(i).bit_num is associated to MSI-X vector (i x 32 + bit_num).
- The legacy EICR[15:0] mirror the content of EICR(1)[15:0]. In the same manner the lower 16 bits of EICS, EIMS, EIMC, EIAM mirror the lower 16 bits of EICS(1), EIMS(1), EIMC(1), EIAM(1). The use of these registers depends on the number of assigned MSI-X interrupts as follows:
- **16 Tx and Rx Interrupts** — When using up to 16 Tx and Rx interrupts, software might access the Tx and Rx interrupt bits in the legacy EICR, EICS,... registers.
- **More than 16 Tx and Rx Interrupts** — When using more than 16 Tx and Rx interrupts, software must use EICS(1)...EICS(2), EIMS(1)...EIMS(2),... In the later case, software should avoid modifying the lower 16 bits in the EICS, EIMS... registers when it accesses the higher bits of these registers as follows:
  - EICR, EICS, EIMS and EIMC — When software programs the higher 16 bits of these registers, it should set their lower 16 bits to zero's keeping the EICR(1), EICS(1), EIMS(1) and EIMC(1) unaffected.
  - EIAM — When software programs the higher 16 bits, it should keep the lower 16 bits at their previous setting so the EIAM(1) is unaffected.
  - EIAC — When software programs the higher 16 bits, it should set the lower 16 bits to one's.
- **Single MSI-X vector** — If the operating system allocates only a single MSI-X vector, the driver might use the non-MSI-X mapping method (setting the GPIE.Multiple_MSIX to 0b). In this case, the *INT_Alloc* field in the IVAR registers might define one of the lower 16 bits in the EICR register while using MSI-X vector 0. The IVAR_MISC should be programmed to MSI-X vector 0.

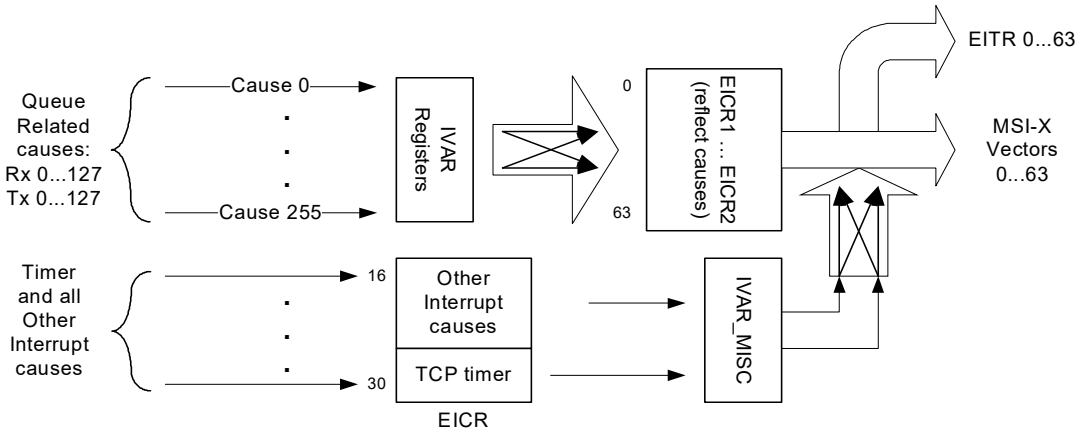


Figure 7-25. Cause Mapping in MSI-X Mode (non-IOV)

7.3.4.3 MSI-X Interrupts in IOV Mode

In IOV mode, interrupts must be implemented by MSI-X vectors. The X550 supports up to 64 virtual functions VF(0...63). Each VF can generate up to three MSI-X vectors. The number of requested MSI-X vectors per VF is loaded from NVM in the PCI_CNF2.*MSI_X_VF_N* field. It is reflected in the *Table Size* field in the PCIe MSI-X capability structure of the VF's. In addition, the PF requires its own interrupts. The number of requested MSI-X vectors is loaded from NVM in the PCI_CNF2.*MSI_X_PF_N* field up to maximum of 64 MSI-X vectors. It is reflected in the *Table Size* field in the PCIe MSI-X capability structure.

7.3.4.3.1 MSI-X Vectors Used by Physical Function (PF)

PF is responsible for the timer and other interrupt causes that include the VM to PF mailbox cause (explained in the virtualization sections). These events are reflected in EICR[30:16] and MSI-X vectors are the same as the non-IOV mode (illustrated in Figure 7-23). When there are less than the maximum possible active VF's, some of the Tx and Rx queues can be associated with the PF. These queues can be used for the sake of additional VM's serviced by the hypervisor (the same as VMDq mode) or some Kernel applications handled by the hypervisor. Tx and Rx mapping to the IVAR registers is shown in Figure 7-24 and mapping to the EICR, EICR(1),...EICR(2) registers as well as the MSI-X vectors is shown in Figure 7-25. See Section 7.3.4.3.3 for MSI-X vectors mapping of PF and VF's to the EITR registers.

Note: Software should not assign MSI-X vectors in the PF to Tx and Rx queues that are assigned to other VF's. In the case that VF's become active after the PF used the relevant Tx and Rx queues, it is the responsibility of the PF driver to clear all pending interrupts of the associated MSI-X vectors.

7.3.4.3.2 MSI-X Vectors Used by Virtual Functions (VFs)

Each of the VFs in IOV mode is allocated separate IVAR(s) called VFIVAR registers, and a separate IVAR_MISC called VFIVAR_MISC register. The VFIVAR_MISC maps the mailbox interrupt of the VF to its VFEICR and the MSI-X vector. The VFIVAR registers map the Tx and Rx interrupts of the VF to its VFEICR and the MSI-X vector. The mapping is similar to the mapping in the PF as shown in Figure 7-26 with the following comments:

- Each VF cannot have more than three MSI-X vectors. It has only three active bits in the VFEICR register while VFEICR.bit_num is associated with MSI-X vector (bit_num).
- The Tx and Rx interrupt can be mapped only to MSI-X 0 and MSI-X 1 (associated with VFEICR.0 and VFEICR.1).
- The mailbox interrupt can be mapped to any of the three MSI-X vectors. However, when all three of them are allocated by the operating system, software should map the mailbox to MSI-X 2 (associated with VFEICR.2). This rule should be kept since only VFEICR.0 and VFEICR.1 have ITR registers (VFEITR-0 and VFEITR-1).
- Association between the Tx and Rx queues and the VFIVAR registers is shown in the Figure 7-26, Figure 7-27 and Figure 7-28 for IOV-64 (64 VF's), IOV-32 and IOV-16. The colored boxes in the figures show the mapping between VF Rx and Tx queues to VFIVAR registers while the dashed boxes show the physical IVAR registers and the associated physical Rx and Tx queues.

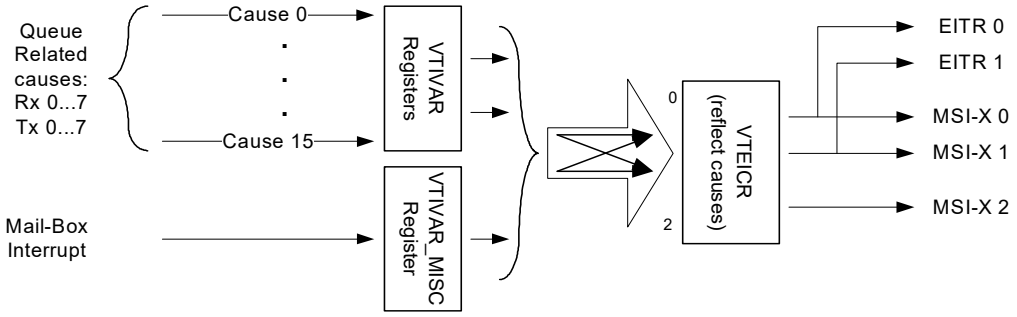


Figure 7-26. VF Interrupt Cause Mapping (MSI-X, IOV)

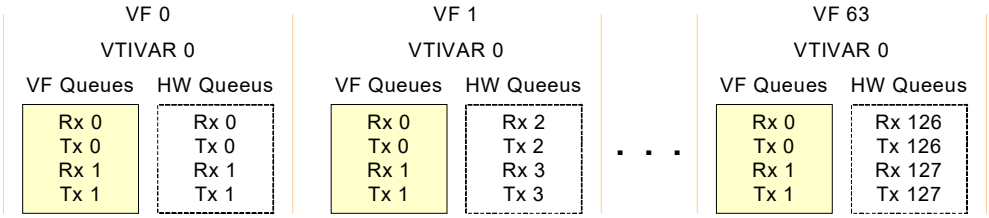


Figure 7-27. VF Mapping of Rx and Tx Queue to VFIVAR in 64 VF's Mode

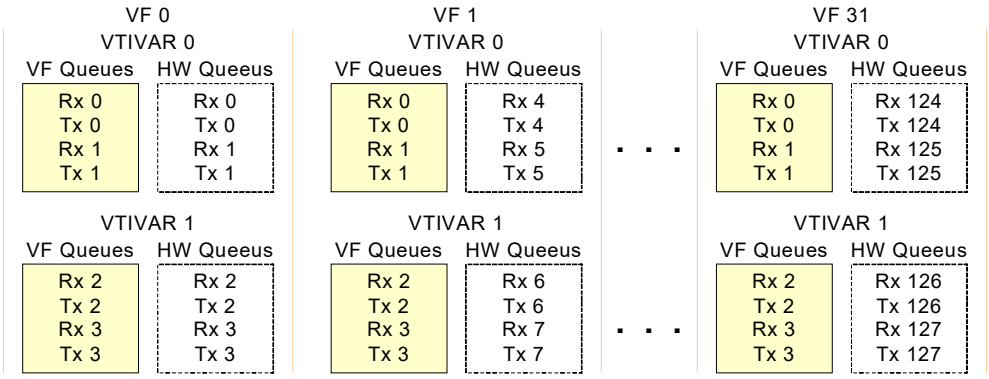


Figure 7-28. VF Mapping of Rx and Tx Queue to VFIVAR in 32 VF's Mode

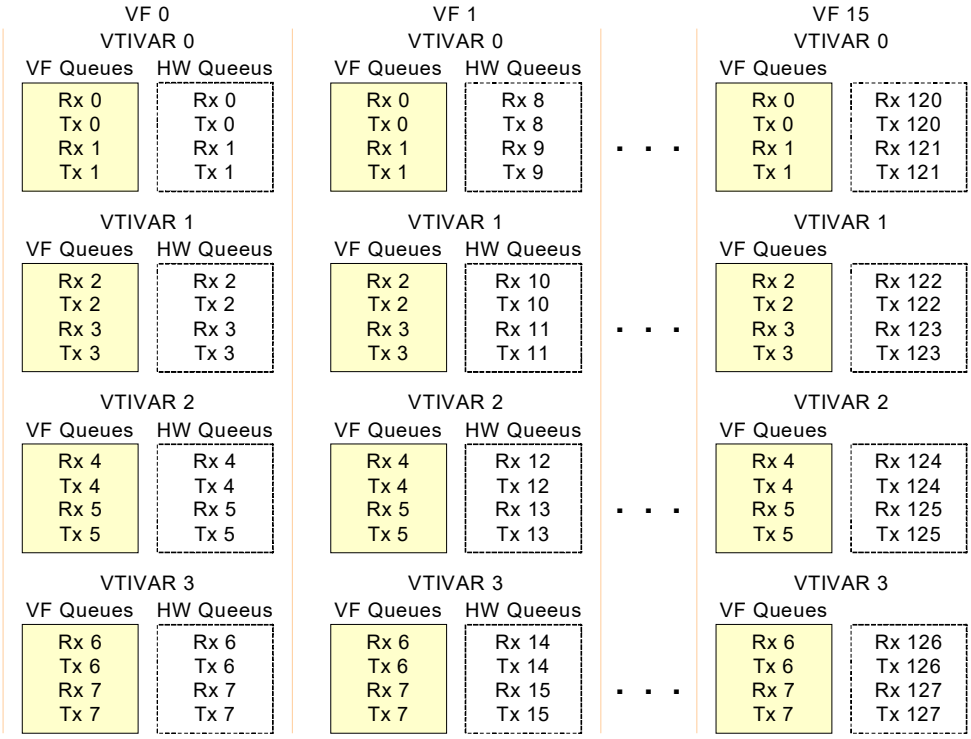
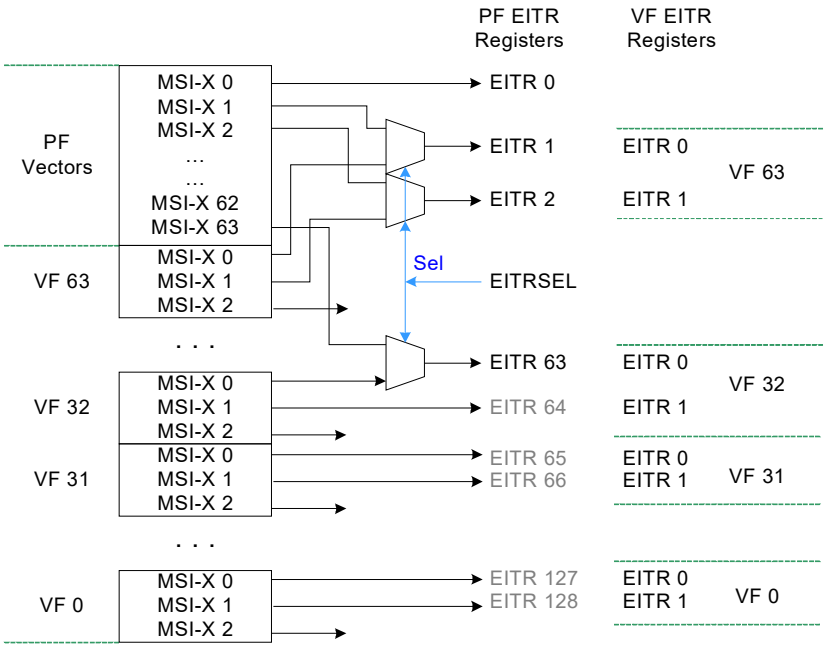


Figure 7-29. VF Mapping of Rx and Tx Queue to VFIVAR in 16 VF's Mode

7.3.4.3.3 MSI-X Vectors Mapping to EITR

EITR registers are aimed for Tx and Rx interrupt throttling. In IOV mode, the Tx and Rx queues might belong to either the PF or to the VF's. EITR(1...63) are multiplexed between the PF and the VF's as configured by the EITRSEL register. Figure 7-30 and Table 7-47 show the multiplexing logic and required software settings. For any active VF (starting from VF32 and above), software should program the matching bit in the EITRSEL to 1b. For any EITR that belongs to a VF, software should not map any interrupt causes in the PF to an MSI-X vector that is associated with the same EITR register.

Any RSCINT[n] register is associated with an MSI-X vector 'n'. As indicated above, the EITRSEL setting affects the MSI-X mapping. It also maps their associated RSCINT registers to either the PF or the VFs.



MSI-X 2 on each VF has no associated EITR register. It is useful for the mailbox interrupts that do not require interrupt moderation.

Figure 7-30. PF/VF MSI-X Vectors Mapping to EITR

Table 7-47. PF/VF MSI-X Vectors Mapping Table to EITR Registers

VM Active	EITRSEL.N Setting	MSI-X Routing to EITR
Non-IOV or VF(32...63) inactive	EITRSEL must be set to 0x0000	MSI-X(1...63) -> EITR(1...63)
VF(32) active	EITRSEL[0] must be set to 1b	VF(32) MSI-X(0) -> EITR(63)
VF(33) active	EITRSEL[1] must be set to 1b	VF(33) MSI-X(1) -> EITR(62) VF(33) MSI-X(0) -> EITR(61)
VF(34) active	EITRSEL[2] must be set to 1b	VF(34) MSI-X(1) -> EITR(60) VF(34) MSI-X(0) -> EITR(59)
...	...	...
VF(62) active	EITRSEL[30] must be set to 1b	VF(62) MSI-X(1) -> EITR(4) VF(62) MSI-X(0) -> EITR(3)
VF(63) active	EITRSEL[31] must be set to 1b	VF(63) MSI-X(1) -> EITR(2) VF(63) MSI-X(0) -> EITR(1)

7.4 802.1q VLAN Support

The X550 provides several specific mechanisms to support 802.1q VLANs:

- Optional adding (for transmits) and stripping (for receives) of IEEE 802.1q VLAN tags.
- Optional ability to filter packets belonging to certain 802.1q VLANs.

7.4.1 802.1q VLAN Packet Format

The following table compares an untagged 802.3 Ethernet packet with an 802.1q VLAN tagged packet:

802.3 Packet	#Octets	802.1q VLAN Packet	#Octets
DA	6	DA	6
SA	6	SA	6
Type/Length	2	802.1q Tag	4
Data	46-1500	Type/Length	2
CRC	4	Data	46-1500
		CRC*	4

Note: The CRC for the 802.1q tagged frame is re-computed, so that it covers the entire tagged frame including the 802.1q tag header. Also, maximum frame size for an 802.1q VLAN packet is 1522 octets as opposed to 1518 octets for a normal 802.3z Ethernet packet.

7.4.2 802.1q Tagged Frames

For 802.1q, the *Tag Header* field consists of four octets comprised of the Tag Protocol Identifier (TPID) and Tag Control Information (TCI); each taking two octets. The first 16 bits of the tag header makes up the TPID. It contains the protocol type that identifies the packet as a valid 802.1q tagged packet.

The two octets making up the TCI contain three fields as follows:

- User Priority (UP)
- Drop Eligible Indicator (DEI) — Should be set to 0b for transmits. For receives, the device has the capability to filter out packets that have this bit set. See the *DEIEN* and *DEI* bits in the VLNCTRL.
- VLAN Identifier (VID)

Octet 1		Octet 2
UP	DEI	VID

7.4.3 Transmitting and Receiving 802.1q Packets

Since the 802.1q tag is only four bytes, adding and stripping of tags can be done completely in software. In other words, for transmits, software inserts the tag into packet data before it builds the transmit descriptor list, and for receives, software strips the 4-byte tag from the packet data before delivering the packet to upper layer software. However, because adding and stripping of tags in software adds overhead for the host, the X550 has additional capabilities to add and strip tags in hardware. See Section 7.4.3.1 and Section 7.4.3.2.

7.4.3.1 Adding 802.1q Tags on Transmits

The inner VLAN header can be added by software in one of the following methods:

- The header is included in the transmit data buffers.
- Software might instruct the X550 to insert an 802.1q VLAN tag on a per-packet basis. If the *VLE* bit in the transmit descriptor is set to 1b, the X550 inserts a VLAN tag into the packet that it transmits over the wire. The Tag Protocol Identifier — TPID (VLAN Ether Type) field of the 802.1q tag comes from the DMATXCTL.*VT*, and the Tag Control Information (TCI) of the 802.1q tag comes from the VLAN field of the legacy transmit descriptor or the *VLAN Tag* field of the advanced data transmit descriptor.
- In IOV mode, the priority tag, DEI and VLAN ID can be taken from the *PFVMVIR* (see details in Section 7.7.11.2)

7.4.3.2 Stripping 802.1q Tags on Receives

Software might instruct the X550 to strip 802.1q VLAN tags from received packets. The policy whether to strip the VLAN tag is configurable per queue.

If the RXDCTL.*VME* bit for a given queue is set to 1b, and the incoming packet is an 802.1q VLAN packet (that is, its *Ethernet Type* field matched the VLNCTRL.*VET*), the X550 strips the 4-byte VLAN tag from the packet, and stores the TCI in the *VLAN Tag* field of the receive descriptor.

The X550 also sets the *VP* bit in the receive descriptor to indicate that the packet had a VLAN tag that was stripped. If the RXDCTL.*VME* bit is not set, the 802.1q packets can still be received if they pass the receive filter, but the VLAN tag is not stripped and the *VP* bit is not set. If PFQDE.*HIDE_VLAN* is set, the VLAN tag is stripped as above, but the *VLAN Tag* field and the STATUS.*VP* bit in the Rx descriptor are cleared.

7.4.4 802.1q VLAN Packet Filtering

VLAN filtering is enabled by setting the VLNCTRL.*VFE* bit to 1b. If enabled, hardware compares the *Type* field of the incoming packet to a 16-bit field in the VLAN Ether Type (VET) register. If the *VLAN Type* field in the incoming packet matches the VET register, the packet is compared against the VLAN Filter Table Array for acceptance.

The VLAN filter register VTFA, is a vector array composed of 4096 bits. The VLAN ID (VID) is a 12-bit field in the VLAN tag that is used as an index pointer to this vector. If the VID in a received packet points to an active bit (set to 1b), the packet matches the VLAN filter. The 4096-bit vector is comprised of 128 x 32 bit registers. The upper 7 bits of the VID selects one of the 128 registers while the lower 5 bits map the bit within the selected register.

Two other bits in the VLNCTRL register, *DEIEN* and *DEI*, are also used in conjunction with 802.1q VLAN filtering operations. *DEIEN* enables the comparison of the value of the *DEI* bit in the 802.1q packet to the Receive Control register *DEI* bit as acceptance criteria for the packet.

Note: The *VFE* bit does not effect whether the VLAN tag is stripped. It only effects whether the VLAN packet passes the receive filter.

7.4.5 Double VLAN and Single VLAN Support

The X550 supports a mode where all received and sent packets have at least one VLAN tag in addition to the regular tagging that might optionally be added. In this document, when a packet carries two VLAN headers, the first header is referred to as an outer VLAN and the second header as an inner VLAN header (as listed in the table that follows). This mode is used for systems where the near end switch adds the outer VLAN header containing switching information. This mode is enabled by the following configuration:

- This mode is activated by setting the DMATXCTL.*GDV* and the *EXTENDED_VLAN* bit in the CTRL_EXT register.
- The EtherType of the VLAN tag used for the additional VLAN is defined in the *VET_EXT* field in the EXVET and EXVET_T registers.

7.4.5.1 Cross Functionality with Manageability

The X550 does not provide any stripping or adding VLAN header(s) to manageability packets. Therefore, packets that are directed to/from the manageability controller should include the VLAN headers as part of the Rx/Tx data. The manageability controller should know if the X550 is set to double VLAN mode as well as the VLAN EtherType(s).

Table 7-48. Double VLAN Packet Format

MAC Address	Outer VLAN	Inner VLAN	L2 Payload	Ethernet CRC

7.4.5.2 Transmit Functionality

7.4.5.2.1 Transmit Functionality on the Outer VLAN Header

- A packet with a single VLAN header is assumed to have only the outer VLAN.
- The outer VLAN header must be added by software as part of the Tx data buffers.
- Hardware does not relate to the outer VLAN header other than the capability of skipping it for parsing inner fields.
- Hardware expects that any transmitted packet (see the disclaimer that follows) has at least the outer VLAN added by software. For any offload that hardware might provide in the transmit data path, hardware assumes that the outer VLAN is present. For those packets that an outer VLAN is not present, any offload that relates to inner fields to the EtherType might not be provided.

7.4.5.2.2 Transmit Functionality on the Inner VLAN Header

- Inner VLAN insertion is handled as described in Section 7.4.3.1.
- Hardware identifies and skips the VLAN header for parsing inner fields.
- DCB — The traffic class is dictated by the Tx queue.
- Pool Filtering — Destination pool(s) and anti-spoofing functionality is based on the Ethernet MAC Address and inner VLAN (if present) as described in Section 7.7.10.4 and Section 7.7.11.2.

7.4.5.3 Receive Handling of Packets with VLAN Header(s)

A received frame is analyzed for the existence of the outer and inner VLAN headers. The procedure is as follows:

```
Check the EtherType against the outer VLAN ID. If match then
{
    Check the next EtherType against the inner VLAN ID. If match
    {
        This is the double VLAN case. Process both outer and inner VLANs as described below
    }
    Else
    {
        Only an outer VLAN exists. Process the outer as described below. Assume no inner VLAN
    }
}
Else
{
    Check the EtherType against the inner VLAN ID. If match
    {
        This is the case of an inner VLAN only. Handle the frame as an unknown frame – device
        does not provide any offloads
    }
    Else
    {
        This is the case of no VLAN. Process the frame (ignore outer and inner VLAN processing)
    }
}
```

7.4.5.3.1 Receive Functionality on the Outer VLAN Header

- Hardware checks the EtherType of the outer VLAN header against the programmed value in the EXVET register. VLAN header presence is indicated in the STATUS.*VEXT* bit in the Rx descriptor.
- The outer VLAN header is posted as is to the receive data buffers.

7.4.5.3.2 Receive Functionality on the Inner VLAN Header

- Hardware checks the EtherType of the inner VLAN header against the programmed value in the VLNCTRL.*VET.* VLAN header presence is indicated in the STATUS.*VP* bit in the Rx descriptor.
- L2 packet filtering is based on the VLAN ID in the inner VLAN header.
- Pool Filtering — Destination pool(s) are defined by the Ethernet MAC Address and inner VLAN (if presence) as described in Section 7.7.10.3.
- Inner VLAN tag striping is handled as described in Section 7.4.3.2.

7.4.5.4 Packets with No VLAN Headers in Double VLAN Mode

There are some cases when packets might not carry any VLAN headers, even when extended VLAN is enabled. A few examples for packets that might not carry any VLAN header are: flow control and priority flow control, LACP, LLDP, GMRP, and optional 802.1x packets. When it is expected to transmit untagged packets by software in Double VLAN Mode the software must not enable VLAN anti-spoofing and VLAN validation nor transmit to receive switching.

7.4.5.4.1 Transmit Functionality

DCB — The TC in the Tx data path is directed by the Tx queue of the transmitted packet.

Transmit offload functionality — Software should not enable any offload functions.

7.4.5.4.2 Receive Functionality

Receive offload functionality — pool and queue are selected by the Ethernet MAC Address or ETQF/ ETQS registers. Filtering to host and manageability remains functional.

7.4.5.5 Packets with Two VLAN Headers Not in Double VLAN Mode

When the *EXTENDED_VLAN* bit in the CTRL_EXT register and *DMATXCTL.GDV* are not set, hardware expects that Rx and Tx packets might not carry a VLAN header or a single VLAN header. Hardware does not relate to the programming of the *VET_EXT* field in the EXVET register. Tx and Rx handling of packets with double VLAN headers is unexpected.

7.4.6 E-tag and VLAN

In some systems an additional external tag (E-tag) can be present before the VLAN. This section describes the support for VLANs in presence of external tags.This mode is used for systems where the device adds a tag to identify a subsystem (usually a VM) and the near end switch adds a tag indicating the destination subsystem. External tags may be present on part of the packets and missing in others.

7.4.6.1 Transmit Functionality

7.4.6.1.1 Transmit Functionality on the External Tag

The X550 supports insertion of an external tag from a per pool register (PFVMTIR - Section 8.2.2.22.30) and a VLAN tag from a per pool register (PFVMVIR - Section 8.2.2.22.14) or from the descriptor as indicated by the *VLE* bit. The following options are supported:

VLAN source/External Tag Source	Hardware Register - PFVMTIR/PFVMVIR	Embedded in Packet
Hardware register - PFVMVIR	Supported	Not supported
Descriptor	Supported	Not supported
Embedded in Packet	Supported	Supported

After the tags are inserted in the packet, the X550 uses the VLAN tag and the EtherType as part of the forwarding decision as described in Section 7.7.10.4. In a packet with at most one external tag and one VLAN, the VLAN and EtherType are identified correctly.

7.4.6.2 Receive Handling of Packets with External Tags

External tags can be extracted from the packet according to the PFQDE.*STRIP_TAG* field (Section 8.2.2.22.4). Inner tag extraction is handled as described above (Section 7.4.3.2).

7.4.6.2.1 Packet Priority in Presence of External Tags

The user priority used to define the traffic class of a packet is always taken from the inner VLAN

7.4.6.3 Cross Functionality with Manageability

The X550 does not provide any stripping or adding VLAN header(s) to manageability packets. Therefore, packets that are directed to/from the manageability controller should include the L2 headers as part of the Rx/Tx data. The manageability controller should know if it is expected to add/receive an external tag as part of the packet.

7.4.6.4 Packet User Priority (802.1P) Bits Handling

The user priority bits may be used by the device for multiple purposes:

1. Defining the traffic class to which the traffic is associated (Section 7.6.3.1).
2. Defining which packets get a Timestamp in Buffer (Section 7.1.6.2).
3. Defining if a packet matches and EtherType filter (Section 7.1.3.3).

For all these purposes, the *UP* bits are taken from the outermost tag with *UP* bits (E-tag or VLAN), unless the VLNCTRL.*UP_FIRST_TAG_EN* bit is cleared, in which case, it is always taken from the inner VLAN.

The table below summarize the tag from which the user priority is extracted from the packets.

Table 7-49. UP Extraction Rules

Packet Type	VLNCTRL.*UP_FIRST_TAG_EN* = 1b	VLNCTRL.*UP_FIRST_TAG_EN* = 0b
Un-tagged Packet	User priority = 0	User priority = 0
Packet with a single VLAN	The user priority field from the VLAN header.	**Single VLAN mode:** The user priority field from the VLAN header. **Double VLAN mode:** User priority = 0
Packet with a VLAN and an outer tag (outer VLAN or E-tag)	The user priority field from the outer VLAN header	The user priority field from the inner VLAN header.
Packet with E-tag only	The user priority field from the E-tag	User priority = 0

7.5 TLP Processing Hints (TPH)

The X550 supports the TPH capability defined in the PCI Express specification. It does not support Extended TPH requests.

On the PCIe link existence of a TLP Process Hint (TPH) is indicated by setting the *TH* bit in the TLP header. Using the PCIe TLP Steering Tag (ST) and Processing Hints (PH) fields, the X550 can provide hints to the root complex about the destination (socket ID) and about data access patterns (locality in Cache), when executing DMA memory writes or read operations. Supply of TLP Processing Hints facilitates optimized processing of transactions that target Memory Space.

To enable TPH usage:

1. For a given function, the *TPH Requester Enable* bit in the PCIe configuration TPH Requester Control Register should be set.
2. Appropriate *TPH Enable* bits in TPH_RXCTRL or TPH_TXCTRL registers should be set.
3. Processing hints should be programmed in the TPH_CTRL.*DESC_PH* and TPH_CTRL.*DATA_PH* Processing hints (PH) fields.
4. Steering information should be programed in the CPUID fields in the TPH_RXCTRL and TPH_TXCTRL registers.

The Processing hints (PH) and Steering Tags (ST) are set according to the characteristics of the traffic as described in Table 7-50.

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

Note: Per queue, the TPH features are exclusive. Software can enable the TPH feature for a given queue.

7.5.1 Steering Tag and Processing Hint Programming

Table 7-50 describes how the Steering tag (socket ID) and Processing hints are generated and how TPH operation is enabled for different types of DMA traffic.

Table 7-50. Steering Tag and Processing Hint Programming

Traffic type	ST Programming	PH value	Enable
Transmit descriptor write back or head write back	TPH_TXCTRL.*CPUID*[1]	TPH_CTRL.*DESC_PH*[2]	*Tx Descriptor Writeback TPH EN* field in TPH_TXCTRL.
Receive data buffers write	TPH_RXCTRL.*CPUID*[1]	TPH_CTRL.*DATA_PH*[3]	*Rx Header TPH EN or Rx Payload TPH EN* fields in TPH_RXCTRL.
Receive descriptor writeback	TPH_RXCTRL.*CPUID*[1]	TPH_CTRL.*DESC_PH*[2]	*Rx Descriptor Writeback TPH EN* field in TPH_RXCTRL.
Transmit descriptor fetch	TPH_TXCTRL.*CPUID*[4]	TPH_CTRL.*DESC_PH*[2]	*Tx Descriptor Fetch TPH EN* field in TPH_TXCTRL.
Receive descriptor fetch	TPH_RXCTRL.*CPUID*[2]	TPH_CTRL.*DESC_PH*[2]	*Rx Descriptor fetch TPH EN* field in TPH_RXCTRL.
Transmit packet read	TPH_TXCTRL.*CPUID*[2]	TPH_CTRL.*DATA_PH*[3]	*Tx Packet TPH EN field in* TPH_TXCTRL.

1. The driver should always set bits [7:3] to zero and place Socket ID in bits [2:0].
2. Default is 00b (Bidirectional data structure).
3. Default is 10b (Target).
4. The hints are always zero.

7.6 Data Center Bridging (DCB)

See Section 4.6.11 for the DCB configuration sequence.

7.6.1 Overview

DCB is a set of features that improve the capability of Ethernet to handle multiple traffic types (such as LAN, storage, IPC) by answering the various needs of those types. DCB enables multiple traffic types that have different requirements of packet delivery, bandwidth allocation and delay. Each traffic type can have one or several user priorities, or Traffic Classes (TCs). For example, IPC might have a high priority class for synchronization messages between servers and lower priority traffic class for bulk traffic exchange between servers. Most of the DCB functions impact the transmit traffic generated from the end node to the network (traffic generation). The receive data path needs be compliant with the requirements of DCB and provide the required functions as a traffic termination point.

DCB system requirements include:

- **Bandwidth grouping** — For effective multiplexing that simulates a separate link for the separate types of traffic, DCB requires that traffic types be recognized as groups in the bandwidth and priority handling by nodes in the network. Traffic types are associated to Bandwidth Groups (BWGs). The system needs to be able to allocate bandwidth to the BWGs in a way that emulates that group being on its own separate link.
- **Bandwidth fairness** — DCB multiplexing functions (transmit) and de-multiplexing functions (receive) need to guarantee minimum allocation of bandwidth to traffic types and traffic classes. Fairness between groups comes first, then fairness between TCs. If system resources (such as PCIe bandwidth) limit total throughput, then the available bandwidth should be distributed among consumers proportionally to their allocations.
- **Latency of operation** — DCB multiplexing and de-multiplexing functions need to allow minimum latency for some TCs. Arbitration mechanisms, packet buffers, descriptor queues and flow control algorithm need to be defined and designed to allow this. The best example is the control/sync traffic in IPC. The expectation for end-to-end IPC control is measured in the low 10's of µs for the X550 and is expected to drop to a singe digit µs later. Some elements in multimedia traffic also bear similar requirements. Although some of the end-to-end delays can be quite long, the individual contribution of the arbitration in each node must be kept to a minimal. End-to-end budgets do not comprehend large delays within transmission nodes.
- **No-drop behavior and network congestion management** — The end node must be able to guarantee no-drop behavior for some TCs or some packets within TCs. As a termination point in receive, it is the end node’s responsibility to properly control traffic coming from the network to achieve this end. For traffic generation in transmit, the end station must be able to positively respond to flow control from the network as the must have tool to prevent packet drop. It also needs to participate in network congestion management.
- **Compatibility with existing systems** — The DCB implementation needs to be usable by IT using known configurations and parameters, unless new ones are made expressly available. For example, DCB implementation cannot assume new knowledge regarding bandwidth allocation of traffic types that do not have known bandwidth requirements.

The layer 2 features of DCB implemented in the X550 are:

- **Multi-class priority arbitration and scheduling** — The X550 implements an arbitration mechanism on its transmit data path. The arbitration mechanism allocates bandwidth between TC in BWGs and between Virtual Machines (VMs) or Virtual Functions (VFs) in a virtualization environment. The BWGs can be used to control bandwidth and priority allocated to traffic types. Typically BWGs should be used to represent traffic types. TC arbitration allows control of bandwidth and priority control within BWGs as well as within the entire link bandwidth. The arbitration is designed to respect the bandwidth allocations to BWGs. The priority allocation allows minimization of delay for specific TCs. In the X550, TCs and user priorities are processed on a packet-by-packet basis based on the 802.1p identifier in the 802.1Q-tag.
- **Class-based flow control (PFC — Priority Flow Control)** — Class-based flow control functionality is similar to the IEEE802.3X link flow control. It is applied separately to the different TCs.
  - — Transmit response to class-based flow control from the ingress switch it is connected to.
  - — Receive class-based flow control commands to the switch in response to packet buffers filling status.
- **DMA queuing per traffic type** — Implementation of the DCB transmit, minimization of software processing and delays require implementation of separate DMA queues for the different traffic types. The X550 implements 128 descriptor queue in transmit and 128 descriptor queues in receive.
- **Multiple Buffers** — The X550 implements separate transmit and receive packet buffers per TC.
- **Rate-limiter per Tx queue** — Limiting the transmit data rate for each Tx queue.

Latency requirements:

Quantitative latency requirements are defined for a single 64-byte packet at the highest priority traffic class. Latency is defined separately for transmit and receive:

- **Transmit latency** — Measured from a tail update until the packet is transmitted on the wire. It is assumed that a single packet is submitted for this TC and its latency is then measured in the presence of traffic belonging to other TCs.
- **Receive latency** — Measured from packet reception from the wire and until the descriptor is updated on PCIe.

Figure 7-31 shows the latency requirements as previously defined.

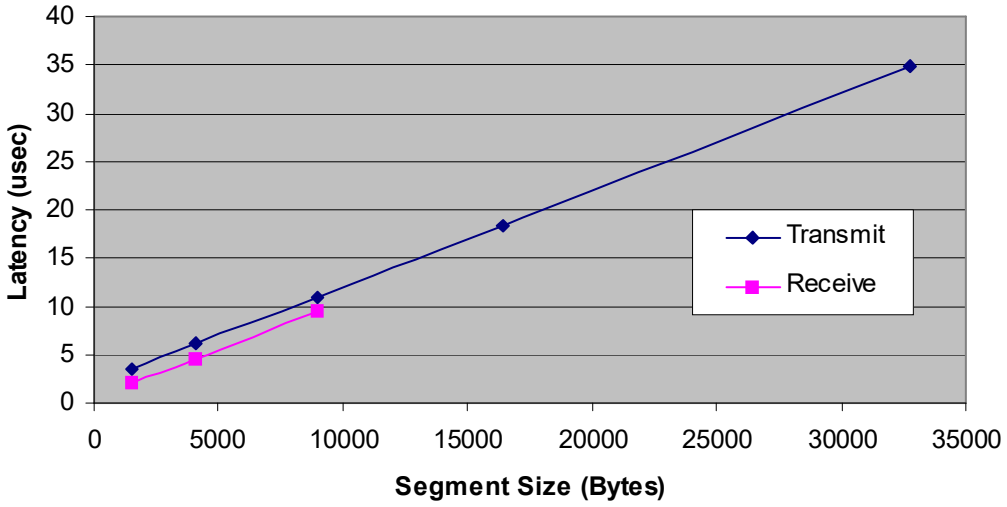


Figure 7-31. Latency Requirements

Note: In DCB mode, it is assumed all traffic is tagged (contains a VLAN header) — except for Layer2 frames with special Ethernet MAC Addresses that goes un-tagged. GMRP frames (special Ethernet MAC Addresses starting with 0x0180C20000) must, however, go tagged. Un-tagged packets must be delivered to the host and are assumed to belong to User Priority 0.

7.6.2 Transmit-Side Capabilities

When configured for DCB mode or when using the Tx rate-limiting functionality, the X550 software driver should only use advanced transmit descriptors. Refer to Section 7.2.3.2.3.

7.6.2.1 Transmit Rate Scheduler (RS)

7.6.2.1.1 Basic Rate Control Operation

Rate control is defined in terms of maximum payload rate, and not in terms of maximum packet rate. This means that each time a rate controlled packet is sent, the next time a new packet can be sent out of the same rate controlled queue is relative to the packet size of the last packet sent. The minimum spacing in time between two starts of packets sent from the same rate controlled queue is recalculated in hardware on every packet again, by using the following formula:

MIFS = PL x RF

Where:

- Packet Length (PL), is the Layer2 length (such as without preamble and IPG) in bytes of the previous packet sent out of that rate controller. It is an integer ranging from 64 to 9 K (at least 14-bits).

- RF = 10 Gb/s / target rate (rate factor) is the ratio between the nominal link rate and the target maximum rate to achieve for that rate-controlled queue. It is dynamically updated by software via the RTTQCNRC register or periodically according to the self-recovery algorithm run in software. It is a decimal number ranging from 1 to 1,000 (10 Mb/s minimum target rate). For example, at least 10 bits before the hexadecimal point and 14 bits after as required for the maximum packet length by which it is multiplied. For links at 1 Gb/s, the rate factor must be configured relatively to the link speed, replacing 10 Gb/s by 1 Gb/s in the above formula.
- Minimum Inter Frame Space (MIFS) is the minimum delay in bytes units, between the starting of two Ethernet frames issued from the same rate-controlled queue. It is an integer ranging from 76 to 9,216,012 (at least 24 bits). In spite of the 8-byte resolution provided at the internal data path, the byte-level resolution is required here to maintain an acceptable rate resolution (at 1% level) for the small packets case and high rates.

Note: It might be that a pipeline implementation causes the MIFS calculated on a transmitted packet to be enforced only on the subsequent transmitted packet.

Time Stamps — A rate-scheduling table contains the accumulated interval MIFS, for each rate-controlled descriptor queue separately, and is stored as an absolute Time Stamp (TS) relative to an internal free running timer. The TS value points to the time in the future at which a next data read request can be sent for that queue. Whenever updating a Timestamp:

TimeStamp(new) = TimeStamp(old) + MIFS

When a descriptor queue starts to be rate controlled, the first interval MIFS value is equal to 0 (TS equal to the current timer value) — without taking into account the last packet sent prior to rate control. When the TS value stored becomes equal to or smaller than the current free running timer value, it means that the switch is on and that the queue starts accumulating compensation times from the past (referred as a negative TS). When the TS value stored is strictly greater than the current free running timer value, it means that the switch is off (referred as a positive TS).

(CurrentTime) < TimeStamp <--> switch is off
(CurrentTime) >= TimeStamp <--> switch is on

MMW — The ability to accumulate negative compensation times that saturates to a Max Memory Window (MMW) time backward. MMW size is configured per TC via the *MMW_SIZE* field of the RTTQCNRM register, and is expressed in 1 KB units of payload, ranging from 0 up to 2 KB units (at least 11 bits). The *MMW_SIZE* configured in KB units of payload has to be converted in time interval *MMW_TIME* expressed in KB, before a new time stamp is checked for saturation. It is computed for each queue according to its associated Rate Factor (RF) using the following formula:

MMW_TIME = MMW_SIZE x RF

Note: *MMW_TIME* is rounded by default to a 1 KB precision level and must be at least 31 bits long. Hence, the time stamp byte-level values stored must be at least 32 bits long for properly handling the wrap-around case. 29 bits are required for the internal free running timer clocked once every 8 bytes.

Whenever updating a time stamp verify:

TimeStamp(old) + MIFS >= (CurrentTime) — MMW_TIME

and then the time stamp is updated according to the non-saturated formula:

TimeStamp(new) = TimeStamp(old) + MIFS

Otherwise, enforced saturation by assigning:

TimeStamp(new) = (CurrentTime) — MMW_TIME + MIFS

Note: Non-null MMW introduces some flexibility in the way controlled rates are enforced. It is required to avoid overall throughput losses and unfairness caused by rate-controlled packets over-delayed, consequently to packets inserted in between. Between two rate-limited packets spaced by at least the MIFS interval, non-rate-limited packets, or rate-limited packets from other rate-controlled queues, might be inserted. If a rate controlled packet has been delayed by more time than it was required for rate control (because of arbitration between VMs or TCs), the next MIFS accumulates from the last time the queue was switched on by the QCN rate scheduling table — and not from the current time. Refer to Figure 7-32 for visualizing the effect of MMW.

MMW_SIZE set to 0b must be supported as well.

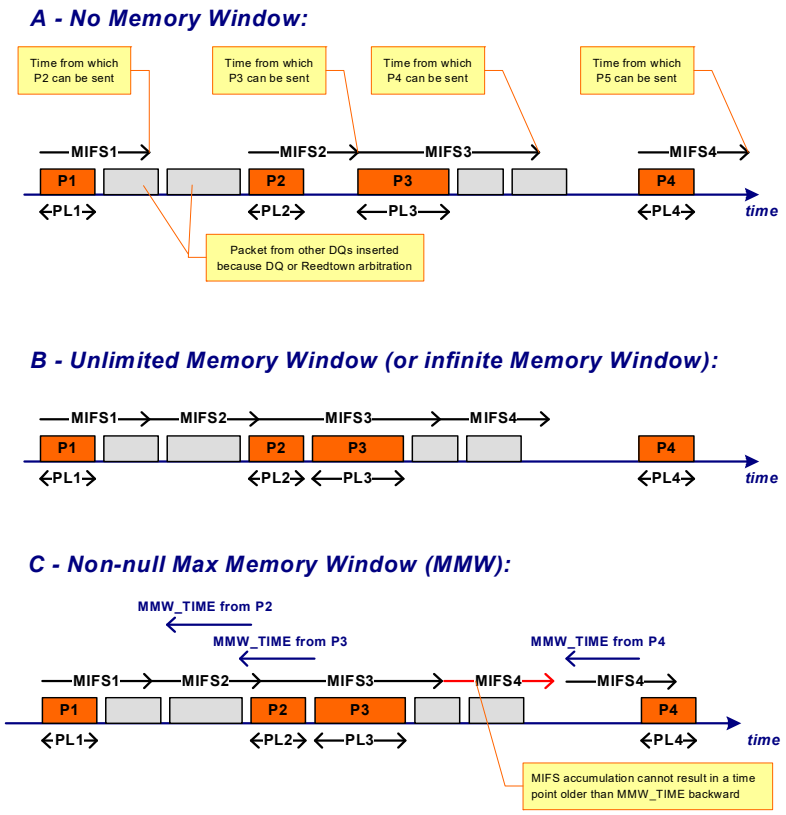


Figure 7-32. Minimum Inter-Frame Spacing for Rate-Controlled Frames (in Orange)

7.6.2.2 User Priority to Traffic Class Mapping

DCB-enabled software is responsible for classifying any Tx packet into one of the eight 802.1p user priorities, and to assure it is tagged accordingly by either software or hardware. The driver dispatches classified Tx traffic into the Tx queues attached to the proper TC, according to a UP-to-TC Tx mapping policy decided by the IT manager.

Caution: When translating XON/XOFF priority flow control commands defined per UP into commands to the Tx packet buffers, the X550 is required to use the same UP-to-TC Tx mapping table that software is using. The RTTUP2TC register must be configured by software accordingly. Refer to Section 3.7.4.1.3 for details on priority flow control.

7.6.2.3 VM-Weighted Round-Robin Arbiters

The X550 implements VM-weighted arbiter(s) for virtualized environments and according to the following case:

- If DCB is enabled, there is one such arbiter per TC, arbitrating between the descriptor queues attached to the TC (one queue per VF). Bandwidth allocation to VMs is enforced at the descriptor plane, per each TC separately. The VM arbiter instantiated for each TC is aimed to elect the next queue for which a data read request is sent in case the TC is elected for transmission by the next level arbiter. For example, the TC weighted strict priority descriptor plane arbiter.
- If DCB is disabled, there is one single VM weighted arbiter, arbitrating between pools of descriptor queues, where a pool is formed by the queues attached to the same VF. Bandwidth allocation to VMs is enforced at the descriptor plane, between the pools, where queues within a pool are served on a frame-by-frame round-robin manner.

Refer to the different arbitration schemes where virtualization is enabled, as described in Section 7.7.7.

Note: In this section, VM is considered a generic term used to refer to the arbitrated entity, whether it is a Tx descriptor queue within the TC or whether it is a pool of Tx descriptor queues. In the later case, pool parameters are allocated only to the lowest indexed queue within the pool, taken as the representation of the entire pool.

7.6.2.3.1 Definition and Description of Parameters

Credits: Credits regulate the bandwidth allocated to VMs. As part of the Weighted Round Robin (WRR) algorithm, each VM has pre-allocated credits. They are decremented upon each transmission and replenished cyclically. The ratio between the credits of the VMs represents the relative bandwidth percentage allocated to each VM (within the TC for the DCB enabled case). The X550 effectively maintains one table that represents these ratios. Note that credits can get negative values down to the maximum sized frame allowed on the TC/pool.

Note: The absolute value of the credits has no direct bearing on the allocation of bandwidth. The ratio between the credits does. However, since credits can accumulate only up to twice the credit refills, the refills should be allocated as low as possible but must be set greater than the maximum sized frame allowed on the TC or on the pool.

WRR: The algorithm implemented in the X550 for VM arbitration.

Table 7-51 (T1) defines the VMs and their bandwidth allocation. The following elements are defined in this table:

Table 7-51. Bandwidth Allocation to VMs

T1: VM Bandwidth Allocation			
VM_N	VM Refill	VM Max Credits	VM Min Credits
0	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
1	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
2	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
3	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
...			
15	MSS:1,024 KB	2xMSS:2,048 KB	-MSS

Notes:
1. Due to a pipelined implementation, the VM credits range is enlarged by one MSS, beyond negative limits.
2. All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).

VM: Configuration — The unique Tx descriptor queue attached to a VF within a TC, or the pool of Tx descriptor queues attached to the same VF.

VM Credit Refill: Configuration — The X550's WRR algorithm implement credit refill as the technique for percentage allocation to VMs. The credits refill are added to each VM credit count on each completion of the full round of the algorithm (after all the VMs had their chance to send what they had in store or at least one frame).

The X550's driver needs to calculate the VM credit refill to match the percent allocated through management (such as in the MIB). Since the WRR arbitration is self timed, the ratio between the credits refill is the only defining parameter for the VMs. However, the refills must be greater or equal to the maximum sized frame allowed on the TC or on the pool to guaranty transmission of at least one frame on each recycling round. The X550 allows a value of 1.5 KB to 1,024 KB for a dynamic range of x1000.

VM Maximum Accumulated Credits: Deducted from Refill Configuration — To prevent the use of stale credits, the number of credits each VM can accumulate upon refill is limited. The credits for each VM can only reach twice their refill. The maximum range for the credits is thus -9.5 KB to 2,048 KB, assuming negative credits can accumulate up to a maximum sized frame (9.5 KB if jumbo frames are allowed), and where positive credits can accumulate up to twice the maximum credit refill.

VM Credits: Run time parameter — VM credits is a running counter for each VM. It holds the number of available credits in 64-byte units. The algorithm runs sequentially between the VMs and enables transmission for those VMs that have enough pending credits (their credit number is greater than zero).

Table 7-52. Registers Allocation for Tx VM Arbiters

Attribute	Tx VM Arbiter
VM Control registers	RTTDT1C
VMC Status registers	RTTDT1S
VM credit refill	CRQ
VM credits	CCC

7.6.2.3.2 WRR Arbiter Algorithm

Note: This feature should be combined with configuring at least twice the sum of T1[VM]. Refills over a TC to its corresponding T2.[TC]MaxCredits. See Section 4.6.11.4.1 for details.

Round Robin — The round-robin aspect of the VM WRR arbiter resides in the fact that once a VM has been granted for a data read request, the next VMs are checked in a cyclic round-robin order, even if the granted VM still has credits for another data read request.

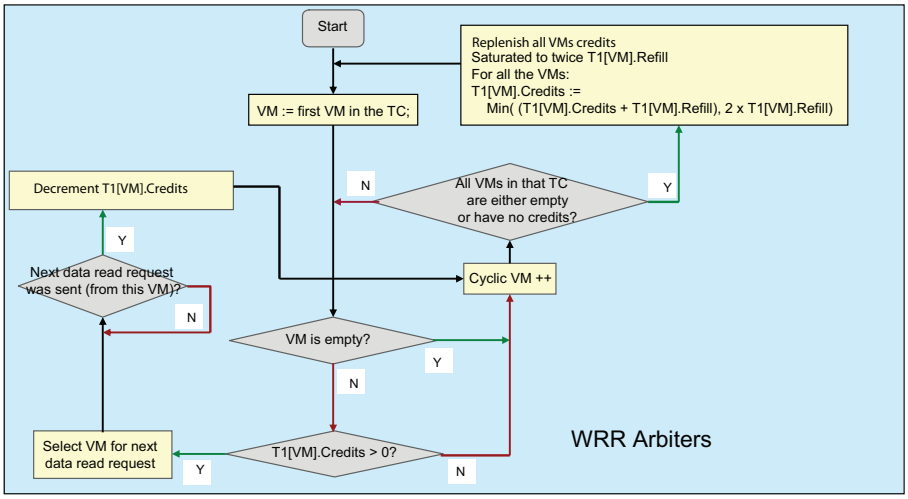


Figure 7-33. Tx VM WRR Arbiters Operation

7.6.2.4 Tx TC Weighted Strict Priority Arbiters

In DCB, multiple traffic types are essentially multiplexed over the Ethernet 10 Gb/s network. There is a need to allow different behavior to different traffic flows as they pass-through multiple Ethernet switches and links. For example, for LAN, SAN and IPC connections are consolidated on a single Ethernet link. Each traffic type (BWG) is guaranteed the bandwidth it has been allocated and is prevented from usurping bandwidth from other types. However, if a BWG does not use its bandwidth, that bandwidth is made available to the other BWGs. The same holds for TCs within a BWG. If allocated some bandwidth, TCs are guaranteed to have it, and if unused, that bandwidth can be used by the other TCs within the BWG. Information regarding bandwidth allocation for some TCs might not be available. In the case of LAN, the entire allocation of bandwidth within the LAN link is typically undefined in today's networks. The arbitration scheme includes Group Strict Priorities (GSP) to cover for that. TCs for which the *GSP* bit is set are limited by the total throughput allocated to their BWG rather than to TC allocation.

Link bandwidth is divided among the BWGs for guaranteed minimum behavior. For example: LAN: 4 Gb/s, SAN: 4 Gb/s, IPC: 2 Gb/s. The X550 supports two types of bandwidth allocation within BWGs. TCs can be either allocated bandwidth or be used as in strict priority. If a TC does not use all of its allocated bandwidth, that bandwidth is recycled to other TCs in the BWG.

The X550 implements two replications of the weighted TC arbiter:

- One in the descriptor plane, arbitrating between the different descriptor queues, deciding which queue is serviced next. It is aimed to enforce the TC bandwidth allocation and prioritization scheme for a case when PCIe bandwidth is smaller than the link bandwidth.

- A second in the packet plane, at the output of the packet buffers, deciding which packet to transmit next. It is aimed to enforce the TC bandwidth allocation and prioritization scheme for a case when PCIe bandwidth is greater than the link bandwidth.

The condition for entry into the bandwidth allocation algorithm sequence differs for the descriptor and data arbiters:

- The descriptor arbiter queries whether there is at least one queue attached to a TC that is not empty, not switched off by the rate scheduler, with positive VM weighted arbiter credits (when relevant), and for which the destined packet buffer has room for the worst case maximum sized frame. This last condition is controlled by RTTDCS.*BPBFSM*.
- The packet arbiter queries whether the packet buffer has a packet to send and whether it is not stalled by priority flow control.

7.6.2.4.1 Definition and Description of Parameters

User Priority (UP): There are eight traffic priorities, determined by 802.1p tag bits on an Ethernet link. The *Q-Tag* field holds UP's. Per 802.1p, Priority #7 is the highest priority. User priorities are assigned by the application or the system to certain usage classes, such as manageability, IPC control channel, VoIP. And additional bit within the VLAN TCI field (bit 5 - DEI) defines whether the packet has a no drop requirement. This bit is not being used by DCB mechanisms.

User Bandwidth Group (UBWG): A user bandwidth group is a management parameter that is a binding of user priorities into bandwidth groups for provisioning purposes. The hardware implementation does not recognize the UBWG entity.

Traffic Class (TC): Incoming packets are placed in traffic classes. Per the DCB functional specification, there might be a 1:1 mapping between UP and TC, or more than one priority can be grouped into a single class. Such grouping does not cross boundaries of traffic BWGs. The X550 implements eight or four TCs and maps them to UP's according to a programmable register. This provides the best flexibility for the IT manager. However, when more than one UP is mapped to the same TC, they must have the same no-drop policy network wide.

Packet Buffer (PB): TCs are mapped to packet buffers in a straightforward 1:1 mapping. Packets are also placed in packet buffers based on their class assignments.

Traffic Bandwidth Group (BWG): For bandwidth allocation and grouping, one or more TC can be grouped into a Traffic Bandwidth Group (BWG). A BWG is a logical association at a node, and has no markings inside a packet header. End stations and switches are independent in their definition and allocation of grouping of different TCs. Consistency of behavior throughout the network is handled by the UBWG provisioning mechanism.

One or more TCs can be grouped in a BWG. BWGs are allocated a percentage of bandwidth of available Ethernet link. The allocated bandwidth for BWG can be further divided among the TCs that are inside the BWG.

Credits: Credits regulate the bandwidth allocated to BWGs and TCs. As part of the WSP algorithm, each BWG and TC has pre-allocated credits. Those are decremented upon each transmission and replenished cyclically. The ratio between the credits of the BWGs represents the relative bandwidth percentage allocated to each BWG. The ratio between the credits of the TCs represents the relative bandwidth percentage allocated to each TC within a BWG. The X550 effectively maintains one table that represents both ratios at once. Note that credits can get negative values down to the maximum sized frame allowed on the TC.

Maximum Credit Value: The maximum credit value establishes a limit for the running sum of credits allotted to a class or group. This value prevents stacking up stale credits that can be added up over a relatively long period of time and then used by TCs all at once, altering fairness and latency.

Note: The absolute value of the credits has no direct bearing on the allocation of bandwidth. The ratio between the credits does. However, the absolute value might have substantial impact on the algorithm behavior. Larger absolute values can impact the latency of high priority queues and their ability to serve bursts with minimum latency, whereas too small credit values might impact the correct functionality in presence of jumbo frames. The speed of the algorithm implementation should also be taken into account. The value of the maximum credit limit are also in principle not part of the main WSP algorithm. However, they impact the fairness of bandwidth reallocation between queues in case some queues do not transmit the full amount they have been permitted to. Also, small values prevent correct functionality of jumbo frames. From high-level simulations it appears that credits should be allocated as low as possible based on the speed of the algorithm. Maximum credit values for TCs should be 1.5x to 2x the size of the maximum entity expected in that TC.

Weighted Strict Priority (WSP): The algorithm implemented in the X550 for TC arbitration.

Group Strict Priority (GSP): Refer to the sections that follow for details.

Link Strict Priority (LSP): Refer to the sections that follow for details.

Table 7-53 (T2) defines the TCs, their association to BWGs and their bandwidth allocation. The following elements are defined in this table:

Table 7-53. Bandwidth Allocation to TCs and BWGs

T2: Traffic Class Bandwidth Allocation Within a BWG						
TC_N	BWG	TC Refill	TC Max Credits	LSP	GSP	TC Min Credits (according to GSP 0/1)
0	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
1	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
2	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
3	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
4	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
5	3	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
6	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	1/1	-MSS / -2,048 KB
7	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB

Notes:
1. Due to a pipelined implementation, the TC credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.
2. All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).

TC: Configuration — The traffic type associated to the packet buffer where incoming packets are kept before transmission (or discard — not implemented in transmit in the X550). TC7 is the highest priority TC.

BWG: Configuration — Traffic BWG that a TC belongs to.

TC Credit Refill: Configuration — The X550's WSP algorithm implements credit refill as the technique for traffic class percentage allocation. The credits refill are added to each TC credit count on each completion of the full round of the algorithm (after all TCs had their chance to send what they had in store and if they had credits for it).

The X550's driver needs to calculate the TC Credit refill to match the percentage allocated through management (in the MIB). The TC credit refill table includes, in one table both the TC and the BWG. Since the WSP arbitration is self timed, the ratio between the credits refill is the only defining parameter for the TC and BWG. The absolute values of the refill have significance as to the rate of distribution between the queues and can have impact on latency and on the momentary stability of the bandwidth fairness. Results from simulations indicate that the quantum for the refill should be small to prevent large swings. It should not be too small as to create overhead in the mechanism due to the execution time; however. The X550 allows a value of 64 bytes to 32 KB, A dynamic range of x500. The usage model is likely to call for a smallest refill value of 256 bytes to 512 bytes. This leaves a dynamic range of 80x-200x. For example, if a queue is assigned 1% and this is translated into a 256-byte increment, another assigned with 99% would have a refill value of 25344 bytes.

TC MaxCredit: Configuration — To prevent use of stale credits, the number of credits each TC can accumulate upon refill is limited. The TC credit can only reach *MaxCredit*, beyond that its value gets recycled to other queues. Refer to the recycling mode for more details. The maximum range for the *MaxCredit* is 256 KB.

Note: Full testing of many values for maximum value is unnecessary. Hierarchical testing should be applied. For the random test, four to six values should be sufficient. It is important that the testing includes values that are relevant to the interesting zone of maximum value. For example, in the 0.8x to 2x the relevant largest entity in the class. Although class with an expected shorter packet could use a smaller *MaxCredit*, it is recommended that testing fully covers the cases where all the classes have similar values of *MaxCredit*, as it is a possible variant use of the algorithm.

Link Strict Priority (LSP): Configuration — If set, this bit specifies that this TC can transmit without any restriction of credits. This effectively means that this TC can take up the entire link bandwidth, unless preempted by higher priority traffic. If this bit is set, *TC.CreditRefill* must be set to 0b to ensure fair bandwidth allocation. Preferably, the algorithm implementation should disregard non-zero values in all its calculations.

Group Strict Priority (GSP): Configuration — This bit defines whether strict priority is enabled or disabled for this TC within its BWG. If *TC.GSP* is set to 1b, the TC is scheduled for transmission using strict priority. It does not check for availability of TC.Credits. It does check whether the BWG of this TC has credits (such as the amount of traffic generated from this TC is still limited by the BWG allocated for the BWG (T3. BWGP). If this bit is set, *TC.CreditRefill* values can be set to 0b, if a non-zero value is configured, TC credits are reduced first from the GSP TC and if reached to zero from other TCs in the group, if the refill credits are configured to zero the TC credits are reduced from the other TCs in the BWG.

Note: Since the *TC.GSP* parameter relates to individual TCs, some BWGs might have both TCs with bandwidth allocation and TCs with GSP. This is a hybrid usage mode that is complex to validate and is possibly secondary in importance.

Usage Note — It is possible that a TC using LSP dominates the link bandwidth if there are no packets waiting and eligible in higher priority TCs. To guarantee correct bandwidth allocation, all TCs with the unlimited bit set should be in the same traffic BWG (high priority group). Note that this is different than a typical DCB deployment considered where BWG is created with functional grouping, like LAN, SAN, and IPC etc. TCs with the *LSP* bit set should be the first to be considered by the scheduler (the first TCs). For example, from queue 7 to queue 5 with the other five TCs for groups with bandwidth allocation. These are not strict requirements, if these rules are not followed, undesirable behavior could occur in some cases. A group containing only TCs with the unlimited bit set effectively has zero BWG credits since TCs with the LSP unlimited bit set should have TC credits set to zero. Use of LSP / TC.GSP should be restricted to UPs/TCs that service trusted applications.

TC Credits: Run-time parameter — TC credits is a running algebraic counter for each TC. It holds the number of available credits in 64-byte units. The algorithm runs sequentially between the TCs and enables transmission for those TCs that have positive pending credits. Note that credits can get negative values down to the maximum sized frame allowed on the TC.

Table 7-54 (T3) defines the hierarchy of BWG similarly to T2 defining the hierarchy within the BWG's. T3 implementation should include eight rows.

Table 7-54. Line Bandwidth Allocations to Bandwidth Groups (BWG) (with example)

T3: Line bandwidth allocation to Traffic Bandwidth Groups (BWG)				
BWG	**BWG Refill Credits**	**BWG MaxCredit**	**BWG Credit**	**Description**
0	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	IPC
1	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	SAN
2	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	LAN
3	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	Manageability
-				
-				
-				
7				

Note: Due to a pipelined implementation, the BWG credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.

BWG: Configuration — This is the number of the traffic BWG that is three bits wide. This field corresponds to the *TC.BWG* field in Table 7-53.

BWG Refill Credits: A virtual number — The credits provisioned for this BWG. The credits ratio between the BWG's should reflect the ratio of bandwidth between the BWG's. In the actual implementation, this number is the sum of the credit Refills of the TC's associated with this BWG.

BWG MaxCredit: A virtual number — The maximum credits for a BWG. Credits in the BWG.Credit counter are limited to this value. Credits that should have been refilled above this value are lost. In effect, due to the self-timed cyclic nature of the WSP algorithm, those credits are distributed between all BWG's. In the actual implementation, this number is the sum of the MaxCredit of the TC's associated with this BWG.

BWG.Credit: Run-time parameter — A running algebraic counter that is decremented for each transmission. At the end of each cycle, this counter is synchronized with the sum of the TC.Credit counter associated with this BWG. The synchronization algorithm depends on the recycling mode. refer to the sections that follow for details about arbitration configurations. Note that credits can get negative values down to the maximum sized frame allowed on the BWG.

7.6.2.4.2 Arbiters Conventions

The WSP scheme previously described is written with the data plane arbiter in mind. However, the same scheme is used by the transmit descriptor plane arbiter and a subset of it is used by the receive data arbiter. To distinguish between the two arbiters, attributes of the each arbiter are prefixed as depicted in Table 7-55.

Table 7-55. Attributes of Tx Arbiters

Attribute	Tx Packet Arbiter	Tx Descriptor Arbiter
TC	P-TC	D-TC
BWG	P-BWG	D-BWG
TC Credit Refill	P-TC Credit Refill	D-TC Credit Refill
TC MaxCredit	P-TC MaxCredit	D-TC MaxCredit
LSP	P-LSP	D-LSP
GSP	P-GSP	D-GSP
TC Credits	P-TC Credits	D-TC Credits
BWG Refill Credits	P-BWG Refill Credits	D-BWG Refill Credits
BWG MaxCredits	P-BWG MaxCredits	D-BWG MaxCredits
BWG Credits	P-BWG Credits	D-BWG Credits

Table 7-56 lists the register fields that contain the relevant attributes from Table 7-55.

Table 7-56. Registers Allocation for Tx TC Arbiters

Attribute	Tx Packet Arbiter	Tx Descriptor Arbiter
TC Control registers	RTTPT2C	RTTDT2C
BWG	BWG	BWG
TC credit refill	CRQ	CRQ
TC MaxCredit	MCL	MCL
LSP	LSP	LSP
GSP	GSP	GSP
TC credits	CCC	CCC

7.6.2.4.3 Tx TC WSP Arbitration Configurations

There are two parameters influencing the Tx TC WSP arbitration:

- RR/WSP
- Recycle mode

A separate set of configuration parameters exists for each of the three TC arbiters, as listed in Table 7-57.

Table 7-57. Configuration Parameters for the Tx and Rx TC Arbiters

Parameter	Tx Packet Arbiter	Tx Descriptor Arbiter	Rx Packet Arbiter
	RTTPCS	RTTDCS	RTRPCS
RR/WSP Mode	TPPAC	TDPAC	RAC
Recycle Mode	TPRM	TDRM	RRM

The impact of these settings on the arbitration is described in the sections that follow.

7.6.2.4.3.1 RR/WSP

When this bit is set, the arbitration is in WSP mode. When reset, the arbitration is in flat frame-based round robin mode. In RR mode, one frame is transmitted from each packet buffer in its turn. BWG and TC parameters do not apply.

7.6.2.4.3.2 Recycle Mode

Architecture Overview of Recycle — As a result of GSP transmits and TCs that reach their maximum credit limit, the credit count of a BWG might not match the total credit count of its TCs (refer to the sections that follow for more details). It is not merely an arithmetic issue. The WSP algorithm, dual hierarchy behaves as a maximum allocation algorithm within the BWG's and minimum allocation algorithm between the BWG's. Since the recycle is self timed, when a BWG does not transmit all of its allocated bandwidth within a cycle, at the end of the cycle its bandwidth is in effect reallocated to all BWG's. This results in a minimum allocation behavior. Inside the BWG's; however, this notion of self timing does not exist. Some explicit mechanism is required to recycle bandwidth within a BWG rather than to all the BWG's — The requirement to have a minimum allocation behavior.

- A BWG credit count might not match the total credit count of its TCs in the following cases:
  - **A TC is defined as GSP** — When a GSP is selected, the BWG credits are decremented but no TC is deducted. Therefore, the BWG credit count would be lower that the credit count of its TCs.
  - **Max credits during refill** — If a TC reaches its max credits value during refill, some credits are lost for that TC. However, the BWG for that TC is provided with the full refill count. Therefore, the BWG credit count would be higher that the credit count of its TCs.
- One bit per TC arbiter governs the recycle mode of the WSP algorithm:
  - **0: No Recycling** — At the end of each full arbitration cycle all TC's are refilled with their TC.Refill up to their TC.MaxCredits values. All BWG.Credit are loaded by the sum of the BWG TC.Credit.
  - **1: Recycle** — TC credits for TC's that have reached their maximum are recycled to other TC's of the BWG. The operation is calculated based on the BWG.Credit and the TC.Credits after their refill. The difference between them is the BWG.Recycle value.
    - **Positive BWG.Recycle** — The recycle algorithm adds credits from the BWG.Recycle to the TC.Credits starting from the highest priority TC in the BWG down, considering the TC.MaxCredit, until BWG.Recycle is zero.
    - **Negative BWG.Recycle** — The recycle algorithm subtracts credits from the BWG.Recycle to the TC.Credits starting from the lowest priority TC in the BWG up, until BWG.Recycle is zero.

7.6.2.4.4 Tx TC WSP Arbiter Algorithm

The Transmit Packet Plane Arbitration Control (TPPAC) bit in the RTTPCS registers determines the scheduler type (RR or WSP).

Strict Priority — The strict-priority aspect of the TC WSP arbiter resides in the fact that once a TC has been granted for a data read request or for transmission, the highest priority TCs are checked (again) in a strict-priority order, starting from TC7, even if the granted TC still has credits for another data read request or transmission.

Note: The descriptor plane arbiter cannot issue a data read request unless there is an unused request for data. Therefore the arbiter stalls in its current state each time there are no data read requests available. The next arbitration decision is only done once there is at least one free data read request.

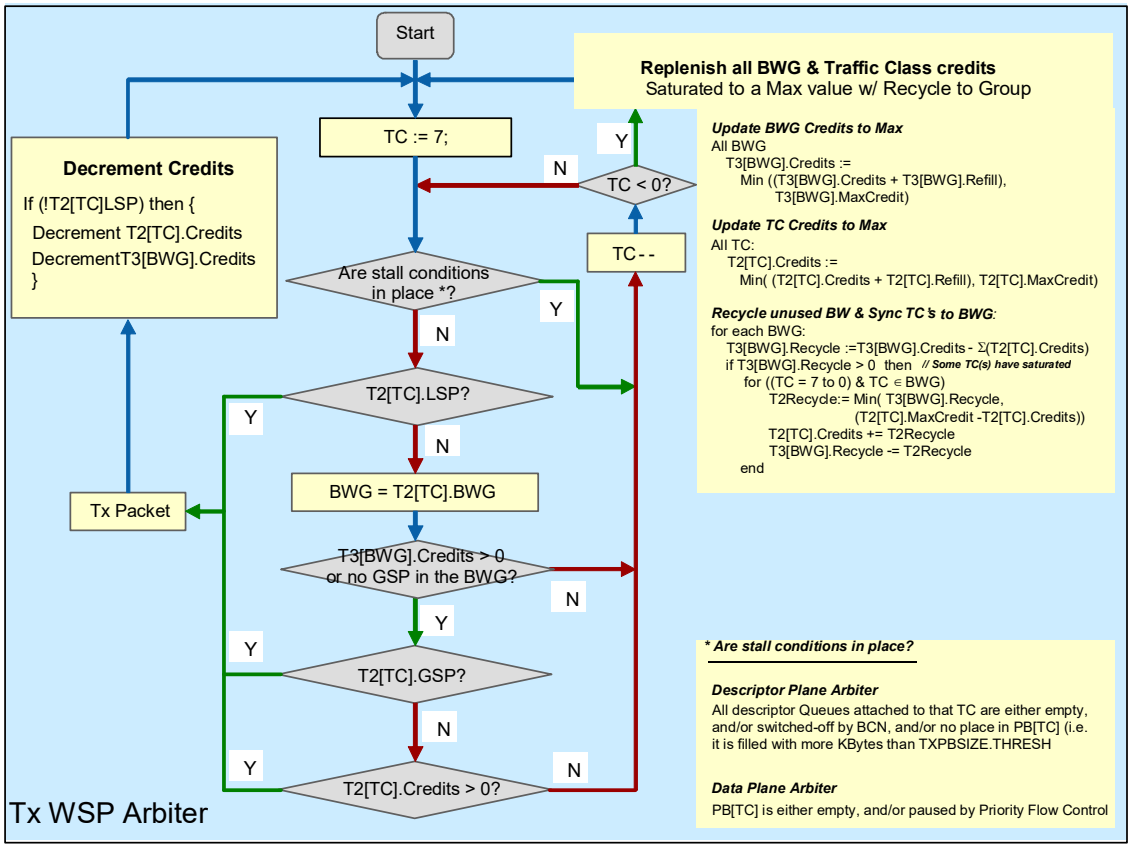


Figure 7-34. Tx TC WSP Arbiters Operation

7.6.3 Receive-Side Capabilities

7.6.3.1 User Priority to Traffic Class Mapping

To enable different TC support for incoming packets and proper behavior per TC, the X550's receive packet buffer is segmented into several packet buffers. The X550 supports the following configurations of packet buffer segmentation:

- DCB disabled — Single buffer of 384 KB
- DCB enabled with 4 TCs — 4 buffers, 96 KB each
- DCB enabled with 8 TCs — 8 buffers, 48 KB each
- DCB enabled with 8 TCs — 8 buffers, buffers 3:0 are 72 KB and buffers 7:4 are 24 KB

Incoming packets are transferred from the packet buffers into data buffers in system memory. Data buffers are arranged around descriptor rings described in Section 7.1.5.5. Each descriptor queue is assigned dynamically to a given TC (and therefore to a packet buffer) as described in Section 7.1.3.

Configuration registers:

- The size of each buffer is defined by the RXPBSIZE[0-7] registers. Note that it is possible to configure the buffers at 1 KB granularity.
- A received packet is assigned by the X550 to a TC, and is thus routed to the corresponding Rx packet buffer according to its *User Priority* field in the 802.1Q tag and according to a UP to TC mapping table loaded into the RTRUP2TC register.

Caution: Different UP to TC mappings can be loaded in each direction Tx and Rx, as per RTTUP2TC and RTRUP2TC registers, respectively. But in such a case, when a packet is looped back by the internal VM to VM switch, it is routed to the Rx packet buffer that corresponds to the TC that was used in Tx.

7.6.3.2 Rx PB Weighted Strict Priority Arbiter

The X550's Rx arbiter determines the order in which packets are written from the different packet buffers into system memory. Note that each packet buffer by itself is drained in the order packets arrived, as long as it deals with packets destined to the same Rx queue.

The arbitration algorithm between the receive packet buffers is WSP, similar to the TC scheme on the transmit side. Motivation for this scheme is as follows:

1. The major consideration is to prevent any delay in delivery of high-priority traffic.
2. Allocation of credits controls the bandwidth allocated to the different packet buffers.
3. A secondary mean of bandwidth allocation is the priority flow control. By altering the flow control high watermark, the X550 can effectively (if coarsely) regulate bandwidth allocation to types of traffic.

Table 7-58 (T4) defines the TCs and their bandwidth allocation. The following elements are defined in this table:

Table 7-58. Bandwidth Allocation to Traffic Classes and Bandwidth Groups

T4: Packet Buffer Bandwidth Allocation Within a BWG						
PB	**BWG**	**PB Refill**	**PB Max Credits**	**LSP**	**GSP**	**PB Min Credits (according to GSP 0/1)**
0	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
1	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
2	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
3	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
4	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
5	3	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
6	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
7	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB

Note:
1. Due to a pipelined implementation, the PB credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.
2. All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).

Table 7-59 (T5) defines the hierarchy of BWG similarly to T4 defining the hierarchy within the BWG's. T4 implementation should include eight rows.

Table 7-59. Packet Buffer Allocations to BWGs (with Example)

T5: Line Bandwidth Allocation to Traffic BWGs				
BWG	**BWG Credit Refill**	**BWG MaxCredit**	**BWG Credit**	**Description**
0	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	IPC
1	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	SAN
2	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	LAN
3	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	MGMT
-				
-				
-				
7				

The attributes of the Rx packet arbiter are described in Section 7.6.2.4.2.

7.6.3.2.1 Rx TC Arbitration Configurations

Table 7-60 lists the register fields that control the Rx arbiter.

Table 7-60. Registers Allocation for DCB Rx Arbiters

Attribute	Rx Packet Arbiter
PB Control registers	RTRPT4C
PB Status registers	RTRPT4S
BWG	BWG
PB credit refill	CRQ
PB MaxCredit	MCL
LSP	LSP
GSP	GSP
PB credits	CCC

Configuration parameters for the Rx packet arbiter are defined and listed in Section 7.6.2.4.1.

7.6.3.2.2 Rx TC WSP Arbiter Algorithm

The Rx packet arbiter operates in RR or WSP mode, configured through the *RAC* bit in the RTRPCS register. The WSP arbiter operation is described as follows.

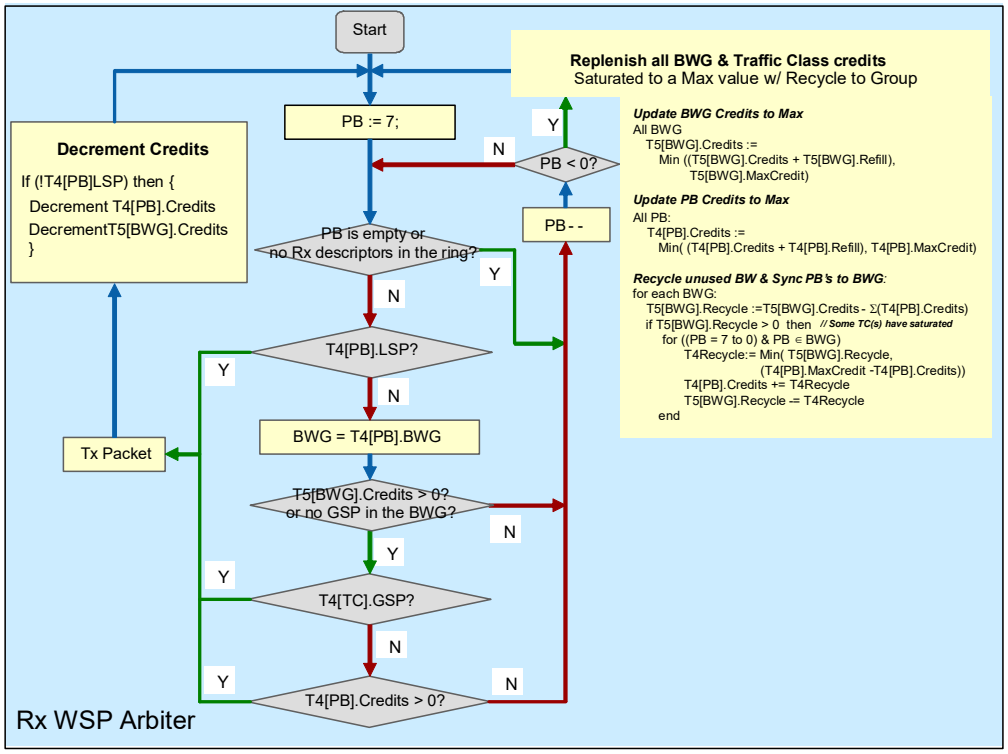


Figure 7-35. Rx Packet WSP Arbiter Operation

7.7 Time SYNC (IEEE1588 and 802.1AS)

7.7.1 Overview

IEEE 1588 addresses the clock synchronization requirements of measurement and control systems. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources. The protocol is spatially localized and allows simple systems to be installed and operate.

The IEEE802.1AS standard specifies the protocol used to ensure that synchronization requirements are met for time sensitive applications, such as audio and video, across Bridged and Virtual Bridged Local Area Networks consisting of LAN media where the transmission delays are almost fixed and symmetrical; for example, IEEE 802.3 full duplex links. This includes the maintenance of synchronized time during normal operation and following addition, removal, or failure of network components and network re-configuration. It specifies the use of IEEE 1588 specifications where applicable.

Note: The time sync mechanism activation is possible in full-duplex mode only and is not supported at 100 Mb/s link speed.

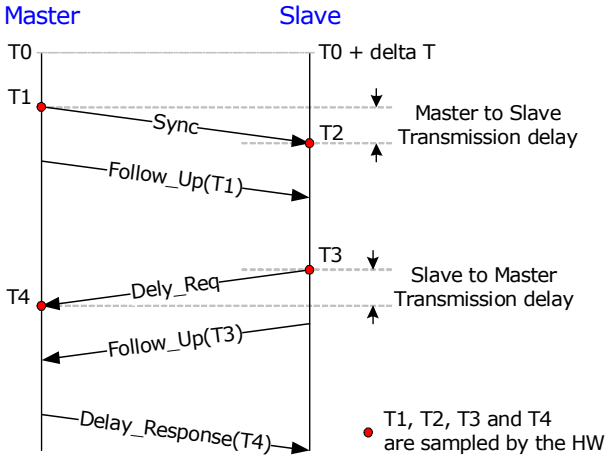
Note: The clock used for Time Sync is a 80 MHz clock, so the limit of the accuracy of all Time Sync operations is 12.5 ns.

7.7.2 Flow and Hardware/Software Responsibilities

The operation of a Precision Time Protocol (PTP) enabled network is divided into two stages: initialization and time synchronization.

At the initialization stage, every master-enabled node starts by sending sync packets that include the clock parameters of its clock. Upon receipt of a sync packet, a node compares the received clock parameters to its own and if the received parameters are better, then this node moves to a slave state and stops sending sync packets. While in slave state, the node continuously compares the incoming packet to its currently chosen master and if the new clock parameters are better, than the master selection is transferred to this master clock. Eventually the best master clock is chosen. Every node has a defined time-out interval that if no sync packet was received from its chosen master clock it moves back to a master state and starts sending sync packets until a new best master clock (PTP) is chosen.

The time synchronization stage is different to master and slave nodes. If a node is in a master state it should periodically send a sync packet that is time stamped by hardware on the Tx path (as close as possible to the PHY). After the sync packet, a Follow_Up packet is sent that includes the value of the time stamp kept from the sync packet. In addition, the master should time stamp Delay_Req packets on its Rx path and return to the slave that sent the time stamp value using a Delay_Response packet. A node in a slave state should time stamp every incoming sync packet and if it came from its selected master, software uses this value for time offset calculation. In addition, it should periodically send Delay_Req packets to calculate the path delay from its master. Every sent Delay_Req packet sent by the slave is time stamped and kept. With the value received from the master with Delay_Response packet, the slave can now calculate the path delay from the master to the slave. The synchronization protocol flow and the offset calculation are shown in Figure 7-36.



Calculated delta T = [(T2-T1)-(T4-T3)]/2 // assuming symmetric transmission delays

Toffset = - delta T // offset at the Slave

Figure 7-36. Sync Flow and Offset Calculation

Hardware responsibilities are:

1. Identify the packets that require time stamping.
2. Time stamp the packets on both Rx and Tx paths.
3. Store the time stamp value for software.
4. Keep the system time in hardware and give a time adjustment service to software.
5. Maintain auxiliary features related to the system time.

Software responsibilities are:

1. Manageability controller protocol execution, which means defining the node state (master or slave) and selection of the master clock if in slave state.
2. Generate PTP packets, consume PTP packets.
3. Calculate the time offset and adjust the system time using a hardware mechanism for that.
4. Enable configuration and usage of the auxiliary features.

Action	Responsibility	Node Role
Generate a sync packet with time stamp notification in the descriptor.	Software	Master
Time stamp the packet and store the value in registers (T1).	Hardware	Master
Time stamp incoming sync packet, store the value in register	Hardware	Slave
Read the time stamp from register put in a Follow_Up packet and send.	Software	Master
Once received, the Follow_Up store T2 from registers and T1 from Follow_up packet.	Software	Slave
Generate a Delay_Req packet with time stamp notification in the descriptor.	Software	Slave
Time stamp the packet and store the value in registers (T3).	Hardware	Slave
Time stamp incoming Delay_Req packet, store the value in register	Hardware	Master

Action	Responsibility	Node Role
Read the time stamp from register and send back to slave using a Delay_Response packet.	Software	Master
Once received, the Delay_Response packet calculate offset using T1, T2, T3 and T4 values.	Software	Slave

7.7.2.1 Time Sync Indications in Rx and Tx Packet Descriptors

Some indications need to be transferred between software and hardware regarding PTP packets. On the Tx path, software should set the 1588 bit in the Tx packet descriptor (bit 9). On the Rx path, hardware has two indications to transfer to software, one is to indicate that this packet is a PTP packet (whether time stamp is taken or not). This is also for other types of PTP packets needed for management of the protocol and this bit is set only for the L2 type of packets (the PTP packet is identified according to its EtherType). PTP packets have the *L2 Packet* bit in the packet type field set (bit 11 in the receive descriptor) and the EtherType matches the filter number set by software in the ETQF registers to filter PTP packets. The UDP type of PTP packets do not need such indication since the port number (319 for event and 320 all the rest PTP packets) directs the packets toward the time sync application. The second indication is RDESC.*STATUS.TS* (bit 16) to indicate to software that time stamp was taken for this packet. Software needs to access the time stamp registers to get the time stamp values.

7.7.3 Hardware Time Sync Elements

All time sync hardware elements are reset to their initial values (as defined in Section 8.2.2.21) upon Master reset. Upon change in link speed some of the time sync parameters should be changed accordingly.

7.7.3.1 System Time Structure and Mode of Operation

The *SYSTIME* is a 96 bit register is composed of: SYSTIMR, SYSTIMEL and SYSTIMEH registers: The SYSTIMR register holds the sub-nanosecond fraction, the SYSTIMEL register holds the nanosecond fraction and the SYSTIMEH register holds the sec fraction of the time (note that the upper two bits of the SYSTIMEL register are always zero while the max value of this register is 999,999,999 decimal).

- **Initial Setting** — Setting the initial time is done by direct write access to the SYSTIME register. Software should set first the SYSTIMEL and then set the SYSTIMEH. Setting the SYSTIMR is meaningless while it represents sub-nanosecond units.
- **Run Time** — During run time the *SYSTIME* timer value in the SYSTIMEH, SYSTIMEL and SYSTIMR registers, is updated periodically each 12.5 ns clock cycle according to the following formula:
  - Define: ***INC_TIME* == 12.5 ns +/- *TIMINCA.Incvalue* * 2^{-32} ns**. Add or subtract the TIMEINCA.*INCVALUE* is defined by TIMINCA.*ISGN* (where 0b means Add and 1b means Subtract)
  - Then: ***SYSTIME* = *SYSTIME* + *INC_TIME***
- **Reading the *SYSTIME*** register by software is done by the following sequence:
  - Read the SYSTIMEL register.
  - Read the SYSTIMEH register.

- **Dynamic update of *SYSTIME*** registers is done by using the TIMADJ registers by the following flow:
  - Write the *TADJL* value and its *SIGN* to the TIMADJ register (the *SIGN* bit indicates if the *TADJL* value should be added or subtracted).
  - Following the write access to the TIMADJ register, the hardware repeats the following two steps at each 12.5 ns clock as long as the Tadjust > 0.
    - ***SYSTIME* = *SYSTIME* + *INC_TIME* +/- 1 ns**. Add or subtract 1 ns is defined by TIMADJ.*Sign* (where '0' means Add and '1' means Subtract).
    - ***TADJL* = *TADJL* - 1 ns**
  - As shown above, the time adjustment might take multiple clocks. Software might write a new value to the TIMADJ register before the hardware completed the previous adjustment. In such a case, the new value written by software, overrides the above equation. If such a race is not desired, the software could check that the previous adjustment is completed by one of the following methods:
    - Wait enough time before accessing the TIMADJ register which guarantees that the previous update procedure is completed.
    - Poll the matched TSICR.*TADJ* flag which is set by the hardware each time the update procedure is completed.

7.7.3.2 Time Stamping Mechanism

The time stamping logic is located as close as possible to the PHY. Figure 7-37 shows the exact point in time where the time value is captured by the hardware relative to the packet content. This is to reduce delay uncertainties originated from implementation differences. As the time stamp is sampled at a very late phase in the data path, the X550 does not insert it to the transferred packet. Instead, the X550 supports the two-step operation as follows for Tx and Rx.

Note: As the time stamping logic is located at the MAC/PHY interface, there is no timestamping of packets when MAC loopback is activated.

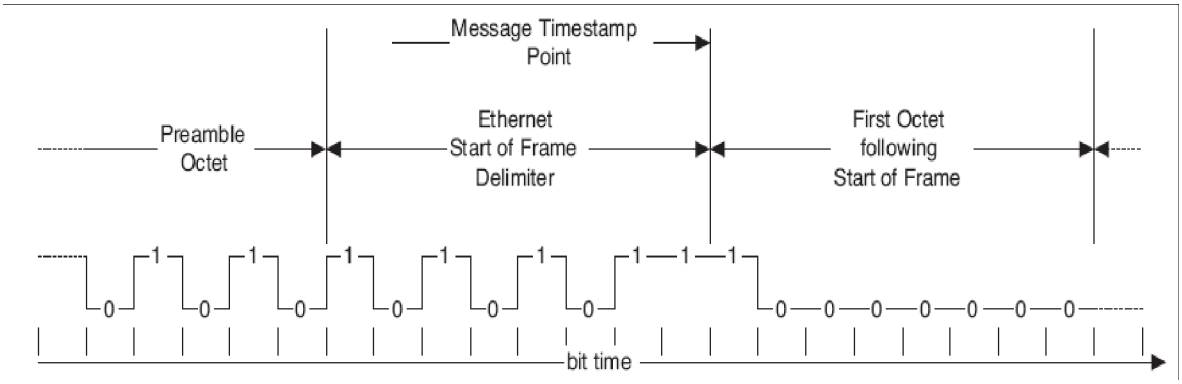


Figure 7-37. Time Stamp Point

7.7.3.2.1 Single Tx Time Stamping

The time stamp logic is activated if enabled by the TSYNCTXCTL.*EN* bit and the time stamp bit in the packet descriptor is set. In this case, hardware captures the packet's transmission time in the TXSTMPL and TXSTMPH registers. Software is responsible to read the transmission time and append it in the Folow_Up packet as shown in Figure 7-36.

7.7.3.2.2 Single Rx Time Stamping

On the Rx, this logic parses the traversing frame. If it is matching the message type defined in *RXMTRL* register and the TSYNCRXCTL.*TYPE* field, and timestamping is enabled by the TSYNCRXCTL.*EN* bit the reception time stamp is stored in the RXSTMPL and RXSTMPH registers. In addition, the TS status bits is reported in the Rx descriptor to identify that a time stamp was taken for this packet (stored in the RXSTMPL and RXSTMPH registers).

Note: The time stamp values are locked in the RXSTMPL and RXSTMPH registers until software accesses them. As long as software does not read these registers, hardware does not capture the time stamp of further Rx packets. To avoid potential deadlocks, it is recommended that software read the Rx time stamp registers at some time after sync or Delay_Req packets are expected. It would overcome erroneous cases on which the hardware latches a packet reception time while the packet's content was not posted properly to the software. Master software must not initiate consecutive sync requests before the previous response is received.

Note: If TSYNCRXCTL.*TYPE* = 4 (sample time stamp of all packets), the timestamps are not locked in the RXSTMPL and RXSTMPH registers, and each new packet timestamp is stored in these registers.

7.7.3.2.3 Multiple Rx Time Stamping

Packets that are identified to be time stamped by hardware are also indicated by the *TS* or *TSIP* flags in the receive descriptors. If the *TS* flag is set, the packet reception time is sampled by the hardware in the RXSTMPL/H registers (see Single Rx Time Stamping above). These registers are locked till the software reads its value. If the *TSIP* flag is set, the packet reception time is posted to the packet buffer in host memory. For more information about posting the time stamp in the receive buffer refer to Section 7.1.6.2.

7.7.4 Hardware Time Sync Elements

All time sync hardware is initialized as defined in the registers section upon MAC reset. The time sync logic is enabled if the TSAUXC.*Disable systime* flag is cleared.

The 1588 logic includes multiple registers larger than 32 bits which are indicated as xxxL (Low portion - LS) and xxxH (High portion - MS). When software accesses these registers (either read or write) it should access first the xxxL register (LS) and only then the xxxH register (MS). Accessing the xxxH might impact the hardware functionality which should be triggered only after both portions of the register are valid.

7.7.4.1 Target Time

The two target time registers TRGTTIMEL0/H0 and TRGTTIMEL1/H1 enable generating a time triggered event to external hardware using one of the SDP pins according to the setup defined in the TSSDP and TSAUXC registers. Each target time register is structured the same as the SYSTIMEL/H registers. If the value of SYSTIMEL/H is equal or larger than the value of the TRGTTIMEL/H registers, a change in level or a pulse is generated on the matched SDP outputs.

7.7.4.1.1 SYSTIM Synchronized Level Change Generation on SDP Pins

To generate a level change on one of the SDP pins when System Time (SYSTIM) reaches a pre-defined value, the driver should do the following:

1. Select a specific SDP pin by setting the TSSDP.*TS_SDPx_EN* flag to 1b (where 'x' is 0, 1, 2 or 3).
2. Assign a target time register to the selected SDP by setting the TSSDP.*TS_SDPx_SEL* field to 00b or 01b if level change should occur based on TRGTTIMEL0/H0 or TRGTTIMEL1/H1, respectively.
3. Define the selected SDPx pin as output, by setting the appropriate *SDPx_IODIR* bit in the ESDP register (where 'x' is 0, 1, 2, or 3).
4. Define that this SDP is used for TimeSync function by setting the appropriate *SDPx_NATIVE* bit in the ESDP register (while 'x' is 0, 1, 2, or 3). If used with SDP1, clear ESDP.*SDP1_FUNCTION* field.
5. Program the target time TRGTTIMEL*x*/H*x* to the required event time (where 'x' is 0 or 1).
6. Program the TRGTTIMEL*x*/H*x* to "Level Change" mode by setting the TSAUXC.*PLSG0* bit to 0b and TSAUXC.*EN_TTx* bit to 1b (where 'x' is 0 or 1).
7. To make this a one time operation, the TSAUXC.*DIS_TS_CLEAR* field should be cleared.
8. When the SYSTIMEL/H registers become equal or larger than the selected TRGTTIML/H registers, the selected SDP changes its output level.

7.7.4.1.2 SYSTIM Synchronized Pulse Generation on SDP Pins

An output pulse can be generated by using one of the target time registers to define the beginning of the pulse and the other target time registers to define the pulse completion time. To generate a pulse on one of the SDP pins when System Time (SYSTIM) reaches a pre-defined value, the driver should do the following:

1. Select a specific SDP pin by setting the TSSDP.*TS_SDPx_EN* flag to 1b (where 'x' is 0, 1, 2 or 3).
2. Set TSSDP.*TS_SDPx_SEL* field to 00b to define that the TRGTTIMEL0/H0 register defines the start of pulse time, and the TRGTTIMEL1/H1 register defines the end of pulse time.
3. Define the selected SDPx pin as output, by setting the appropriate *SDPx_IODIR* bit in the ESDP register (where 'x' is 0, 1, 2, or 3).
4. Define that this SDP is used for TimeSync function by setting the appropriate *SDPx_NATIVE* bit in the ESDP register (where 'x' is 0, 1, 2, or 3). If used with SDP1, clear ESDP.*SDP1_FUNCTION* field.
5. Program the target time TRGTTIMEL*x*/H*x* to the required event time (while 'x' is 0b or 1b).
6. TRGTTIMEL0/H0 should be set to a lower value than TRGTTIMEL1/H1.
7. Program the TRGTTIMEL0/H0 defined by the TSSDP.*TS_SDPx_SEL* to "Start of Pulse" mode by setting the TSAUXC.*PLSG0* bit to 1b and TSAUXC.*EN_TT0* bit to 1b (where 'x' defines the SDP used). The TRGTTIMEL1/H1 register *should be set* to indicate the end of the pulse and TSAUXC.*EN_TT1* bit should be set to 1b.

8. To make this a one time operation, the TSAUXC.*DIS_TS_CLEAR* field should be cleared.
9. When the SYSTIMEL/H registers becomes equal or larger than the TRGTTIMEL0/H0 registers the selected SDP changes its level. Then, when the SYSTIMEL/H registers becomes equal or larger than TRGTTIMEL0/H1 registers (that define the trailing edge of the pulse), the selected SDP changes its level back.

7.7.4.1.3 Synchronized Output Clock on SDP Pins

The X550 supports driving a programmable Clock on the SDP pins (up to two output clocks). The output clocks generated are synchronized to the global System time registers (SYSTIM). The Target Time registers (TRGTTIMEL0/H0 or TRGTTIMEL2/H1) can be used for the clock output generation. To start an clock output on one of the SDP pins when System Time (SYSTIM) reaches a pre-defined value, the driver should do the following:

1. Select a specific SDP pin by setting the TSSDP.*TS_SDPx_EN* flag to 1b (where 'x' is 0, 1, 2 or 3).
2. Select the target time register for a selected SDP, by setting the TSSDP.*TS_SDPx_SEL* field to 10b or 11b if output clock should occur based on TRGTTIMEL0/H0 or TRGTTIMEL2/H1 respectively.
3. Program the matched FREQOUT0/1 register to define clock half cycle time.
4. Define the selected SDPx pin as output, by setting the appropriate *SDPx_IODIR* bit in the ESDP register (where 'x' is 0, 1, 2, or 3).
5. Define that this SDP is used for TimeSync function by setting the appropriate *SDPx_NATIVE* bit in the ESDP register (where 'x' is 0, 1, 2, or 3). If used with SDP1, clear ESDP.*SDP1_FUNCTION* field.
6. If the output clock should start at a specific time, the TSAUXC.*ST0/1* flag should be set to 1b and the matched TRGTTIMEL*x*/H*x* should be set to the required start time.
7. Enable the clock operation by setting the relevant *TSAUXC.EN_CLK0/1* bit to 1b.

An interrupt can be generated from the clock output generated by the device by setting the relevant TSAUXC.*EN_TT0/1* bit to 1b and by setting the TSAUXC.*DIS_TS_CLEAR* to allow it to work continuously. The clock out drives initially a logical '0' level on the selected SDP. If the TSAUXC.*ST0/1* flag is cleared, it happens instantly when setting the TSAUXC.*EN_CLK0/1* bit. Otherwise it happens when SYSTIM is equal or larger than the TRGTTIM. Since then, the hardware repeats endlessly the following two steps:

1. Increment the used TRGTTIMEL*x*/H*x* by *FREQOUT*.
2. When SYSTIM is equal or larger than the TRGTTIM, the SDP reverts its output level.

Note: When clearing TSAUXC.*EN_CLK0/1* while TSAUXC.*EN_TT0/1* was set to generate interrupts, clear the matching TSAUXC.*EN_TT0/1* too to avoid one unexpected toggle.

7.7.4.2 Time Stamp Events

Upon a change in the input level of one of the SDP pins that was configured to detect Time stamp events using the TSSDP register or upon a setting of one of the TSAUXC.*SAMP_AUTx* fields, a time stamp of the system time is captured into one of the two auxiliary time stamp registers (AUXSTMPL0/H0 or AUXSTMPL1/H1). Software enables the timestamp of input event as follows:

1. Define the sampled SDP on AUX time 'x' ('x' = 0b or 1b) by setting the TSSDP.*AUXx_SDP_SEL* field while setting the matched TSSDP.*AUXx_TS_SDP_EN* bit to 1b.
2. Set also the TSAUXC.*EN_TSx* bit ('x' = 0b or 1b) to 1b to enable "timestamping".

Following a transition on the selected SDP, the hardware does the following:

1. The SYSTIM registers (low and high) are latched to the selected AUXSTMP registers (low and high).
2. The selected *AUTT0* or *AUTT1* flags are set in the TSICR and TSAUXC registers. If the *AUTT* interrupt is enabled by the TSIM register, and the 1588 interrupts are enabled by the *TIMESYNC* flag in the ICR register, an interrupt is asserted as well. After the hardware reports that an event time was latched, the software should read the latched time in the selected AUXSTMP registers. Software should read first the Low register and only then the High register. Reading the high register clears the relevant TSAUC.*AUTTx* field and releases the registers to sample a new event.

The software device driver may initiate a sampling of the current time by setting one of the TSAUXC.*SAMP_AUTx* fields. When one of these bits is written the hardware The SYSTIM registers (low and high) are latched to the selected AUXSTMP registers (low and high) and the selected *AUTT0* or *AUTT1* flags are set in the TSAUXC register. Once the device driver reads the selected AUXSTMP registers as described above, the hardware clears the relevant TSAUXC.*AUTTx* field and releases the registers to sample a new event.

7.7.5 Time Sync Interrupts

Time Sync related interrupts can be generated by programming the TSICR and TSIM registers. The TSICR register logs the interrupt cause and the TSIM register enables masking specific TSICR bits. Occurrence of a Time Sync interrupt sets the ICR.TIME_SYNC interrupt bit.

7.7.6 PTP Packet Structure

The time sync implementation supports both the 1588 V1 and V2 PTP frame formats. The V1 structure can come only as UDP payload over IPv4 while the V2 can come over L2 with its EtherType or as a UDP payload over IPv4 or IPv6. The 802.1AS uses only the layer 2 V2 format. Note that PTP frame structure over UDP is not supported in the X550 for IP tunneling packets.

Offset in Bytes	V1 Fields	V2 Fields	
Bits	**7 6 5 4 3 2 1 0**	**7 6 5 4 3 2 1 0**	
0	***versionPTP***	transportSpecific[1]	messageType
1		Reserved	versionPTP
2	versionNetwork	messageLength	
3			

Offset in Bytes	V1 Fields	V2 Fields
Bits	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
4	Subdomain	SubdomainNumber
5		Reserved
6		flags
7		
8		Correction Field
9		
10		
11		
12		
13		
14		
15		
16		reserved
17		
18		
19		
20	messageType	Source Port ID
21	Source communication technology	
22	***Sourceuuid***	
23		
24		
25		
26		
27		
28	sourceportid	
29		
30	sequenceId	sequenceId
31		
32	control	control
33	reserved	logMessagePeriod
34	falgs	n/a
35		

1. Should all be zero.

Note: Only the fields with the bold italic format colored red are of interest to hardware.

Ethernet (L2)	VLAN (Optional)	PTP EtherType	PTP message
Ethernet (L2)	IP (L3)	UDP	PTP message

When a PTP packet is recognized (by EtherType or UDP port address) on the Rx side, the version should be checked if it is V1, then the control field at offset 32 should be compared to message field in the RXMTRL register described in Section 8.2.2.21.2. Otherwise, the byte at offset 0 should be used for comparison to the rest of the needed field are at the same location and size for both V1 and V2.

Enumeration	Value
PTP_SYNC_MESSAGE	0
PTP_DELAY_REQ_MESSAGE	1
PTP_FOLLOWUP_MESSAGE	2
PTP_DELAY_RESP_MESSAGE	3
PTP_MANAGEMENT_MESSAGE	4
reserved	5–255

MessageId	Message Type	Value (hex)
PTP_SYNC_MESSAGE	Event	0
PTP_DELAY_REQ_MESSAGE	Event	1
PTP_PATH_DELAY_REQ_MESSAGE	Event	2
PTP_PATH_DELAY_RESP_MESSAGE	Event	3
Unused		4-7
PTP_FOLLOWUP_MESSAGE	General	8
PTP_DELAY_RESP_MESSAGE	General	9
PTP_PATH_DELAY_FOLLOWUP_MESSAGE	General	A
PTP_ANNOUNCE_MESSAGE	General	B
PTP_SIGNALLING_MESSAGE	General	C
PTP_MANAGEMENT_MESSAGE	General	D
Unused		E-F

If V2 mode is configured in TSAUXC register (Section 8.2.2.21.13), time stamp should be taken on PTP_PATH_DELAY_REQ_MESSAGE and PTP_PATH_DELAY_RESP_MESSAGE for any value in the message field in the RXMTRL register described at Section 8.2.2.21.2.

Note: A PTP over UDP packet encapsulated in a VXLAN or NVGRE tunnel is still identified as a PTP packet.

7.7.6.1 Time Sync Packets Identification Configuration

Table 7-61 summarize the setting needed to identify Time Sync packets.

Table 7-61. Enabling Receive Timestamp

Functionality	Register	Field	Setting Options
Enable receive timestamp in register	TSYNCRXCTL	En	En = 1b (must be set in all the following options).
Sampled V1 Control value	RXMTRL	CTRLT	The *CTRLT* defines the recognized V1 *Control* field. This field must be defined if V1 packets recognition is required. 0x0 = PTP_SYNC_MESSAGE
Sampled V2 MessageType value	RXMTRL	MSGT	The *MSGT* defines the recognized V2 *MessageType* field. This field must be defined if V2 packets recognition is required. 0x0 = Sync
Enable all packets for timestamp	TSYNCRXCTL	Type	Type equals to 100b enables sampling all packets. Useful only when posting the timestamp to the packet buffer in host memory, enabled per queue by the SRRCTL[n].*Timestamp*.
Enable L2 1588 packets for timestamp sampling	TSYNCRXCTL	Type	Type equals to 000b or 010b enable V2 packets with MessageType equals to MSGT Type equals to 101b enable all V2 packets with Message Type bit 3 zero (means any event packets)
	ETQF[n]	EType Filter enable IEEE_1588_TIME_STAMP	The EType on one of the enabled ETQF registers (Filter enable is '1') should be set to the 1588 EtherType (equals to 0x88F7) and *IEEE_1588_TIME_STAMP* field should be set.
Enable 1588 packets over UDP for timestamp sampling	TSYNCRXCTL	Type	Type equals to 001b enables V1 packets with *Control* field equals to CTRLT parameter Type equals to 010b enables V2 packets with *MessageType* fields equals to MSGT parameter Type equals to 101b enables all V2 packets with *MessageType* bit 3 zero (which means any event packets)
	RXMTRL	UDPT	Defines the UDP port to use for V1 detection
Define specific receive queue for the L2 1588 packets	ETQF[n]	Rx Queue Queue Enable	Setting the "Queue Enable" on the same ETQF register as above, the receive queue is defined by the *Rx Queue* field.

7.7.7 Virtualization

7.7.8 Overview

I/O virtualization is a mechanism that can be used to share I/O resources among several consumers. For example, in a virtual system, multiple operating systems are loaded and each operates as though the entire system's resources were at its disposal. However, for the limited number of I/O devices, this presents a problem because each operating system might be in a separate memory domain and all the data movement and device management has to be done by a Virtual Machine Monitor (VMM). VMM access adds latency and delay to I/O accesses and degrades I/O performance. Virtualized devices are designed to reduce the burden of the VMM by making certain functions of an I/O device shared among multiple guest operating systems or a Virtual Machine (VM), thereby allowing each VM direct access to the I/O device.

The X550 supports two modes of operations of virtualized environments:

1. Direct assignment of part of the port resources to different guest operating systems using the PCI SIG SR IOV standard. Also known as native mode or pass-through mode. This mode is referenced as **IOV mode** throughout this section.
2. Central management of the networking resources by an IOVM or by the VMM. Also known as software switch acceleration mode. This mode is referred to as **VMDq2 mode in this section**.

The virtualization offloads capabilities provided by the X550 apart from the replication of functions defined in the PCI SIG IOV specification are part of **VMDq2**.

A hybrid model, where part of the VMs are assigned a dedicated share of the port and the rest are serviced by an IOVM is also supported. However, in this case the offloads provided to the software switch might be more limited. This model can be used when parts of the VMs run operating systems for which VF drivers are available and thus can benefit from an IOV and others that run older operating systems for which VF drivers are not available and are serviced by an IOVM. In this case, the IOVM is assigned one VF and receives all the packets with Ethernet MAC Addresses of the VMs behind it.

The following section describes the support the X550 provides for these modes.

This section assumes a single-root implementation of IOV and no support for multi-root.

7.7.8.1 Direct Assignment Model

The direct assignment support in the X550 is built according to the following model of the software environment.

It is assumed that one of the software drivers sharing the port hardware behaves as a master driver (**Physical Function** or **PF** driver). This driver is responsible for the initialization and the handling of the common resources of the port. All the other drivers (**Virtual Function** drivers or **VF** drivers) might read part of the status of the common parts but cannot change them. The PF driver might run either in the VMM or in some service operating system. It might be part of an IOVM or part of a dedicated service operating system.

In addition, part of the non time-critical tasks are also handled by the PF driver. For example, access to CSR through the I/O space or access to the configuration space are available only through the master interface. Time-critical CSR space like control of the Tx and Rx queue or interrupt handling is replicated per VF, and directly accessible by the VF driver.

Note: In some systems with a thick hypervisor, the service operating system might be an integral part of the VMM. For these systems, each reference to the service operating system in the sections that follow refer to the VMM.

7.7.8.1.1 Rationale

The direct assignment model enables each of the VMs to receive and transmit packets with minimum of overhead. Non time-critical operations such as initialization and error handling can be done via the PF driver. In addition, it is important that the VMs can operate independently with minimal disturbance. It is also preferable that the VM interface to hardware should be as close as possible to the native interface in non-virtualized systems to minimize the software development effort.

The main time critical operations that require direct handling by the VM are:

- Maintenance of the data buffers and descriptor rings in host memory. To support this, the DMA accesses of the queues associated to a VM should be identified as such on the PCIe using a different requester ID.
- Handling of the hardware ring (tail bump and head updates)
- Interrupts handling

The capabilities needed to provide independence between VMs are:

- Per-VM reset and enable capabilities
- Tx rate control
- Allocating separate CSR space per VM. This CSR space is organized as close as possible to the regular CSR space to enable sharing of the base driver code.

Note: The rate control and VF enable capabilities are controlled by the PF.

7.7.8.2 System Overview

The following drawings show the various elements involved in the I/O process in a virtualized system. Figure 7-38 shows the flow in software VMDq2 mode and Figure 7-39 shows the flow in IOV mode.

This section assumes that in IOV mode, the driver on the guest operating system is aware that it operates in a virtual system (para-virtualized) and there is a channel between each of the VM drivers and the PF driver allowing message passing such as configuration request or interrupt messages. This channel can use the mailbox system implemented in the X550 or any other means provided by the VMM vendor.

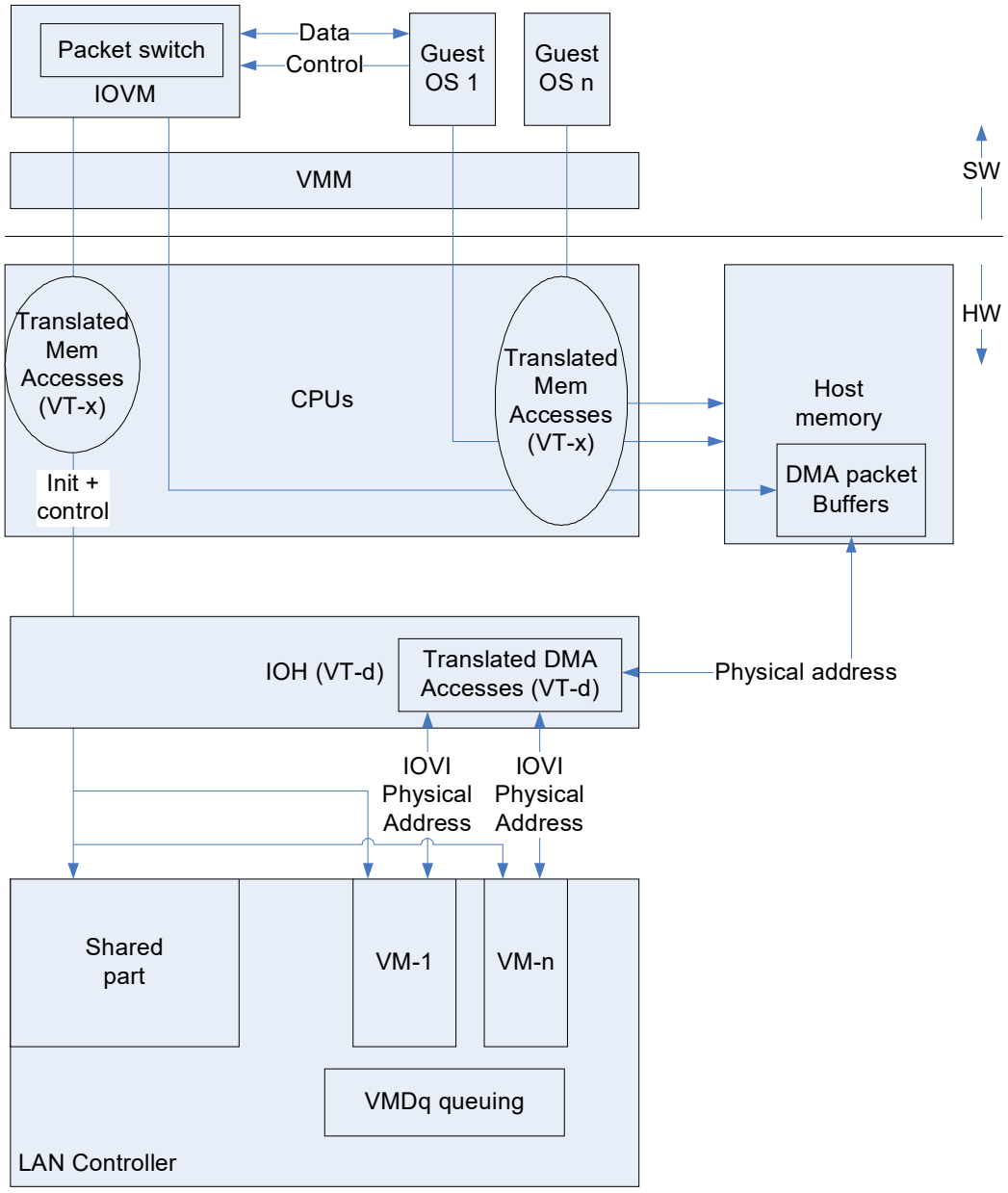


Figure 7-38. System Configuration for VMDq2 Mode

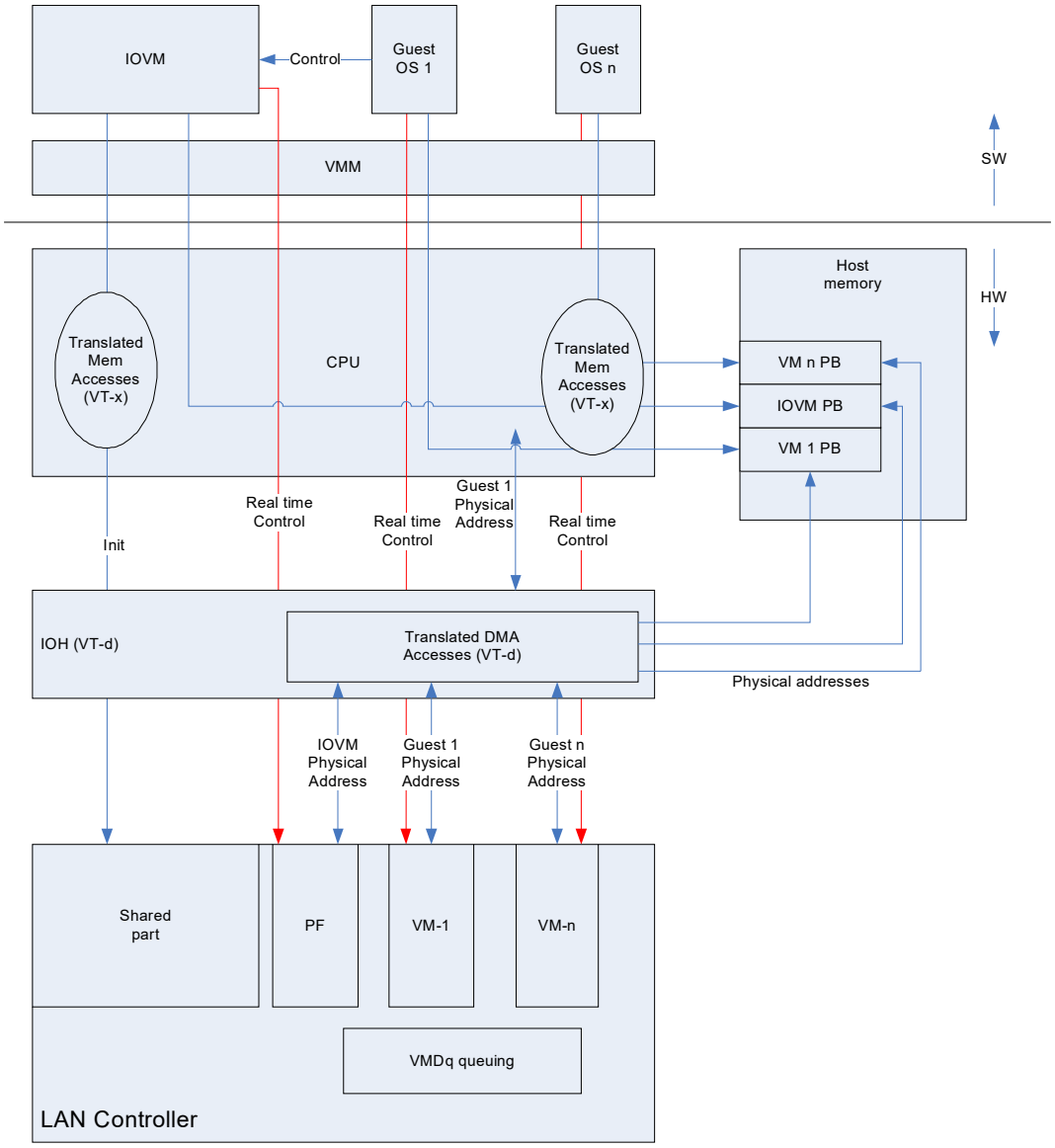


Figure 7-39. System Configuration for IOV Mode

7.7.9 PCI-SIG SR-IOV Support

7.7.9.1 SR-IOV Concepts

SR-IOV defines the following entities in relation to I/O virtualization:

- **Virtual Image (VI)** — Part of the I/O resources are assigned to a A VM.
- **I/O Virtual Intermediary (IOVI) or I/O Virtual Machine (IOVM)** — A special VM that owns the physical device and is responsible for the configuration of the physical device.
- **Physical function (PF)** — A function representing a physical instance — One port for the X550. The PF driver is responsible for the configuration and management of the shared resources in the function.
- **Virtual Function (VF)** — A part of a PF assigned to a VI.

7.7.9.2 Configuration Space Replication

The SR-IOV specification defines a reduced configuration space for the virtual functions. Most of the PCIe configuration of the VFs comes from the PF.

This section describes the expected handling of the different parts of the configuration space for virtual functions. It deals only with the parts relevant to the X550.

Details of the configuration space for virtual functions can be found in Section 9.3.

7.7.9.2.1 Legacy PCI Configuration Space

The legacy configuration space is allocated to the PF only and emulated for the VFs. A separate set of BARs and one bus master enable bit is allocated in the SR-IOV capability structure in the PF and is used to define the address space used by the entire set of VFs.

All the legacy error reporting bits are emulated for the VF. See Section 7.7.9.4 for details.

7.7.9.2.2 Memory BARs Assignment

The SR-IOV specification defines a fixed stride for all the VF BARs, so that each VF can be allocated part of the memory BARs at a fixed stride from a basic set of BARs. In this method, only two decoders per replicated BAR per PF are required and the BARs reflected to the VF are emulated by the VMM.

The only BARs that are useful for the VFs are BAR0 and BAR3, so only those are replicated. Table 7-62 lists the existing BARs and the stride used for the VFs:

Table 7-62. BARs in the X550 (64-bit BARs)

BAR	Type	Usage	Requested Size per VF (=Stride)
0, 1	Mem	CSR space	Maximum (16 KB, page size). For page size see Section 9.2.4.4.8 for more details.
2	n/a	Not used	N/A
3, 4	Mem	MSI-X	Maximum (16 KB, page size).
5	n/a	Not used	N/A

BAR0 of the VFs are a sparse version of the original PF BAR and include only the register relevant to the VF. For more details see Section 7.7.9.7.

Figure 7-40 shows the different BARs in an IOV-enabled system:

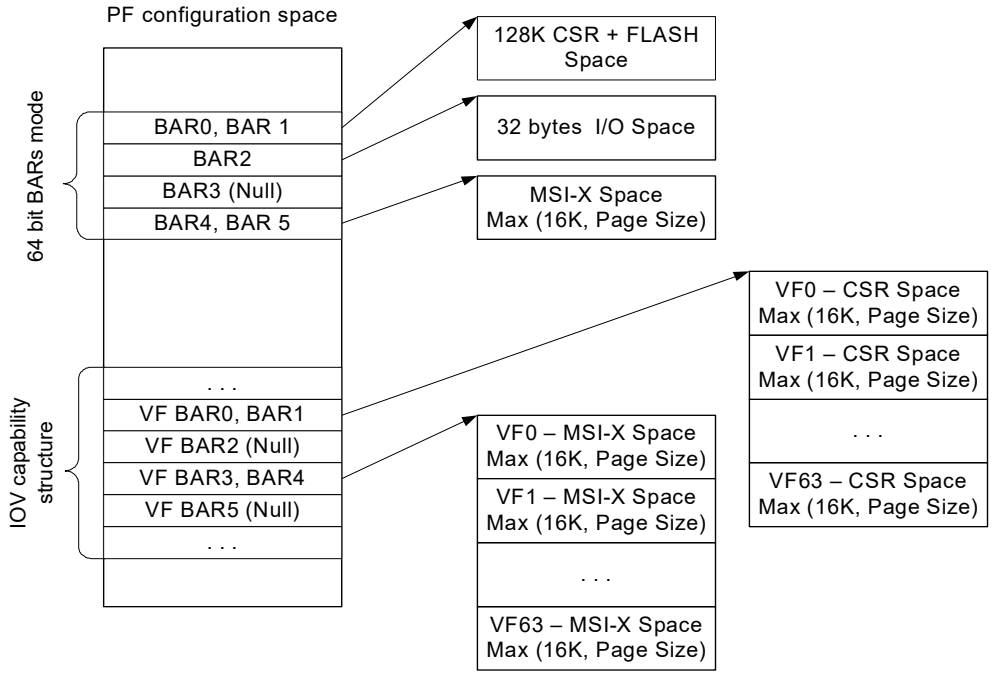


Figure 7-40. BARs in an IOV-enabled System

7.7.9.2.3 PCIe Capability Structure

The PCIe capability structure is shared between the PF and the VFs. The only relevant bits that are replicated are:

- Transaction pending.
- Function Level Reset (FLR). See Section 7.7.9.3 for details.

7.7.9.2.4 MSI and MSI-X Capabilities

Both MSI and MSI-X are implemented in the X550. MSI-X vectors can be assigned per VF. MSI is not supported for the VFs.

See Table 9-16 for more details of the MSI-X and PBA tables implementation.

7.7.9.2.5 VPD Capability

VPD is implemented only once and is accessible only from the PF.

7.7.9.2.6 Power Management Capability

The X550 does not support power management per VF. The power management registers exist for each VF, but only the D0 power state is supported.

7.7.9.2.7 Serial ID

Serial ID capability is not exposed in VFs.

7.7.9.2.8 Error Reporting Capabilities (Advanced and Legacy)

All the bits in this capability structure are implemented only for the PF. Note that the VMs see an emulated version of this capability structure. See Section 7.7.9.4 for details.

7.7.9.3 FLR Capability

The *FLR* bit is required per VF. Setting of this bit resets only a part of the logic dedicated to the specific VF and does not influence the shared part of the port. This reset should disable the queues, disable interrupts and the stop receive and transmit process per VF.

Setting the PF *FLR* bit resets the entire function.

7.7.9.4 Error Reporting

Error reporting includes legacy error reporting and Advanced Error Reporting (AER) or role-based capability.

The legacy error management includes the following functions:

- **Error capabilities enablement** — These are set by the PF for all the VFs. Narrower error reporting for a given VM can be achieved by filtering of the errors by the VMM. This includes:
  - SERR# Enable
  - Parity Error Response
  - Correctable Reporting Enable
  - Non-Fatal Reporting Enable
  - Fatal Reporting Enable
  - UR Reporting Enable
- **Error status in the configuration space** — These should be set separately for each VF. This includes:
  - Master Data Parity Error
  - Signaled Target Abort
  - Received Target Abort
  - Master Abort
  - SERR# Asserted
  - Detected Parity Error
  - Correctable Error Detected
  - Non-Fatal Error Detected
  - Unsupported Request Detected

AER capability includes the following functions:

- **Error capabilities enablement** — The *Error Mask*, and *Severity* bits are set by the PF for all the VFs. Narrower error reporting for a given VM can be achieved by filtering of the errors by the VMM. These includes:
  - Uncorrectable Error Mask Register
  - Uncorrectable Error Severity Register
  - Correctable Error Mask Register
  - ECRC Generation Enable
  - ECRC Check Enable
- **Non-Function Specific Errors Status in the configuration space**
  - Non-Function Specific Errors are logged in the PF
  - Error logged in one register only
  - VI avoids touching all VFs to clear device level errors
  - The following errors are not function specific
    - All Physical Layer errors
    - All Link Layer errors
    - ECRC Fail
    - UR, when caused by no function claiming a TLP
    - Receiver Overflow
    - Flow Control Protocol Error
    - Malformed TLP
    - Unexpected Completion
- **Function Specific Errors Status in the configuration space**
  - Allows Per-VF error detection and logging
  - Help with fault isolation
  - The following errors are function specific:
    - Poisoned TLP received
    - Completion Timeout
    - Completer Abort
    - UR, when caused by a function that claims a TLP
    - ACS Violation
- **Error logging** — Each VF has it’s own header log.
- **Error messages** — To ease the detection of the source of the error, the error messages should be emitted using the requester ID of the VF in which the error occurred.

7.7.9.5 Alternative Routing ID (ARI) and IOV Capability Structures

To allow more than eight functions per end point without requesting an internal switch, as usually needed in virtualization scenarios, the PCI-SIG defines the ARI capability structure. This is a new capability that enables an interpretation of the *Device* and *Function* fields as a single identification of a function within the bus. In addition, a new structure used to support the IOV capabilities reporting and control is defined. Both structures are described in sections Section 9.2.4.3 and Section 9.2.4.4. Refer to the following section for details on the Requester ID (RID) allocation to VFs.

7.7.9.6 RID Allocation

RID allocation of the VF is done using the *Offset* field in the IOV structure. This field should be replicated per VF and is used to do the enumeration of the VFs.

Each PF includes an offset to the first associated VF. This pointer is a relative offset to the Bus/Device/Function (BDF) of the first VF. The *Offset* field is added to PF's requester ID to determine the requester ID of the next VF. An additional field in the IOV capability structure describes the distance between two consecutive VF's requester IDs.

7.7.9.6.1 Bus Device Function Layout

7.7.9.6.1.1 ARI Mode

ARI allows interpretation of the device ID part of the RID as part of the function ID inside a device. Thus, a single device can span up to 256 functions. To ease the decoding, the least significant bit of the function number points to the physical port number. The *Next* bits indicate the VF number. Table 7-63 lists the VF RIDs.

The layout of RID's used by the X550 is reported to the operating system via the PCIe IOV capability structure. See Section 9.2.4.4.6.

Table 7-63. RID per VF — ARI Mode

Port	VF#	B,D,F	Binary	Notes
0	PF	B,0,0	B,00000,000	PF
1	PF	B,0,1	B,00000,001	PF
0	1	**B**,16,0	B,10000,000	Offset to first VF from PF is 128.
1	1	**B**,16,1	B,10000,001	
0	2	**B**,16,2	B,10000,010	
1	2	**B**,16,3	B,10000,011	
0	3	**B**,16,4	B,10000,100	
1	3	**B**,16,5	B,10000,101	
...				
0	64	**B**,31,6	B,11111,110	
1	64	**B**,31,7	B,11111,111	Last

7.7.9.6.1.2 Non-ARI Mode

When ARI is disabled, non-zero devices in the first bus cannot be used, thus a second bus is needed to provide enough RIDs. In this mode, the RID layout is as follows:

Table 7-64. RID per VF — Non-ARI Mode

Port	VF#	B,D,F	Binary	Notes
0	PF	B,0,0	B,00000,000	PF
1	PF	B,0,1	B,00000,001	PF
0	0	**B+1**,16,0	B+1,10000,000	Offset to first VF from PF is 384.
1	0	**B+1**,16,1	B+1,10000,001	
0	1	**B+1**,16,2	B+1,10000,010	
1	1	**B+1**,16,3	B+1,10000,011	
0	2	**B+1**,16,4	B+1,10000,100	
1	2	**B+1**,16,5	B+1,10000,101	
...				
0	63	**B+1**,31,6	B+1,11111,110	
1	63	**B+1**,31,7	B+1,11111,111	Last

Note: When the device ID of a physical function changes (because of LAN disable or LAN function select settings), the VF device IDs changes accordingly.

7.7.9.7 Hardware Resources Assignment

The main resources to allocate per VM are queues and interrupts. The assignment is static. If a VM requires more resources, it might be allocated to more than one VF. In this case, each VF gets a specific Ethernet MAC Address/VLAN tag to enable forwarding of incoming traffic. The two VFs are then teamed in software.

7.7.9.7.1 PF Resources

A possible use of the PF is for a configuration setting without transmit and receive capabilities. In this case, it is not allocated to any queues but is allocated to one MSI-X vector.

The PF has access to all the resources of all VMs, but it is not expected to make use of resources allocated to active VFs.

7.7.9.7.2 Assignment of Queues to VF

See Section 7.2.1.2.1 for allocating Tx queues.

See Section 7.1.3.2 for allocating Rx queues.

Table 7-65 lists the Tx and Rx queues to VF allocation.

Table 7-65. Queue to VF Allocation

VF	Queues in 16 VMs Mode	Queues in 32 VMs Mode	Queues in 64 VMs Mode
0	0-7	0-3	0-1
1	8-15	4-7	2-3
...	...	...	...
15	120-127	...	...
...		...	...
31		124-127	...
...			...
63			126-127

7.7.9.7.3 Assignment of MSI-X Vector to VF

See Section 7.3.4.3 for allocating MSI-X vectors in IOV mode.

7.7.9.8 CSR Organization

CSRs can be divided into three types:

- Global Configuration registers that should be accessible only to the PF — For example, link control and LED control. These types of registers also include all of the debug features such as the mapping of the packet buffers and is responsible for most of the CSR area requested by the X550. This includes per VF configuration parameters that can be set by the PF without performance impact.
- Per-VF parameters — For example, per VF reset, interrupt enable, etc. Multiple instances of these parameters are used only in an IOV system and only one instance is needed for non IOV systems.
- Per-queue parameters that should be replicated per queue — For example, head, tail, Rx buffer size, TPH tag, etc. These parameters are used by both a VF in an IOV system and by the PF in a non-IOV mode.

To support IOV without distributing the current drivers operation in legacy mode, the following method is used:

- The PF instance of BAR0 continues to contain the legacy and control registers. It is accessible only to the PF. The BAR enables access to all the resources including the VF queues and other VF parameters. However, it is expected that the PF driver does not access these queues in IOV mode.
- The VF instances of BAR0 provide control on the VF specific registers. These BARs have the same mapping as the original BAR0 with the following exceptions:
  - Fields related to the shared resources are reserved.
  - The queues assigned to a VF are mapped at the same location as the first same number of queues of the PF.
- Assuming some backward compatibility is needed for IOV drivers, The PF/VF parameters block should contain a partial register set as described in VF Device Registers section.

7.7.9.9 SR IOV Control

To control the IOV operation, the physical driver is provided with a set of registers. These include:

- The mailbox mechanism described in the next section.
- The switch and filtering control registers described in Section 7.7.10.9.
- PFVFLREC register indicating that a VFLR reset occurred in one of the VFs (bitmap).

7.7.9.9.1 VF-to-PF Mailbox

The VF drivers and the PF driver require some means of communication between them. This channel can be used for the PF driver to send status updates to the VFs (such as link change, memory parity error, etc.) or for the VF to send requests to the PF (add to VLAN).

Such a channel can be implemented in software, but requires enablement by the VMM vendors. To avoid the need for such an enablement, the X550 provides such a channel that enables direct communication between the two drivers.

The channel consists of a mailbox. Each driver can then receive an indication (either poll or interrupt) when the other side wrote a message.

Assuming a maximum message size of 64 bytes (one cache line), a memory of 64 bytes x 64 VMs = 4 KB is provided per port. The RAM is organized as follows:

Table 7-66. Mailbox Memory

RAM Address	Function	PF BAR 0 Mapping[1]	VF BAR 0 Mapping[2]
0 — 63	VF0 <-> PF	0 — 63	VF0 + MBO
64 — 127	VF1 <-> PF	64 — 127	VF1 + MBO
....			
(4 KB-64) — (4 KB-1)	VF63<-> PF	(4 KB-64) — (4 KB-1)	VF63 + MBO

1. Relative to mailbox offset.
2. MBO = mailbox offset in VF CSR space.

In addition for each VF, the VFMailbox and PFMailbox registers are defined to coordinate the transmission of the messages. These registers contain a semaphore mechanism to enable coordination of the mailbox usage.

The PF driver can decide which VFs are allowed to interrupt the PF to indicate a mailbox message using the PFMBIMR mask register.

The following flows describe the usage of the mailbox:

Table 7-67. PF-to-VF Messaging Flow

Step	PF Driver	Hardware	VF #n driver
1	Set PFMAILBOX[n].*PFU*.		
2		Set *PFU* bit if PFMAILBOX[n].*VFU* is cleared.	
3	Read PFMAILBOX[n] and check that *PFU* bit was set. Otherwise wait and go to Step 1.		
4	Write message to relevant location in VFMBMEM.		

Table 7-67. PF-to-VF Messaging Flow [continued]

Step	PF Driver	Hardware	VF #n driver
5	Set the PFMAILBOX[n].*STS* bit and wait for ACK[1].		
6		Indicate an interrupt to VF #n.	
7			Read the message from VFMBMEM.
8			Set the VFMAILBOX[n].*ACK* bit.
9		Indicate an interrupt to PF.	
10	Clear PFMAILBOX[n].*PFU*.		

1. The PF might implement a timeout mechanism to detect non-responsive VFs.

Table 7-68. VF-to-PF Messaging Flow

Step	PF Driver	Hardware	VF #n Driver
1			Set VFMAILBOX.*VFU*.
2		Set *VFU* bit if VFMAILBOX[n].*PFU* is cleared.	
3			Read VFMAILBOX[n] and check that *VFU* bit was set. Otherwise wait and go to Step 1.
4			Write message to relevant location in VFMBMEM.
5			Set the VFMAILBOX[n].*REQ* bit.
6		Indicate an interrupt to PF.	
7	Read PFMBICR to detect which VF caused the interrupt.		
8	Read the adequate message from VFMBMEM.		
9	Set the PFMAILBOX[n].*ACK* bit.		
10		Indicate an interrupt to VF #n.	
11			Clear VFMAILBOX[n].*VFU*.

The content of the message is hardware independent and is determined by software.

The messages currently assumed by this specification are:

- Registration to VLAN/multicast packet/broadcast packets — A VF can request to be part of a given VLAN or to get some multicast/broadcast traffic.
- Reception of large packet — Each VF should notify the PF driver what is the largest packet size allowed in receive.
- Get global statistics — A VF can request information from the PF driver on the global statistics.
- Filter allocation request — A VF can request allocation of a filter for queuing/immediate interrupt support.
- Global interrupt indication.
- Indication of errors.

7.7.9.10 DMA

7.7.9.10.1 RID

Each VF is allocated a RID. Each DMA request should use the RID of the VM that requested it. See Section 7.7.9.6 for details.

7.7.9.10.2 Sharing of the DMA Resources

The outstanding requests and completion credits are shared between all the VFs. The tags attached to read requests are assigned the same way as in a non-virtualized setting, although in VF systems tags can be re-used for different RIDs.

7.7.9.10.3 TPH

The TPH enable is common to all the devices (all PFs and VFs). Given a TPH enabled device, each VM might decide for each queue, on which type of traffic (data, headers, Tx descriptors, Rx descriptors) the TPH should be asserted and what is the CPU ID assigned to this queue.

Note: There are no plans to virtualize TPH in the IOH. Thus, the physical CPU ID should be used in the programming of the *CPUID* field.

7.7.9.11 Timers

7.7.9.11.1 TCP Timer

The TCP timer is available only to the PF. It might indicate an interrupt to the VFs via the mailbox mechanism.

7.7.9.11.2 IEEE 1588

IEEE 1588 is a per-link function and thus is controlled by the PF driver. The VMs have access to the real time clock register.

7.7.9.11.3 Free Running Timer

The free running timer is a PF driver resource the VMs can access. This register is read only to all VFs and is reset only by the PCI reset.

7.7.9.12 Power Management and Wake-Up

Power management is a PF resource and is not supported per VF.

7.7.9.13 Link Control

The link is a shared resource and as such is controllable only by the PF. This includes interface settings, speed and duplex settings, flow control settings, etc. The flow control packets are sent with the station Ethernet MAC Address stored in the NVM. The watermarks of the flow control process and the time-out value are also controllable by the PF only. In a DCB environment, the parameters of the per TC flow control are also part of the PF responsibilities.

Double VLAN is a network setting and as such should be common to all VFs.

7.7.9.13.1 Special Filtering Options

Pass bad packets is a debug feature. As such, pass bad packets is available only to the PF. Bad packets are passed according to the same filtering rules of the regular packets.

Note: Pass bad packets might cause guest operating systems to get unexpected packets. As a result, it should be used only for debug purposes of the entire system.

Receiving long packets is enabled separately per Rx queue in the RXDCTL registers. As this impacts the flow control thresholds, the PF should be made aware of the decision of all the VMs. Because of this, the setup of TSO packets is centralized by the PF and each VF might request this setting.

7.7.10 Packet Switching

7.7.10.1 Assumptions

The following assumptions are made:

- The required bandwidth for the VM-to-VM loopback traffic is low. That is, the PCIe bandwidth is not congested by the combination of the VM-to-VM and the regular incoming traffic. This case is handled but not optimized for. Unless specified otherwise, Tx and Rx packets should not be dropped or lost due to congestion caused by loopback traffic.
- If the buffer allocated for the VM-to-VM loopback traffic is full, it is acceptable to back pressure the transmit traffic of the same TC. This means that the outgoing traffic might be blocked if the loopback traffic is congested.
- The decision on local traffic is done only according to the Ethernet DA address and the VLAN tag. There is no filtering according to other parameters (IP, L4, etc.). The switch has no learning capabilities. In case of double VLAN mode, the inner VLAN is used for the switching functionality.
- The forwarding decisions are based on the receive filtering programming.
- No packet switching between TCs.
- Coexistence with TimeSync: time stamp is not sampled for any VM-to-VM loopback traffic.
- Coexistence with Double VLAN: When double VLAN is enabled by DMATXCTL.*GDV* and it is expected to transmit untagged packets by software, transmit to receive packet switching should not be enabled.

7.7.10.2 Pool Selection

Pool selection is described in the following sections. A packet might be forwarded to a single pool or replicated to multiple pools. Multicast and broadcast packets are cases of replication, as is mirroring.

The following capabilities determine the destination pools of each packet:

- 128 Ethernet MAC Address filters (RAH/RAL registers) for both unicast and multicast filtering. These are shared with L2 filtering. For example, the same Ethernet MAC Addresses are used to determine if a packet is received by the switch and to determine the forwarding destination. In "E-tag pool select" mode RAL registers bits [13:0] are used for E-tag filtering {GRP, E-CID_base} and pool selection.
- 64 shared VLAN filters (PFVLVF and PFVLVFB registers) — Each VM can be made a member of each VLAN.
- Hash filtering of unicast and multicast addresses (if the direct filters previously mentioned are not sufficient).
- Forwarding of broadcast packets to multiple pools.
- Forwarding by EtherType.
- Mirroring by pool, VLAN, or link.

Receive pool/queue allocation refer to Section 7.1.3.2.

7.7.10.3 Rx Packets Switching

Rx packet switching is the second of three stages that determine the destination of a received packet. The three stages are defined in Section 7.1.3.

As far as switching is concerned, it does not matter whether the X550's virtual environment operates in IOV mode or in VMDq2 mode.

When operating in replication mode, broadcast and multicast packets can be forwarded to more than one pool, and is replicated to more than one Rx queue. Replication is enabled by the *Rpl_En* bit in the PFVTCTL register.

Note: For the following algorithms packets with no E-tag in *PFVTCTL.POOLING_MODE* = 01b (E-tag mode), or all packets in *PFVTCTL.POOLING_MODE* = 00b (MAC mode) are defined as packets with no external tags.

7.7.10.3.1 Replication Mode Enabled

When replication mode is enabled, each broadcast/multicast packet can go to more than one pool. Finding the pool list of any packet is provided in the following steps:

1. **Exact unicast or multicast match** — If there is a match in one of the exact filters (RAL/RAH), for unicast or multicast packets, use the MAC Pool Select Array (MPSAR[n]) bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters. The compared field differs according to the filtering mode (PFVTCTL.*POOLING_MODE*):
   a. If *POOLING_MODE* = 00b (MAC mode) or for packets with no external tags, the Destination MAC Address is compared to {RAH[15:0], RAL[31:0]} for all the registers for which RAH.*ADTYPE* = 0 (MAC)

b. If *POOLING_MODE* = 01b (E-tag mode) and the packet has an E-tag, the E-tag E-PID (as extracted from offset 47:34, starting from the EtherType, in the tag identified by the ETAG_ETYPE register) is compared to RAL[13:0] for all the registers for which RAH.*ADTYPE* = 1 (E-tag)

2. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.
3. **Broadcast** — If the packet is a broadcast packet with no external tag, add pools for which their PFVML2FLT.*BAM* bit (*Broadcast Accept Mode*) is set.
4. **Unicast hash** — If the packet is a unicast packet with no external tag, and the prior steps yielded no pools, check it against the Unicast Hash Table (*PFUTA*). If there is a match, add pools for which their PFVML2FLT.*ROPE* bit (*Accept Unicast Hash*) is set.
5. **Multicast hash** — If the packet is a multicast packet with no external tag and the prior steps yielded no pools, check it against the Multicast Hash Table (MTA). If there is a match, add pools for which their PFVML2FLT.*ROMPE* bit (*Receive Multicast Packet Enable*) is set.
6. **Multicast promiscuous** — If the packet is a multicast packet with no external tag, take the candidate list from prior steps and add pools for which their PFVML2FLT.*MPE* bit (*Multicast Promiscuous Enable*) is set.
7. **Unicast Promiscuous** - If the packet is a unicast packet with no external tag, take the candidate list from prior steps and add pools for which their PFVML2FLT.*UPE* bit (*Unicast Promiscuous Enable*) is set.
8. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.*VFE* bit.
   a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n] or pools for which the PFVML2FLT.*VPE* (*VLAN Promiscuous Enable*) is set.
   b. Untagged packets: enable only pools with their PFVML2FLT.*AUPE* bit set.
   c. If there is no match, the pool list should be empty.

   Note: In a VLAN network, un-tagged packets are not expected. Such packets received by the switch should be dropped, unless their destination is a virtual port set to receive these packets. The setting is done through the PFVML2FLT.*AUPE* bit. It is assumed that VMs for which this bit is set are members of a default VLAN and thus only MAC queuing is done on these packets.
9. **Default Pool** — If the pool list is empty at this stage and the PFVTCTL.*DIS_DEF_POOL* bit is cleared, set the default pool bit in the target pool list (from PFVTCTL.*DEF_PL*).
10. **EtherType filters** — If one of the EtherType filters (ETQF) is matched by the packet and pooling action is requested and the *Pool Enable* bit in the ETQF is set, the pool list is set to the pool pointed to by the filter.
11. **Filter Local Packets (source address pruning)** - The pruning operation depends on the filtering mode (PFVTCTL.*POOLING_MODE*):
    a. If *POOLING_MODE* = 00b (MAC mode) or for packets with no external tag, the Source MAC Address is compared to {RAH[15:0], RAL[31:0]} for all the registers for which RAH.*ADTYPE* = 0 (MAC).
    b. If *POOLING_MODE* = 01b (E-tag mode) and the packet has an E-tag, {00,E-tag ingress-E-CID_base} (as extracted from offset 31:20, starting from the EtherType, in the tag identified by the ETAG_ETYPE register) is compared to RAL[13:0] for all the registers for which RAH.*ADTYPE* = 1 (E-tag).

If there is a match to one of the RAL[n]/RAH[n] clear all the pools that are set in both the matching MPSAR[n] and the PFFLP from the pool list. This prunes multicast packets from getting back to the sender in VEPA mode.

12. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers.
13. **Mirroring** — Each of the four mirroring rules adds its destination pool (PFMRCTL.*MP*) to the pool list if the following applies:
    a. **Pool mirroring** — PFMRCTL.*VPME* is set and one of the bits in the pool list matches one of the bits in the PFMRVM register.
    b. **VLAN port mirroring** — PFMRCTL.*VLME* is set and the index of the VLAN of the packet in the PFVLVF table matches one of the bits in the PFMRVLAN register.
    c. **Uplink port mirroring** — PFMRCTL.*UPME* is set, the pool list is not empty.
14. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers. Note that this stage appears twice to handle mirroring cases.

7.7.10.3.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode, the pool list of any packet always contains one pool only according to the following steps:

1. **Exact unicast or multicast match** — The compared field differs according to the filtering mode (PFVTCTL.*POOLING_MODE*):
    a. If *POOLING_MODE* = 00b (MAC mode) or for packets with no external tags, the Destination MAC Address is compared to {RAH[15:0], RAL[31:0]} for all the registers for which RAH.*ADTYPE* = 0 (MAC)
    b. If *POOLING_MODE* = 01b (E-tag mode) and the packet has an E-tag, the E-tag E-PID (as extracted from offset 47:34, starting from the EtherType, in the tag identified by the ETAG_ETYPE register) is compared to RAL[13:0] for all the registers for which RAH.*ADTYPE* = 1 (E-tag)

    If the compared field matches one of the exact filters (RAL/RAH), use the MAC Pool Select Array (MPSAR[n]) bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters.
2. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.
3. **Unicast hash** — If the packet is a unicast packet with no external tag, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add the pool for which their PFVML2FLT.*ROPE* (*Accept Unicast Hash*) bit is set. Refer to the software limitations described after Step 8.
4. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.*VFE* bit.
    a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n] or pools for which the PFVML2FLT.*VPE* (VLAN Promiscuous Enable) is set.
    b. Untagged packets: enable only pools with their PFVML2FLT.*AUPE* bit set.
    c. If there is no match, the pool list should be empty.

5. **Default Pool** — If the pool list is empty at this stage and the PFVTCTL.*DIS_DEF_POOL* bit is cleared, set the default pool bit in the target pool list (from PFVTCTL.*DEF_PL*).
6. **Multicast or Broadcast** — If the packet is a multicast or broadcast packet with no external tag and was not forwarded in Step 1 and Step 2, set the default pool bit in the pool list (from PFVTCTL.*DEF_PL*).
7. **EtherType filters** — If one of the EtherType filters (ETQF) is matched by the packet and queuing action is requested and the *Pool Enable* bit in the ETQF is set, the pool list is set to the pool pointed by the filter.
8. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers.

The following software limitations apply when replication is disabled:

- Software must not set more than one bit in the bitmaps of the exact filters. Note that multiple bits can be set in an RAH register as long as it's guaranteed that the packet is sent to only one queue by other means (such as VLAN).
- Software must not set per-VM promiscuous bits (multicast or broadcast).
- Software must not set the *ROPE* bit in more than one PFVML2FLT register.
- Software should not activate mirroring.
- Software should not filter out local packets by setting PFFLP bits

7.7.10.4 Tx Packets Switching

Tx switching is used only in a virtualized environment to serve VM-to-VM traffic. Packets that are destined to one or more local VMs are directed back (loopback) to the Rx packet buffers. Enabling Tx switching is done by setting the PFDTXGSWC.*LBE* bit. Tx to Rx switching always avoids packet drop as if flow control is enabled. Therefore, the software must set the FCRTH[n].*RTH* fields regardless if flow control is activated on the X550.

Tx switching rules are very similar to Rx switching in a virtualized environment, with the following exceptions:

- If a target pool is not found, the default pool is used only for broadcast and multicast packets.
- A unicast packet that matches an exact filter is not sent to the LAN.
- Broadcast and multicast packets are always sent to the external LAN.
- A packet might not be sent back to the originating pool (even if the destination address is equal to the source address) unless loopback is enabled for that pool by the PFVMTXSW[n] register.

The detailed flow for pool selection as well as the rules that apply to loopback traffic is as follows:

- Loopback is disabled when the network link is disconnected. It is expected (but not required) that system software (including VMs) does not post packets for transmission when the link is disconnected.
- Loopback is disabled when the *RXEN* (*Receive Enable*) bit is cleared.
- Loopback packets are identified by the *LB* bit in the receive descriptor.

Notes: When Tx switching is enabled, the host must avoid sending packets longer than 9.5 KB.

Tx Switching should be enabled only if PFVTCTL.*POOLING_MODE* = 00b (MAC mode).

7.7.10.4.1 Replication Mode Enabled

When replication mode is enabled, the pool list for any packet is determined according to the following steps:

1. **Exact unicast or multicast match** — If there is a match in one of the exact filters (RAL/RAH), for unicast or multicast packets, take the MPSAR[n] bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters.
2. **Ether-type filters** — If one of the enabled (ETQF.*TX_ANTISPOOF*) Transmit Ether-type filters (ETQF.*ETYPE*) is matched by the packet and the appropriate Pool loopback/anti-spoof bit in the *PFVFSPOOF* is set (PFVFSPOOF.*ETHERTYPELB*/PFVFSPOOF.*ETHERTYPEAS*), the pool list is set to the pool pointed to by the ETQS matching register (loopback) or the packet is dropped (anti-spoof). All the subsequent steps after Step 3 are skipped.

   Note: Loopback on ETQF match is possible even if the general loopback enable bit (PFDTXGSWC.*LBE*) is clear.
3. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.
4. **Broadcast** — If the packet is a broadcast packet, add pools for which their PFVML2FLT.*BAM* bit (*Broadcast Accept Mode*) is set.
5. **Unicast hash** — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add pools for which their PFVML2FLT.*ROPE* bit (*Accept Unicast Hash*) is set.
6. **Multicast hash** — If the packet is a multicast packet and the prior steps yielded no pools, check it against the Multicast Hash Table (MTA). If there is a match, add pools for which their PFVML2FLT.*ROMPE* bit (*Receive Multicast Packet Enable*) is set.
7. **Unicast Promiscuous** — If the packet is a unicast packet, take the candidate list from prior steps and add pools for which their PFVML2FLT.*UPE* bit (*Unicast Promiscuous Enable*) is set.
8. **Multicast Promiscuous** — If the packet is a multicast packet, take the candidate list from prior steps and add pools for which their PFVML2FLT.*MPE* bit (*Multicast Promiscuous Enable*) is set.
9. **Filter source pool —** The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.*LLE* bit is set.
10. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.*VFE* bit.
    a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n] or pools for which the PFVML2FLT.*VPE* (*VLAN Promiscuous Enable*) is set.
    b. Untagged packets: enable only pools with their PFVML2FLT.*AUPE* bit set.
    c. If there is no match, the pool list should be empty.
11. **Forwarding to the network** — Packets are forwarded to the network in the following cases:
    a. All broadcast and multicast packets.
    b. Unicast packets that do not match any exact filter. A match of an exact filter that also points to a pool disabled via PFVFRE is not considered a match.
12. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list (pre mirroring step).

13. **Mirroring** — Each of the following three mirroring rules adds its destination pool (PFMRCTL.*MP*) to the pool list if the following applies:
    a. **Pool mirroring** — PFMRCTL.*VPME* is set and one of the bits in the pool list matches one of the bits in the PFMRVM register.
    b. **VLAN port mirroring** — PFMRCTL.*VLME* is set and the index of the VLAN of the packet in the PFVLVF table matches one of the bits in the PFMRVLAN register.
    c. **Downlink port mirroring** — PFMRCTL.*DPME* is set and the packet is sent to the network.
14. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list (post mirroring step).

7.7.10.4.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode the pool list for any packet always contains one pool only according to the following steps:

1. **Exact unicast or multicast match** — If the packet DA matches one of the exact filters (RAL/RAH), take the MPSAR[n] bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters.
2. **Ether-type filters** - if one of the enabled (ETQF.*TX_ANTISPOOF*) Transmit Ether-type filters (ETQF.*ETYPE*) is matched by the packet and the appropriate Poll loopback/anti-spoof bit in the PFVFSPOOF is set (PFVFSPOOF.*ETHERTYPELB*/PFVFSPOOF.*ETHERTYPEAS*), the pool list is set to the pool pointed to by the *ETQS* matching register (loopback) or the packet is dropped (anti-spoof). All the subsequent steps after Step 3 are skipped.

Note: Loopback on EtherType match is possible even if the general loopback enable bit (PFDTXGSWC.*LBE*) is clear.

3. **PFVFRE** — If any bit in the *PFVFRE* register is cleared, clear the respective bit in the pool list.
4. **Unicast hash** — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add the pool for which their PFVML2FLT.*ROPE* bit (*Accept Unicast Hash*) is set. Refer to the software limitations that follow.
5. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.*VFE* bit.
    a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n] or pools for which the PFVML2FLT.*VPE* (*VLAN Promiscuous Enable*) is set.
    b. Untagged packets: enable only pools with their PFVML2FLT.*AUPE* bit set.
    c. If there is no match, the pool list should be empty.
6. **Multicast or Broadcast** — If the packet is a multicast or broadcast packet and was not forwarded in Step 1 and Step 2, set the default pool bit in the pool list (from PFVTCTL.*DEF_PL*).
7. **Filter source pool —** The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.*LLE* bit is set.
8. **Forwarding to the Network** — Packets are forwarded to the network in the following cases:
    a. All broadcast and multicast packets.
    b. Unicast packets that do not match any exact filter. A match of an exact filter that also points to a pool disabled via PFVFRE is not considered a match.
9. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.

The following software limitations apply when replication is disabled:

1. It is software's responsibility not to set more than one bit in the bitmaps of the exact filters. Note that multiple bits can be set in an RAH register as long as it is guaranteed that the packet is sent to only one queue by other means (such as VLAN)
2. Software must not set per-VM promiscuous bits (multicast or broadcast).
3. Software must not set the *ROPE* bit in more than one PFVML2FLT register.
4. Software should not activate mirroring.

7.7.10.5 Mirroring Support

The X550 supports four separate mirroring rules, each associated with a destination pool (mirroring can be done in up to four pools). Each rule is programmed with one of the four mirroring types:

- **Pool Mirroring** — Reflects all the packets received to a pool from the network.
- **Uplink Port Mirroring** — Reflects all the traffic received from the network that reached any pool.
- **Downlink Port Mirroring** — Reflects all the traffic transmitted to the network.
- **VLAN Mirroring** — Reflects all the traffic received from the network in a set of given VLANs (either from the network or from local VMs).

Note: Reflecting all the traffic received by any of the pools (either from the network or from local VMs) is supported by enabling mirroring of all pools.

Note: To get all traffic irrespective of the filtering rules, the mirroring pool should be set to promiscuous mode and promiscuous VLAN mode.

Note: Mirroring and replication on FCoE traffic is not supported on receive if the ETQF filters define FCoE packets and on transmit if the packets are indicated as FCoE (by setting the *FCoE* bit in the *TUCMD* field in the Transmit Context Descriptor).

Mirroring modes are controlled by a set of rule control registers:

- **PFMRCTL** — Controls the rules to be applied and the destination port.
- **PFMRVLAN** — Controls the VLAN ports as listed in the PFVLVF table taking part in the VLAN mirror rule.
- **PFMRVM** — Controls the pools taking part in the pool mirror rule.

7.7.10.6 Offloads

The general rule is that offloads are executed as configured for the pool and queue associated with the receive packet. Some special cases:

- If a packet is directed to a single pool, offloads are determined by the pool and queue for that packet.
- If a packet is replicated to more than one pool, each copy of the packet is offloaded according to the configuration of its pool and queue.
- If replication is disabled, offloads are determined by the unique destination of the packet.

The following subsections describe exceptions to the previously described special cases.

7.7.10.6.1 Local Traffic Offload

The following capabilities are not supported on the loopback path:

- The RSS and VLAN strip offload capabilities are only supported if the *CC* bit in the transmit descriptor of the packet is set and if it is a tunnel packet, the TUNNEL.*OUTERIPCS* is set also. The reason is that when these bits are not set, software does not provide the necessary offload offsets with the Tx packet.
- Receive Side Coalescing (RSC) is not supported.
- FCoE offloads are not supported.

7.7.10.6.2 Rx Traffic Offload

- CRC offload is a global policy. CRC strip is enabled or disabled for all received packets.

7.7.10.7 Rate Control Features

7.7.10.7.1 Congestion Control

Tx packets going through the local switch are stored in the Rx packet buffer, similar to packets received from the network. Tx to Rx switching always avoids packet drop as if flow control is enabled. Therefore, the software must set the FCRTH[n].*RTH* fields regardless if flow control is activated on the X550.

The X550 guarantees that one TC flow is not affected by congestion in another TC.

Receive and local traffic are provided with the same priority and performance expectations. Packets from the two sources are merged in the Rx packet buffers, which can in general support both streams at full bandwidth. Any congestion further in the pipeline (such as lack of PCIe bandwidth) evenly affects Rx and local traffic.

7.7.10.7.2 Tx Queue Arbitration and Rate Control

To guarantee each pool with adequate bandwidth, a per-pool bandwidth control mechanism is added to the X550. Each Tx pool gets a percentage of the transmit bandwidth and is guaranteed it can transmit within its allocation. This arbitration is combined with the TC arbitration. See additional details on DCB Tx capabilities in Section 7.6.2.1.

7.7.10.7.3 Receive Priority

As the switch might decide to loopback packets from the transmit path to the receive path, in case the receive path is full, the transmit path might be blocked (including the traffic to the LAN). The X550 guarantees that packets are not dropped because of that and that one traffic class flow is not affected by congestion in another traffic class. The PF driver might decide to program the X550 to drop packets from receive queues without available descriptors.

To keep the congestion effect locality, receive traffic from the LAN have higher priority than loop back traffic. This way large loopback traffic does not impact the network.

7.7.10.8 Small Packets Padding

In Virtualized systems, the driver receiving the packet in the VM might not be aware of all the hardware offloads applied to the packet. Thus, in case of stripping actions by the hardware (VLAN strip), it might receive packets which are smaller than a legal packet. The X550 provides an option to pad small packets in such cases so that all packets have a legal size. This option can be enabled only if the CRC is stripped. In these cases, all small packets are padded to 60 bytes (legal packet - 4 bytes CRC). The padding is done with zero data. This function is enabled via the RDRXCTL.*PSP* bit.

Packet padding is done after tag extractions and do not take into account if tags are exposed in the Rx descriptor status.

Notes: FCoE DDP packets are not padded.

The packet length reported in the descriptor reflects the packet length before padding.

If a timestamp is added, it is part of the padding data.

7.7.10.9 Switch Control

The PF driver has some control of the switch logic. The following registers are available to the PF for this purpose:

- **PFVTCTL (PF Virtual Control Register)**
  - ***RPL_EN* (Replication Enable)** — Enables replication of multicast and broadcast packets — both in incoming and local traffic. If this bit is cleared, Tx multicast and broadcast packets are sent only to the network and Rx multicast and broadcast packets are sent to the default pool.
  - ***DEF_PL* (Default Pool)** — Defines the target pool for packets that passed L2 filtering but did not pass any of the pool filters. This field is invalid when the *DIS_DEF_POOL* bit is set.
  - ***DIS_DEF_POOL* (Disable Default Pool)** — Disables acceptance of packets that failed all pool filters.
- **PFVFRE (PF VF Receive Enable)** — Enables/disables reception of packets from the link to a specific VF. Used during initialization of the VF. See Section 4.2.2.2 for more details.
- **PFDTXGSWC (PF DMA Tx General Switch Control)**
  - ***LBE* (Loopback Enable)** — VMDQ loopback enables switching of Tx traffic to the Rx path for VM-to-VM communication.
- **PFVFSPOOF (PFVF Anti-Spoof Control)**
  - ***MACAS* (MAC Anti-Spoof Enable)** — Enables filtering of Tx packet for anti-spoof.
- **PFVMTXSW (PF VM Tx Switch Loopback Enable)**
  - ***LLE* (Local Loopback Enable)** — Local Loopback Enable defines whether to allow loopback of a packet from a certain pool into itself.
- **PFQDE (PF Queue Drop Enable Register)** — A register defining global policy for drop enable functionality when no descriptors are available. It lets the PF override the per-queue SRRCTL[n] *DROP_EN* setting. PFQDE should be used in SR-IOV mode as described in Section 4.6.11.3.1.
- **PFVML2FLT (PF VM L2Control Register)**
  - ***ROMPE* (Receive Overflow Multicast Packets)** — Accept multicast hash. Defines whether a pool accepts packets that match the multicast MTA table.

— ***ROPE* (Receive MAC Filters Overflow)** — Accept unicast hash. Defines whether a pool accepts packets that match the unicast PFUTA table.

— ***BAM* (Broadcast Accept)** — Defines whether a pool accepts broadcast packets.

— ***MPE* (Multicast Promiscuous Enable)** — Defines whether or pool accepts all multicast packets.

— ***AUPE* (Accept Untagged Packets Enable)** — Defines whether a pool accepts untagged VLAN packets.

- **Mirror Control** — See Section 7.7.10.5.
- **PFVFTE (PF VF Transmit Enable)** — Enables/disables transmission of packets to the link to a specific VF. Used during initialization of the VF. See Section 4.2.2.2 for more details.
- **PFVLVF/PFVLVFB (PF VM VLAN Pool Filter/Bitmap)** — VLAN queuing table. A set of 64 VLAN entries with an associated bitmap, one bit per pool. Bits are set for each pool that participates in this VLAN.
- **PFUTA (PF Unicast Table Array)** — A 4 Kb array that covers all combinations of 12 bits from the MAC destination address. A received unicast packet that misses the MAC filters is compared against the PFUTA. If the relevant bit in the PFUTA is set, the packet is routed to all pools for which the *ROPE* bit is set.
- **MTA (Multicast Table Array)** — A 4 Kb array that covers all combinations of 12 bits from the MAC destination address. A received multicast packet that misses the MAC filters is compared against the MTA. If the relevant bit in the MTA is set, the packet is routed to all pools for which the *ROMPE* bit is set.

In addition, the rate-control mechanism is programmed as described in Section 7.6.2.1.

7.7.11 Security Features

The X550 allows some security checks on the inbound and outbound traffic of the switch.

7.7.11.1 Inbound Security

Each incoming packet (either from the LAN or from a local VM) is filtered according to the VLAN tag so that packets from one VLAN cannot be received by pools that are not members of that VLAN.

7.7.11.2 Outbound Security

7.7.11.2.1 MAC Anti-Spoofing

Each pool is associated with one or more Ethernet MAC Addresses on the receive path. The association is determined through the MPSAR registers. The MAC anti-spoofing capability insures that a VM always uses a source Ethernet MAC Address on the transmit path that is part of the set of Ethernet MAC Addresses defined on the Rx path. A packet with a non-matching SA is dropped, preventing spoofing of the Ethernet MAC Address. This feature is enabled in the PFVFSPOOF.*MACAS* field, and can be enabled per Tx pool.

Note: Anti-spoofing is not available for VMs that hide behind other VMs whose Ethernet MAC Addresses are not part of the RAH/RAL Ethernet MAC Address registers. In this case, anti-spoofing should be done by software switching, handling these VMs.

7.7.11.2.2 VLAN Anti-Spoofing

Each pool is associated with one or more VLAN tags on the receive path. The association is determined through the PFVLVF and PFVLVFB registers. The VLAN anti-spoofing capability insures that a VM always uses a VLAN tag on the transmit path that is part of the set of VLAN tags defined on the Rx path. A packet with a non-matching VLAN tag is dropped, preventing spoofing of the VLAN tag. This feature is enabled in the PFVFSPOOF.*VLANAS* field, and can be enabled per Tx pool.

Notes: If VLAN anti-spoofing is enabled, MAC anti-spoofing must be enabled as well.

When double VLAN is enabled by DMATXCTL.*GDV* and it is expected to transmit untagged packets by software, VLAN anti-spoofing should not be enabled.

Un-tagged packets are not checked by VLAN anti-spoofing.

7.7.11.2.3 VLAN Tag Validation

In PCI-SIG IOV scenarios the driver might be malicious, and thus might fake a VLAN tag. The X550 provides the ability to force a specific VLAN value for a VM. The possible behaviors are controlled by the PFVMVIR[n] registers as follows:

- **Use descriptor value** — To be used in case of a trusted VM that can decide which VLAN to send. This option should also be used in case one VM is member of multiple VLANs.
- **Always insert default VLAN** — This mode should be used for non-trusted or non-VLAN aware VMs. In this case, any VLAN insertion command from the VM is ignored. If a packet is received with a VLAN, the packet should be dropped.
- **Never insert VLAN** — This mode should be used in a non-VLAN network. In this case, any VLAN insertion command from the VM is ignored. If a packet is received with a VLAN, the packet should be dropped.

Notes: The VLAN insertion settings should be done before any of the queues of the VM are enabled.

When double VLAN is enabled by DMATXCTL.*GDV* and it is expected to transmit untagged packets by software, VLAN validation should not be enabled.

7.7.11.2.4 E-tag Insertion

In addition to the VLAN insertion described above, the device can add an E-tag to the packets sent by a VF. The PFVMVIR[n].*TAGA* bit defines if a tag should be added. If this bit is set, the added tag is as follows:

- *TAGA* = 00b (MAC mode) or 11b (Reserved): Insert no tag.
- *TAGA* = 01b (E-tag mode): An E-tag is added as follows:

  {TAG_ETYPE.*ETAG_ETHERTYPE[15:0]*, PFVMTIR.*PORT_TAG_ID[31:0]*, 0x0000}

Notes: The E-tag insertion settings should be done before any of the queues of the VM are enabled.

7.7.11.2.5 Ether-type Anti-Spoofing/Pruning

Each pool can be set to drop packets associated with one or more (up to 8) Ether-types on the transmit path. The Ether-types are defined using the ETQF registers. The Ether-type pruning (anti-spoof) capability ensures that a VM does not send out LLDP/ECP control frames to external switches. The Ether-type loopback capability enables the loopback of link layer packets to a pre defined pool destination.

Ether-type pruning (anti-spoof) feature is enabled in the PFVFSPOOF.*ETHERTYPEAS* field, and can be enabled per transmit pool. Ether-type loopback feature is enabled in the PFVFSPOOF.*ETHERTYPELB* field, and can be enabled per transmit pool.

7.7.11.3 Malicious Driver Detection

The hardware can be programmed to take some action as a result of some misbehavior of a VM. These actions might hint to the fact that some VM is malicious and the VMM should remedy the situation. To inform the VMM of this fact, The Mailbox bit (EICR.*MAILBOX* bit) may be set to indicate the occurrence of such behavior. The LMVM_RX and LMVM_TX register contains information on which queue (LMVM_TX.*LAST_Q* or LMVM_RX.*LAST_Q*) and port (LMVM_TX.*MAL_PF* and LMVM_RX.*MAL_PF*) the malicious behavior was detected. The LVMMC_TX and LVMMC_RX registers contain information on the type of error detected and are clear by read.

Malicious driver behavior detection is enabled by setting the DMATXCTL.*MDP_EN* and RDRXCTL.*MDP_EN* bits to 1. This capability should be enabled before queues are exposed to non trusted virtual machines.

On detection of a malicious driver event the X550 stops activity of the offending queue, and sets the bit matching the offending queue in WQBR_RX and WQBR_TX registers. If DMATXCTL.*MBINTEN* or RDRXCTL.*MBINTEN* are set, the X550 generates an interrupt for transmit or receive errors respectively by asserting the EICR.*MAILBOX* bit. Cause of Malicious driver activation is reported in the LVMMC_TX and LVMMC_RX registers. To re-activate offending queue, driver should release it by setting the matching bit in WQBR_RX or WQBR_TX register.

The WQBR_RX and WQBR_TX registers keeps track of all the queues on which a malicious activity was detected and can be used by the driver to track multiple events.

After the queue is disabled the initialization flow defined in Section 4.6.7.1 for receive queues and in Section 4.6.8.1 for transmit queues should be applied.

7.7.11.3.1 Queue Context Validation

The X550 checks that the queue context submitted when a queue is enabled is valid. It also prevents change to static configuration while the queue is enabled. These checks are done both for receive and transmit queues. The table below describes the checks done on the queue context:

Check type	Description	Bit in LVMMC_TX/RX	Action
Receive queue context validity	Check that: • RDLEN >0 • SRRCTL.BSIZEPACKET > 0 • SRRCTL.BSIZEHEADER > 0 if DESCTYP requires header split • SRRCTL.BSIZEHEADER <= 1024 • SRRCTL.DESCTYPE is valid	INVALID_RXQ_CONTEXT	Prevent Enable of queue
Receive queue context change on the fly	Attempt to write RDBAL, RDBAH, RDLEN, or SRRCTL while queue is enabled		Disable queue.
Transmit queue context validity	Check that: • TDLEN >0	INVALID_TXQ_CONTEXT	Prevent Enable of queue
Transmit queue context change on the fly	Attempt to write TDBAL, TDBAH, TDLEN, TDWBAL, or TDWBAH while queue is enabled		Disable queue.

Note: Queue zero is enabled by default. Thus, if malicious driver protection is enabled, to change Queue zero configuration, the queue should be disabled.

7.7.11.3.2 Transmit Descriptor Validity Checks

The table below describes the checks are done by the X550 to define if a transmit packet descriptor is valid. All the checks are done on the descriptors. The checks on the packet header are described in the previous sections.

Table 7-69. Malicious Driver - TX Descriptor Checks

Check type	Description	Bit in LVMMC_TX	Action
Mac Header size	Checks that the MAC header size in the context descriptor is at least 14 (or 18 in case of offloaded packet and double VLAN).	MAC_HEADER	Drop Packet and stop offending queue
IPV4 Header	If a checksum or TSO offload is required, checks that the IPv4 header size in the context descriptor is at least 20.	IPV4_HEADER	Drop Packet and stop offending queue
IPV6 Header	If a TSO offload is required, checks that the IPv6 header size in the context descriptor is at least 40.	IPV6_HEADER	Drop Packet and stop offending queue
Wrong MAC_IP	Check that the MAC+ IP (+ OUTERIP + TUNNEL) header size is not bigger than the packet size.	WRONG_MAC_IP	Drop Packet and stop offending queue
TCP header size	If a TCP TSO offload is required, checks that the TCP header size in the context descriptor is at least 20.	TCP_LSO	Drop Packet and stop offending queue
UDP header size	If a UDP TSO offload is required, checks that the TCP header size in the context descriptor is at least 8.	UDP_LSO	Drop Packet and stop offending queue
SCTP data size	If a SCTP checksum offload is required, checks that the SCTP L4 packet size (including header and data) is at least 12.	STCP_CS	Drop Packet and stop offending queue
Packet too big	In case of a single send, check that the packet is not larger than 15.5KBytes.	SIZE	Drop Packet and stop offending queue
Packet too small	Check that the total length of the packet transmitted, not including FCS is at least 13 bytes.	N/A	Silently drop packet.
Illegal offload request.	Check that TSO is no requested for SCTP or that a checksum offload is not requested for IPv6 packets.	OFF_ILL	Drop Packet and stop offending queue
Illegal IPsec offload request by VF	Check that a VF had not requested IPsec offload	IPSEC_OFFLOAD	Drop Packet and stop offending queue
Illegal FCoE offload request by VF	Check that a VF had not requested FCoE offload	FCOE_OFFLOAD	Drop Packet and stop offending queue
SCTP alignment	If an SCTP CRC offload is requested, check that the data size is 4 byte aligned.	SCTP_ALIGNED	Drop Packet and stop offending queue
Zero MSS	Check that the MSS size is larger than zero.	ZERO_MSS	Drop Packet and stop offending queue
Context in middle of packet	Check that a context descriptor is not sent in the middle of a packet.	CONTEXT_IN_PACKET	Drop Packet and stop offending queue
Number of large send header buffers	Check that the Large send header is contained in at most 4 buffers.	LSO_MORE_THAN_4	Drop Packet and stop offending queue
Buffers size and length match - Single Send	For single send, check that the total of all buffers size and the packet length match.	OOS_SSO	Drop Packet and stop offending queue
Buffers size and length match - Large Send	For LSO, check that the total of all buffers size and the packet length match.	OOS_LSO	Drop Packet and stop offending queue
LSO wrong header size	For LSO, check that the size of the header fetched actually match the expected header size.	N/A	Silently drop packet. Counted in SSVPC
UDP data size	If a UDP checksum offload is required, checks that the UDP L4 packet size in the context descriptor is at least 8.	SSO_UDP	Drop Packet and stop offending queue

Table 7-69. Malicious Driver - TX Descriptor Checks [continued]

Check type	Description	Bit in LVMMC_TX	Action
TCP data size	If a TCP checksum offload is required, checks that the TCP L4 packet size in the context descriptor is at least 20.	SSO_TCP	Drop Packet and stop offending queue
Descriptor Type	Check that only descriptor types 2 (context) or 3 (advanced data descriptor) are used.	DESC_TYPE	Drop Packet and stop offending queue
CC bit not set when needed	CC (Check Context) bit in descriptor is not set in virtualization mode.		Drop Packet and stop offending queue
Null packet check	Check that a Null packet has the *EOP* bit set.	WRONG_NULL	Drop Packet and stop offending queue
Packet without EOP	Check that a only entire packets are provided by the driver	NO_EOP	Drop Packet and stop offending queue
Burst of contexts	Check that less than 4 Contiguous context descriptor are sent by the driver.	CONTEXT_BURST	Drop Packet and stop offending queue
Legacy Descriptor in SR-IOV	Legacy descriptor used when SR-IOV is enabled.	LEGACY_DESC_IOV	Drop Packet and stop offending queue

7.7.11.3.3 Reactive Malicious Behavior Detection

The table below describes the checks are done by the X550 to detect a malicious behavior, even if the packet seems valid.

Table 7-70. Reactive Malicious Checks

Check type	Description	Bit in LVMMC_TX/RX	Action
Malicious VF memory access	A PCIe DMA access initiated by a VF ended with Unsupported Request (UR) or Completer Abort (CA). This check is done for both Tx and Rx queues.	INV_MACC	Drop Packet and stop offending queue
DMA access outside of active memory [1]	A Queue attempt to read memory outside of the active memory range of the system. See Section 3.1.5.7 for details.	INV_MACC	Drop PCIe read transaction. Stop queue that requested the transaction
	A Queue attempt to write memory outside of the active memory range of the system. See Section 3.1.5.7 for details.	N/A	Silently drop PCIe write transaction.
VLAN not expected in packet	A packet where VLAN should be added by device is in Tx descriptor. See Section 7.7.11.2.3	VLAN_IERR	Drop Packet
Anti-spoof checks	See Section 7.7.11.2.1 and Section 7.7.11.2.2 and Section 7.7.11.2.5 for details	MAC_VLAN_SPOOF	Drop Packets.

1. A PCIe error interrupt can be asserted when such a transaction is dropped. See Section 3.1.5 for details.

7.7.12 Virtualization of Hardware

This section describes additional features used in both IOV and VMDq2 modes.

7.7.12.1 Per-Pool Statistics

Part of the statistics are by definition shared and cannot be allocated to a specific VM. For example, CRC error count cannot be allocated to a specific VM, as the destination of such a packet is not known if the CRC is wrong.

All the non-specific statistics are handled by the PF driver in the same way it is done in non-virtualized systems. A VM might request a statistic from the PF driver but might not access it directly.

The conceptual model used to gather statistics in a virtualization context is that each queue pool is considered as a virtual link and the Ethernet link is considered as the uplink of the switch. Thus, any packet sent by a pool is counted in the Tx statistics, even if it was forwarded to another pool internally or was dropped by the MAC for some reason. In the same way, a replicated packet is counted in each of the pools receiving it.

The following statistics are provided per pool:

- Good packet received count
- Good packet transmitted count
- Good octets received count
- Good octets transmitted count
- Multicast packets received count

Note: All the per VF statistics are read only and wrap around after reaching their maximum value.

7.8 Tunneling Support

The X550 supports the VXLAN and NVGRE tunneling packet formats. As part of this support the following features are supported:

- LSO and transmit checksum offloads for tunneled packets as described in Section 7.2.4 and Section 7.2.5.
- RSS forwarding based on inner L3/L4 header as described in Section 7.1.3.7
- Flow director forwarding based on Tenant ID, inner MAC and inner VLAN as described in Section 7.1.3.6
- New indications in the receive descriptor with information on the tunnel headers and on the outer header checksum as described in Section 7.1.5.2
- New packet split modes based on the tunnel header or outer L2 header as described in Section A.2.5

7.9 Receive Side Coalescing (RSC)

The X550 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X550 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

The X550 digests received packets and categorizes them by their TCP/IP connections (flows). For each flow, hardware coalesces the packets as shown in Figure 7-41 and Figure 7-42 (the colored parameters are explained in the RSC context table and receive descriptor sections). The X550 can handle up to 32 concurrent flows per LAN port at any given time. Each flow handled by RSC offload has an associated context. The X550 opens and closes the RSC contexts autonomously with no need for any software intervention. Software needs only to enable RSC in the selected receive queues.

Note: When RSC is enabled, advanced receive descriptors should be used and CRC strip should be enabled.

Figure 7-41 shows a top level flow diagram that is used for RSC functionality. The following sections provide a detailed explanation of this flow as well as the memory structures and device settings that support the RSC functionality.

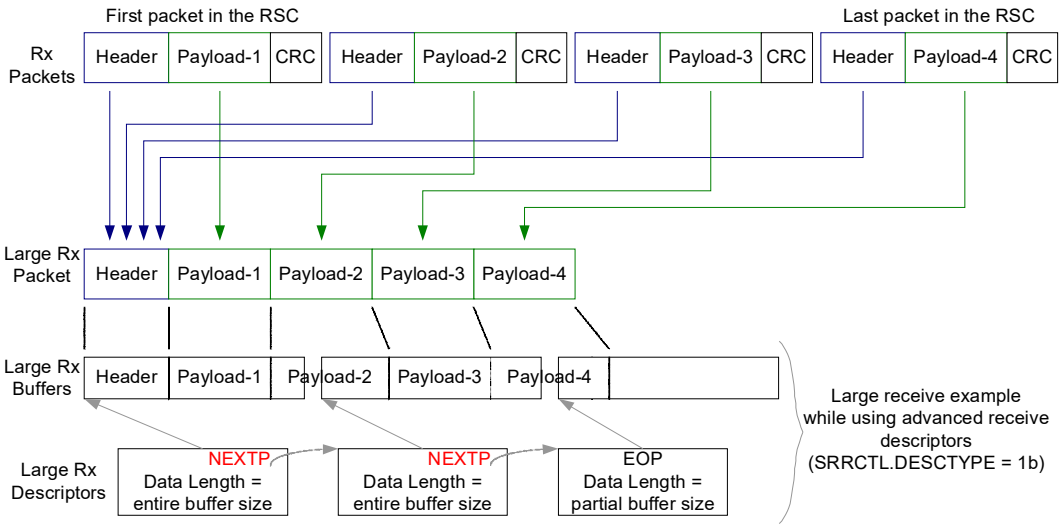


Figure 7-41. RSC Functionality (No Header Split)

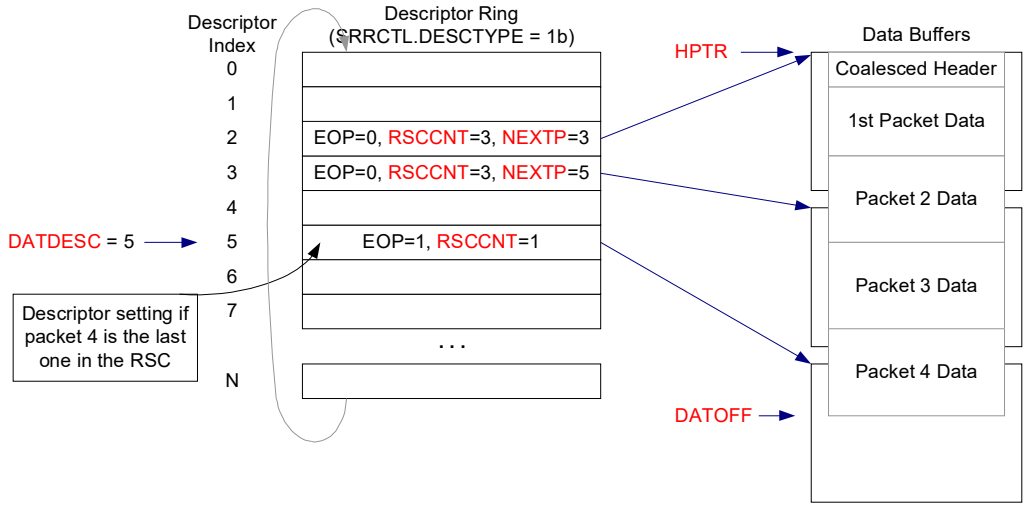


Figure 7-42. RSC Functionality (No Header Split)

Note: Software might abort reception to any queue at any time. For example: VFLR or queue disable. Following these settings, hardware aborts further DMA(s) and descriptor completions. Specifically, active RSC(s) in the specific queue(s) are not completed. In such cases there could be completed packets and RSC(s) hidden from software by prior incomplete RSC(s).

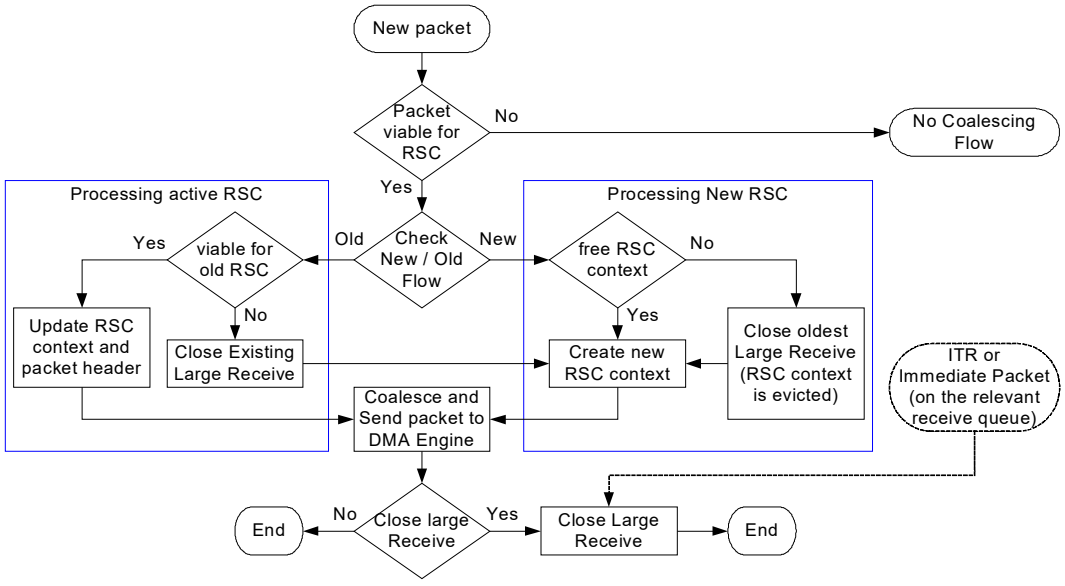


Figure 7-43. RSC Event Flow

7.9.1 Packet Candidacy for RSC

Incoming packets can be good candidates for RSC offload when the following conditions are met. If any of the these conditions are not met, and assuming the queue is configured as required in Section 4.6.7.2 the received packet is processed in the legacy (non-coalescing) scheme.

- RSC is enabled in the destination receive queue by the RSCCTL.*RSCEN*. In this case, software must set the SRRCTL.*DESCTYPE* field in the relevant queues to advanced descriptor modes.
- RSC is further enabled by the RSCINT.*RSCEN* for the receive queues associated to the interrupts defined by the RSCINT registers.
- The SRRCTL[n].*BSIZEHEADER* (header buffer size) must be larger than the packet header (even if header split is not enabled). A minimum size of 128 bytes for the header buffer addresses this requirement.
- The received packet has no MAC errors and no TCP/IP checksum errors. MAC errors are:
  - CRC error.
  - Undersize frame received.
  - Oversize frame received.
  - Error control byte received in mid-packet.
  - Illegal code byte received in mid-packet.
- If the *Length* field in the IP header does not cover the entire packet (as the case for padding bytes), the received packet is not a candidate for RSC.
- The packet type is TCP/IPv4 (non-SNAP) with optional VLAN header(s). RSC is not supported for IPv6 packets.
- The packet is not an NVGRE or VXLAN packet.
- IP header does not carry any option headers.
- See Section 4.6.7.2 for the required configuration to enable or disable NFS coalescing.
- The TCP segment is not fragmented.
- The following TCP flags are inactive: FIN, SYN, RST, URG, ECE, CWR, NS and the other three reserved TCP flags (see TCP Flags mapping in Table 7-72).
- The *ECT* and *CE* bits in the *TOS* field in the IP header are not equal to 11b (see the flags in Table 7-73).
- Packets with PSH TCP flag are coalesced but also close the large receive.
- The packet does not carry any TCP option headers.
- RSC is not supported for a switched packet transmitted from a local VM.
- When a Rx packet is replicated or mirrored, it can be coalesced only on the Rx queue that belongs to the source VM.
- Note that there are no limitations on the maximum packet length including jumbo packets.
- If there is already an active RSC for the matched flow, a few additional conditions should be met as listed in Section 7.9.4.

The supported packet format is as follows:

Table 7-71. Packet Fields

Size	Packet Fields
6 bytes	Destination Ethernet MAC Address.
6 bytes	Source Ethernet MAC Address.
[4/6/8 bytes]	Optional External Tag (VLAN or E-tag).
[4 bytes]	Optional VLAN.
2 bytes	Ethernet type field equals 0x0800 (MS byte first on the wire).
20 bytes	IPv4 header with no options.
20 bytes	Basic TCP header (no options — refer to the rows that follow).
[10 bytes]	Optional TCP time stamp header: 1 byte Kind 0x08 1 byte Length 0x0A 4 bytes TS value variable 4 bytes TS echo reply variable
[1 byte]	Optional TCP no operation header: 1 byte Kind 0x01
[1 byte]	Optional TCP end of option header list: 1 byte Kind 0x00
Variable length	TCP payload (RSC candidate must have payload size greater than zero).

Table 7-72. Packet Format Supported by RSC

11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN

Table 7-73. IP TOS Field — Bit Map

7	6	5	4	3	2	1	0
TOS (DS)						ECT	CE

Table 7-74. TCP Time-Stamp Option Header (RFC 1323)

1 Byte: First on the Wire	1 Byte	4 Bytes	4 Bytes: Last on the Wire
Kind = 0x8	Length = 10	TS Value (TSval)	TS Echo Reply (TSecr)

7.9.2 Flow Identification and RSC Context Matching

TCP/IP packet's flow is identified by its four tuples: Source/Destination IP Addresses and Source/ Destination TCP port numbers. These tuples are compared against the *Flow Identification* fields stored in the active RSC contexts (listed in Table 7-75). Comparison is done in two phases:

- Hash compare — Hardware computes a hash value of the four tuples for each flow. The hash value is stored in the RSC context table. It is used for silicon optimization of the compare logic. The hash value of the incoming packet is compared against the hash values of all RSC contexts. No match between the two hash values means that there is no valid context of the same flow.
- Perfect Match — Hardware checks the four tuples of the RSC context that passed the first step with the received frame.
  - A match between the two means that an active RSC context is found.
  - Mismatch between the two indicates a hash collision, which causes a completion of the collided RSC.
- In any case of context mismatch, a new context might be opened as described in Chapter 7.9.3.
- If the packet's flow matches an active RSC context, the packet might be appended to the existing RSC as described in Chapter 7.9.4.

Table 7-75. RSC Context

Size	Name	Description
Flow Identification[1]		
1 bit	CVALID	Context valid indication. Set to 1b by hardware when a new context is defined. Cleared to 0b when RSC completes.
1 byte	CHASH	Context hash value (logic XOR of all bytes of the four tuples).
16 bytes	IPDADDR	IP destination address (set to zero for inactive context).
16 bytes	IPSADDR	IP source address (set to zero for inactive context).
1 bit	IP4TYPE	Defines IP version type (set to 1b for IPv4).
2 bytes	TCPDPORT	TCP destination port.
2 bytes	TCPSPORT	TCP source port.
37 bytes		Total.
RSC Header[2]		
2 bytes	RSCIPLEN	Total *Length* field in the IP header defines the size of the IP datagram (IP header and IP payload) in bytes. Dynamic parameter updated by each received packet.
5 bits	IPOFF	The word offset of the IP header within the packet that is transferred to the DMA unit.
1 bit	RSCTS	TCP time stamp header presence indication.
1 bit	RSCACK	*ACK* bit in the TCP header is a dynamic parameter taken from the last coalesced packet.
1 bit	RSCACKTYPE	ACK packet type indication (*ACK* bit is set while packet does not has TCP payload).
1 bit	RSCPSH	*PSH* bit in the TCP Header is a dynamic parameter which is a logic OR function of the *PSH* bits in all packets within the Large Receive
2 bits	CE, ECT	*ECN* bits in the IP.TOS header: *CE* and *ECT.*
4 bytes	RSCSEQ	Non-RSCACKTYPE case: Expected sequence number in the TCP header of the next packet. RSCACKTYPE case: The ACK sequence number in the last good packet. Dynamic parameter updated by each received packet.

Table 7-75. RSC Context [continued]

Size	Name	Description
8 bytes		Total.
DMA Parameters		
7 bits	RXQUEUE	Receive queue index. This parameter is set by the first packet in the RSC and expected to be the same for all packets in the RSC.
4 bits	RSCDESC	Remaining descriptors of this context. The device initialized RSCDESC by the *MAXDESC* field in the RSCCTL register of the associated receive queue.
4 bits	RSCCNT	Count the number of packets that are started in the current descriptor. The counter starts at 0x1 for each new descriptor. RSCCNT stops incrementing when it reaches 0xF.
8 bytes	HPTR	Header buffer pointer defines the address in host memory of the large receive header (see Section 7.9.5.2).
2 bytes	DATDESC	Data descriptor is the active descriptor index. Initialized by the first packet in the RSC to the first descriptor. It is updated to the active descriptor at a packet DMA completion.
2 bytes	DATOFF	Offset within the data buffer. The data of the first packet in a large receive is the same as the legacy (non-coalescing) definition. Following a DMA completion, it points to the beginning of the data portion of the next packet.
13 bytes		Total.

1. These parameters are extracted from the first packet that opens (activate) the context.
2. All parameters are set by the first packet that opens the context while some are dynamic.

7.9.3 Processing New RSC

Defining the RSC context parameters activates a new large receive. If a received packet does not match any active RSC context, the packet starts (opens) a new one. If there is no free context, the oldest active large receive is closed and its evicted context is used for the new large receive.

7.9.3.1 RSC Context Setting

The X550 extracts the flow identification and RSC header parameters from the packet that opens the context (the first packet in a large receive that activates an RSC context). The context parameters can be divided into categories: flow identification; RSC header and DMA parameters.

7.9.4 Processing Active RSC

Received packets that belong to an active RSC can be added to the large receive if all the following conditions are met:

- The L2 header size equals the size of previous packets in the RSC as recorded in the internal IPOFF parameter in the RSC context table.
- The packet header length as reported in the *HDR_LEN* field is assumed to be the same as the first packet in the RSC (not checked by hardware).
- The *ACK* flag in the TCP header is equal to the *RSCACK* bit in the RSC context (The value of the *ACK* flag should be constant in all the coalesced packets).
- The packet type remains the same as indicated by the *RSCACKTYPE* bit in the RSC context. Packet type can be either pure ACK packet (with no TCP payload) or other.

- For non-RSCACKTYPE (packet with TCP payload), the sequence number in the TCP header matches the expected value in the RSC context (RSCSEQ).
- For RSCACKTYPE, the Acknowledgment number in the TCP header is greater than the RSCSEQ number in the RSC context. Note that the X550 does not coalesce duplicated ACK nor ACK packets that only updates the TCP window.
- ECN handling: The value of the *CE* and *ECT* bits in the IP.*TOS* field remains the same as the RSC context and different than 11b.
- The target receive queue matches the RXQUEUE in the RSC context.
- The packet does not include a TCP time stamp header unless it was included on the first packet that started the large receive (indicated by the RSCTS). Note that if the packet includes other option headers than time stamp, NOP or End of option header, the packet is not processed by RSC flow at all.
- The packet fits within the RSC buffer(s).
- If the received packet does not meet any of the above conditions, the matched active large receive is closed. Then hardware opens a new large receive by that packet. Note that since the X550 closes the old large receive it is guaranteed that there is at least one free context.

Note: See Section 7.1.6.2 for impact of Time Stamp in packet on RSC decisions.

If the received packet meets all the above conditions, the X550 appends this packet payload to the active large receive and updates the context and header as follows. The packet is then DMA'ed to the RSC buffers (as described in Section 7.9.5).

- Update the TCP PSH: The *PSH* bit in the Large Receive context gets the value of the *PSH* bit in the TCP header of the received packet.
- Update the expected sequence number for non-RSCACKTYPE: The RSCSEQ in the large receive context is increased by the value of the TCP payload size of the received packet.
- Update the expected sequence number for RSCACKTYPE: The RSCSEQ in the large receive context is updated to the value of the ACK sequence number field in the received packet.
- Update the total length: The RSCIPLEN in the large receive context is increased by the value of the TCP payload size of the received packet. The value of the *Total Length* field in the IP header in the received packet gets the updated RSCIPLEN. Note that in RSCACKTYPE packets the received payload size is zero.
- Acknowledgment number: If the ACK flag is set, the *Acknowledgment* number field in the header is updated.
- IP header checksum is modified to reflect the changes in the *Total Length* field as follows (note that there is no special process for RSCACKTYPE packets):

  1's {(RSCIPLEN — Packet total length) + 1's (Packet IP header checksum)}

  where:

  — Packet total length is the total length value in the received packet.

  — Packet IP header checksum stands for the IP header checksum field in the received packet.

  — 1's operation defines a ones complement.

  — Plus (+) operation is a cyclic plus while the carry out is fed as a carry in.
- TCP header checksum is left as is in the first packet in the RSC and is set to zero on any succeeding packets.

- Update the DMA parameters.
  - The RSCCNT is initialized to 0x1 on each new descriptor. It is then increased by one on each packet that starts on the same descriptor as long as it does not exceed a value of 0xF. When the RSCCNT is set to 0xF (14 packets) the RSC completes.
  - Decrement by one the Remaining Descriptors (RSCDESC) for each new descriptor.
  - Update the receive descriptor index (DATDESC) for each new descriptor.
  - Update the offset within the data buffer (DATOFF) at the end of the DMA to its valid value for the next packet.
- All other fields are kept as defined by the first packet in the large receive.

7.9.5 Packet DMA and Descriptor Write Back

The Figure 7-44 shows a top view of the RSC buffers using advanced receive descriptors and header split descriptors.

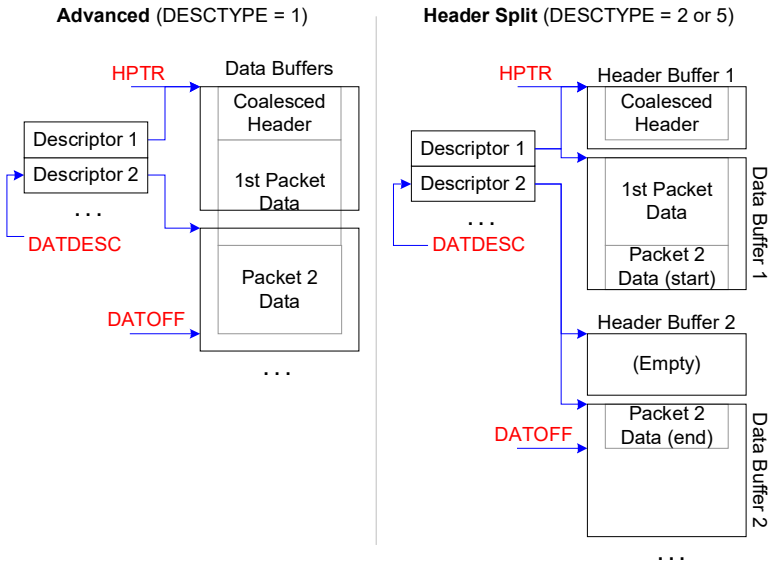


Figure 7-44. RSC — Header and Data Buffers

7.9.5.1 RSC Descriptor Indication (Write Back)

After receiving each packet, the X550 posts the packet data to the data buffers and updates the coalesced header in its buffer. Any completed descriptor is indicated (write back) by setting the fields listed in the following table. A descriptor is defined as the last one when an RSC completes. Section 7.9.6 summarizes all the causes for RSC completion. Any other descriptor in the middle of the RSC is indicated (write back) when the hardware requires the next descriptor so it can report the *NEXTP* field explained in the table that follows.

Fields on the Last Descriptors of Large Receive	Fields on All Descriptors Except for the Last One
EOP: End of packet, and all other fields that are reported together with the EOP.	**NEXTP**: Points to the next descriptor of the same large receive.
DD: indicates that this descriptor is completed by the hardware and can be processed by the software. The descriptor and the header of the packet may still be updated by the hardware.	
RSCCNT: indicates the number of coalesced packets in this descriptor.	

7.9.5.2 Received Data DMA

On the first packet of a large receive, the entire packet is posted to its buffers in host memory. On any other packet, the packet's header and data are posted to host memory as detailed in Section 7.9.5.3 and Section 7.9.5.4.

7.9.5.3 RSC Header

The RSC header is stored at the beginning of the first buffer when using advanced receive descriptors, or at the header buffer of the first descriptor when using header split descriptors (it is defined by the internal HPTR parameter in the RSC context). See Figure 7-44 for more details.

The packet's header is posted to host memory after it is updated by the RSC context as follows:

- **Packets with payload coalescing (RSCACKTYPE=0)** — The TCP sequence number is taken from the TCP context (it is taken from the first packet). The *Total Length* field in the IP header is taken from the RSC context (it represent the length of all coalesced packets). The IP checksum is re-calculated. The TCP checksum is set to zero.
- **ACK no payload coalescing (RSCACKTYPE=1)** — The received packet header is posted as is to host memory. Note that if the received packet includes padding bytes, these bytes are posted in the host memory as well.

7.9.5.4 Large Receive Data

The data of a coalesced packet is posted to its buffer by the DMA engine as follows:

- Ethernet CRC.
  - — When RSC is enabled on any queue, the global CRC strip must be set (HLREG0.*RXCRCSTRP* =1b).
- Packet data spans on a single buffer.
  - — The data of the received packet spans on a single buffer if buffer has the required space.
  - — The DMA engine posts the packet data to its buffer pointed to by DATDESC descriptor at an offset indicated by the DATOFF.

- Packet data spans on multiple buffers.
  - The data of the received packet spans across multiple buffers when it is larger than a single buffer or larger than the residual size of the current buffer.
  - When a new buffer is required (new descriptor) the DMA engine writes back to the completed descriptor linking it to the new one (Section 7.9.5.1 details the indicated descriptor fields).
  - Decrement the RSCDESC parameter by one and update the DATDESC for each new opened descriptor.
- DMA completion.
  - Following DMA completion, set the DATOFF to the byte offset of the next packet.
  - If the PSH TCP flag is active in the coalesced packet, the large receive is completed.
- Not enough descriptors in the receive ring buffer.
  - If the SRRCTL[n].*DROP_EN* bit on the relevant queue is set, The large receive completes and the new packet is discarded.
  - Otherwise (the *DROP_EN* bit is cleared), the packet waits inside the internal packet buffer until new descriptors are added (indicated by the relevant Tail register).
- Not enough descriptors due to RSCDESC exhaust.
  - If the received packet requires more descriptors than indicated by the internal RSCDESC parameter, the X550 completes the current large receive while the new packet starts a new large receive.

7.9.6 RSC Completion and Aging

This section summarizes all causes of large receive completion (the first three cases repeat previous sections).

- A packet of a new flow is received while there are no free RSC contexts. The X550 completes (closes) the oldest large receive (opened first). The new packet starts a new large receive using the evicted context.
- Packets with PSH TCP flags can be only the last ones in active large receive.
- The received packet cannot be added to the active large receive due to one of the following cases (indicated also in Section 7.9.4). In these cases the existing RSC completes and the received packet opens a new large receive.
  - The sequence number does not meet expected value.
  - The receive packet includes a time stamp TCP option header while there was no time stamp TCP option header in the first packet in the RSC.
  - There is not enough space in the RSC buffer(s) for the packet data. Meaning, the received packet requires a new buffer while the RSC already exhausted all permitted buffers defined by the RSCCTL[n].*MAXDESC*.
  - The received packet requires a new buffer while its descriptor wraps around the descriptor ring.
- When a packets is received while there are no more descriptors in the receive queue and the SRRCTL.*DROP_EN* bit is set, the large receive completes and the new packet is discarded.

- EITR expiration while interrupt is enabled — RSC completion is synchronized with interrupt assertion to the host. It enables software to process the received frames since the last interrupt. See more details and EITR setting in Section 7.3.2.1.1.
- EITR expiration while interrupt is disabled — The ITR counter continues to count even when its interrupt is disabled. Every time the timer expires it triggers RSC completion on the associated Rx queues.
- Low number of available descriptors — Whenever crossing the number of free Rx descriptors, the receive descriptor minimum threshold size defined in the SRRCTL[n] registers.
- Interrupt assertion by setting the EICS register has the same impact on packet reception as described in Section 7.3.1.4.1.

Note: In some cases packets that do not meet coalescing conditions might have active RSC of the same flow. As an example: received packets with ECE or CWR TCP flags. Such packets bypass completely the RSC logic (posted as single packets), and do not cause a completion of the active RSC. The active RSC would eventually be closed by either reception of a legitimate packet that is processed by the RSC logic but would not have the expected TCP sequence number. Or, an interrupt event closes all RSC's in its Rx queue. When software processes the packets, it gets them in order even though the RSC completes after the previous packet(s) that bypassed the RSC logic.

Any interrupt closes all RSC's on the associated receive queues. Therefore, when ITR is not enabled any receive packet causes an immediate interrupt and receive coalescing should not be enabled on the associated Rx queues.

Software Note: Whenever an interrupt is generated all active Large Receives on the relevant queues are completed. The Large Receives are indicated to host memory before the interrupt is asserted.

7.10 Fibre Channel over Ethernet (FCoE)

7.10.1 Introduction

Fibre Channel (FC) is the predominant protocol used in Storage Area Networks (SAN). Fibre Channel over Ethernet (FCoE) is used to connect an Ethernet storage initiator and legacy FC storage targets.

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 in such a way that FC-like no-drop is guaranteed as required by FC. Doing so, FC protocol can be transposed to an Ethernet link by Layer 2 encapsulation that is defined by the FCoE protocol. Figure 7-45 shows a connection between an FCoE initiator and legacy FC targets.

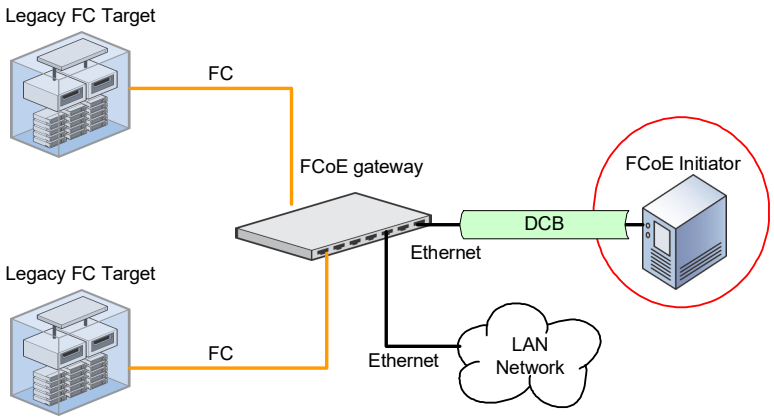


Figure 7-45. Connecting an FCoE Initiator to FC Targets

Existing FC HBAs used to connect between an FC initiator and FC targets provide full offload of the FC protocol to the initiator to maximize storage performance. To compete with this market, the X550 offloads the main data path of I/O Read and Write commands to the storage target.

7.10.1.1 FC Terminology

Useful background on FC framing and its Ethernet encapsulation can be found in Section A.4. More comprehensive material can be found in the FIBRE CHANNEL FRAMING AND SIGNALING-2 (FC-FS-2) specification. Following are some of the most common terms used extensively in the sections that describe the FCoE functionality.

- **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.

- **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_RSP Frame** — FC control frames are sent from the target to the initiator, which defines the completion of an FC read or write exchange.

7.10.2 FCoE Transmit Operation

Transmit FCoE offload is enabled by setting the *TUCMD.FCoE* bit in the transmit context descriptor. The X550 supports the following offload capabilities: FC CRC calculation and insertion, FC padding insertion and FC segmentation. These capabilities are described in the following sections.

7.10.2.1 FCoE Transmit Cross Functionality

After setting the *TUCMD.FCoE* bit, hardware digests the packet's content before it is sent to the wire. In this case, software must enable hardware offload for additional tasks as follows:

Cross Function	Requirements
Ethernet CRC insertion	Software must enable Ethernet CRC insertion by setting the *IFCS* bit in the transmit data descriptor. The Ethernet CRC covers the entire packet. Enabling FCoE offloading, hardware modifies the packet content and must also adjust the Ethernet CRC.
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.
SNAP packet	The X550 does not provide FCoE offload for FCoE frame over SNAP.
Traffic rate control	FC traffic relies on a high quality link that guarantees no packet loss. It is expected that any lost traffic protocols supported by the network are enabled by the X550 as well.
FC and PFC	
Virtualization	It is expected that the VMM abstract the FCoE functionality to the VM(s). FCoE setting and FCoE traffic is expected only by the VMM accessing the LAN via the PF.
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.
Transmit descriptors	Software must use the advanced transmit descriptor to activate either FC CRC offload or TSO functionality.

7.10.2.2 FC Padding Insertion

FC frames always consist of a whole number of four bytes. If user data is not composed of a whole number of four bytes, the FC frames contain padding bytes with a zero value. The length of the padding bytes can be any number between zero to three so together with the user data, the length of the FC frames has a whole number of four bytes. The length of the padding bytes is indicated by software in the *Fill Bytes* field in the FC header. This field is used by the receiving end node (target) to extract these bytes. Hardware does not use this field to identify the required length of the padding bytes. Instead, it checks the transmit buffer size indicated by the *PAYLEN* field in the transmit data descriptor. The length of the padding bytes added by hardware equals:

2's complement {two LS bits of (PAYLEN minus MACLEN)}.

where PAYLEN is defined in the Tx data descriptor and MACLEN is defined in the Tx context descriptor.

The X550 auto-pads the frame with the required zero bytes when FCoE offload is enabled (*TUCMD.FCoE* bit is set). In TSO, padding bytes are added only on the last frame since the MSS must be a whole number of four bytes.

7.10.2.3 SOF Placement

During a single send, the *SOF* field is taken as is from the FCoE header in the data buffer.

During TSO (both the *TUCMD.FCoE* bit in the transmit context descriptor and the DCMD.*TSE* bit in the transmit data descriptor are set), the *SOF* field in the data buffer is replaced by hardware according to the values of the *SOF* and *ORIS* bits in the transmit context descriptor. In this case the value of the *SOF* field in the data buffer is ignored (for future expansion software should set it to zero). The SOF codes that are inserted to the transmitted packets are stored in the TSOFF register. The TSOFF register contains four SOF codes named as SOF0...SOF3 that are supported by the transmit FCoE offload. By default these values are programmed to the following values: SOF0 = SOFi2; SOF1 = SOFi3; SOF2 = SOFn2; SOF3 = SOFn3. The SOF flag and *Orientation Start* (*ORIS*) bit in the *FCoEF* field in the transmit context descriptor define an index value. This index is used to extract the SOF code that is inserted to the packet as listed in the Table 7-76. The *ORIS* bit defines if the TSO starts an FC sequence or if the first frame on the FC sequence is already sent.

Table 7-76. SOF Codes in TSO

***SOF* Bit in the Context Descriptor**	***ORIS* Bit in the Context Descriptor**	**SOF Code in the First Frame**	**SOF Code in Other Frames**	**SOF Code in a Single Packet**
1 (Class 3)	1 (sequence start)	SOF1 (SOFi3)	SOF3 (SOFn3)	SOF1 (SOFi3)
1 (Class 3)	0 (not a sequence start)	SOF3 (SOFn3)	SOF3 (SOFn3)	SOF3 (SOFn3)
0 (Class 2)	1 (sequence start)	SOF0 (SOFi2)	SOF2 (SOFn2)	SOF0 (SOFi2)
0 (Class 2)	0 (not a sequence)	SOF2 (SOFn2)	SOF2 (SOFn2)	SOF2 (SOFn2)

7.10.2.4 EOF Insertion

The X550 automatically inserts the *End of Frame* field when the *TUCMD.FCoE* bit in the transmit context descriptor is set.

The *EOF* and *ORIE* fields define the *EOF* that is inserted by hardware. In a single packet send, the *EOF* field is defined completely by the *EOF* setting while in TSO mode, the *EOF* field is defined by the *EOF* and the *ORIE* bits as listed in Table 7-77.

Table 7-77. EOF Codes in TSO

***EOF* Bits in the Context Descriptor**	***ORIE* Bit in the Context Descriptor**	**Last Frame of a TSO or TSO in single frame**	**Other frames of TSO**
00 (EOFn)	0 (not a sequence end)	EOF0 (EOFn)	EOF0 (EOFn)
	1 (sequence end)	EOF1 (EOFt)	EOF0 (EOFn)
Other	X	N/A	N/A

7.10.2.5 FC CRC Insertion

FC CRC calculation is one of the most CPU intensive tasks in large transactions. The X550 offloads the FC CRC calculation when the *FCoE* bit is set in the *TUCMD* field within the transmit context descriptor. The X550 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 in the following equation. While CRC bytes are transmitted in big endian byte ordering (MS byte first on the wire):

$$X_{32}+X_{26}+X_{23}+X_{22}+X_{16}+X_{12}+X_{11}+X_{10}+X_8+X_7+X_5+X_4+X_2+X+1$$

The size of FCoE payload on which FC CRC is calculated is indicated in the context and data descriptors as follows. Figure 7-46 specifies the FCoE frame and the relevant parameters to CRC calculation.

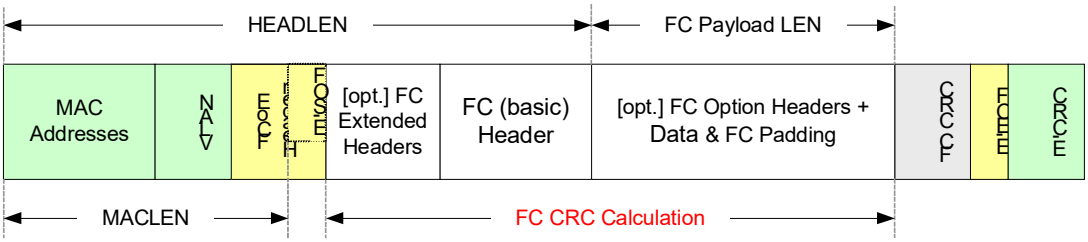


Figure 7-46. FCoE Frame and Relevant Transmit Descriptor Parameters

FC CRC Calculation Beginning:

FC CRC calculation starts after the FCoE header. It equals to byte offset of MACLEN + 4, while the *MACLEN* field in the transmit context descriptor is the byte offset of the last DWord in the FCoE header that contains the SOF flag.

FC CRC Calculation End:

FC CRC calculation ends at the end of the FC Payload LEN shown in Figure 7-46 (eight bytes before the Ethernet CRC).

7.10.2.6 Host Data Buffers Content for a Single Packet Send

Table 7-78 lists the data prepared by software when transmit FCoE offload is enabled (the *FCoE* bit in the *TUCMD* field is set in the transmit context descriptor).

Table 7-78. Transmit FCoE Packet Data Provided by Software (for *TUCMD.FCoE* = 1)

Ethernet MAC Addresses	VLAN Header	FCoE Header	FC Frame (provided by software)			
			[opt.] FC Extended Headers	FC Header	FC Option Header(s)	[Opt.] Data & FC Padding

Listed below are fields in the transmitted FCoE frame that are not included in the data buffers (in host memory) as shown in Table 7-78.

- **VLAN Header** — The VLAN header could be part of the data buffer or in the transmit descriptor depending on *VLE* bit in the *CMD* field in the transmit descriptor.

- **EOF** — The EOF is defined by the *EOF* fields and *ORIE* bit in the context descriptor (more details in Section 7.2.3.2.3).
- **FC-CRC** — The X550 calculates and inserts the FC CRC bytes.
- **FC-Padding** — The X550 calculates the padding length and inserts these bytes as required (all zeros).
- **Ethernet CRC** — Insertion should be enabled by the *IFCS* bit in transmit data descriptor.

7.10.2.7 FC 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 (single instruction). It is activated by using the advanced Tx context descriptor (DTYP equals 0010b) and setting both the *TUCMD.FCoE* in the context descriptor and setting the DCMD.*TSE* bit in the transmit data descriptor. The X550 splits the transmitted content to multiple packets as defined by the *MSS* field in the Tx context descriptor.

TSO Parameters:

- The frame header includes the Ethernet MAC Addresses, VLAN Tag, FCoE header, and the FC header. The header size is defined by the *HEADLEN* and *MACLEN* in the context descriptor as illustrated in Figure 7-46.
- The *SOF* and *EOF* fields are defined by the *SOF*, *ORIS*, *EOF* and *ORIE* fields in the context descriptor as described in Section 7.10.2.3 and Section 7.10.2.4.
- MSS – the maximum segment size in the context descriptor that define the FC data (payload) size on each packet other than the last frame which can be smaller.

7.10.2.7.1 Host Data Buffers Content for TSO Offload

Figure 7-47 shows the data in host memory when FCoE TSO is activated. The TSO header is repeated on all frames of the TSO. The header includes static and dynamic fields that are modified by hardware from packet-to-packet. The payload size is reflected in all frames.

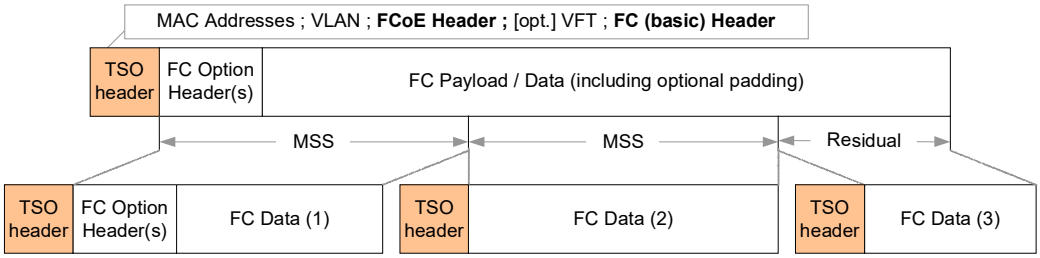


Figure 7-47. FCoE TSO Provided by the FCoE Driver

FCoE Header:

The FCoE packet header must not span more than two buffers. For best bus use it is recommended that the header be located in a single buffer (the first one).

- Ethernet MAC Addresses are the source and destination Ethernet MAC Addresses
- VLAN tag can be provided by the driver as part of the packet header or as part of the data descriptor.

- FCoE header (shown in Figure 7-47) includes the FCoE Ethernet type, FCoE Version and SOF flag. Software should leave the *SOF* fields as zero while hardware inserts it according to the *SOF* and *ORIS* bits in the Tx context descriptor.
- FC (basic) header as shown in Section A.4.2.3.

FCoE TSO — Payload:

- FC option headers as described in Section A.4.2.5.
- FC data to be segmented
- The payload may or may not include the optional FC padding bytes. Hardware adds any required padding bytes not included in the data buffers according to the *PAYLEN* field in the data descriptor.

Modified fields between consecutive frames within TSO.

SOF The SOF flags are defined by the *SOF* bit and *ORIS* bit in the context descriptor (more details in Section 7.10.2.3)

F_CTL Table 7-79 lists those fields in the F_CTL that are modified between consecutive frames of a TSO (see Section A.4.2.3 for a complete description of the *F_CTL* field). If a TSO is transmitted by a single packet all *F_CTL* fields are taken from the data buffer (as if it is the last frame in the TSO).

Table 7-79. F_CTL Codes in TSO

F_CTL Bits	Last Frame in TSO when the *ORIE* Bit in the Tx Context Descriptor is Set	Any Other Frame
Fill Bytes (1:0)	Taken from the F_CTL(1:0) in the data buffer. It defines the length of the FC padding required to make the FC data a complete multiply of four bytes.	00b if not last frame on TSO or taken from the F_CTL(1:0) in the data buffer if last frame in TSO (even if *ORIE* bit is not set).
Continue Sequence Condition (7:6)	Taken from the F_CTL(7:6) in the data buffer. The continue sequence condition is meaningful only if F_CTL(19) is set and F_CTL(16) is cleared.	00b
Sequence Initiative (16)	Taken from the F_CTL(16) in the data buffer. The sequence initiative is meaningful only if F_CTL(19) is also set.	0b
End Sequence (19)	Taken from the F_CTL(19) in the data buffer. The end sequence should be set to 1b by software only if the frame is the last one of a sequence.	0b

DF_CTL Table 7-80 lists those fields in the *DF_CTL* that can be modified between consecutive frames of a TSO. Note that the *ESP Header Presence* bit is not listed in this table. When *ESP Header* is present, software must not use a TSO that spans across multiple packets. If a TSO is transmitted by a single packet all *DF_CTL* fields are taken from the data buffer (as if it is the first frame in the TSO).

Table 7-80. DF_CTL Codes in TSO

DF_CTL Fields	1st Frame in TSO when *ORIS* Bit in the Tx Context Descriptor is Set	Any Other Frame
Device Header Indication (1:0)	Taken from the data buffer.	00b
Association Header Indication (4)	Taken from the data buffer.	0b
Network Header Indication (5)	Taken from the data buffer.	0b

SEQ_CNT SEQ_CNT in the first frame is taken from the *SEQ_CNT* field in the FC header in the data buffers. On any other frame, the value of *SEQ_CNT* is increased by one from its value in the previous frame. The *SEQ_CNT* wrap-to-zero after reaching a value of 65,535.

PARAM The *PARAM* field in the first frame is taken from the *PARAM* field in the FC header in the data buffers. If the FCoEF.*PARINC* bit is set in the transmit context descriptor, the value of the *PARAM* becomes dynamic. In that case, the *PARAM* is increased by hardware by the MSS value on each frame. Software should set the FCoEF.*PARINC* bit when the *PARAM* field indicates the data offset (*Relative Offset Present* bit in the *F_CTL* field is set).

FC_CRC Calculated and inserted on each frame as described in Section 7.10.2.5.

FC_Padding Calculated the number of required padding bytes and inserted them on the last frame as described in Section 7.10.2.2.

EOF The EOF flags are defined by the *EOF* field and the *ORIE* bit in the context descriptor (more details in Section 7.10.2.4).

7.10.3 FCoE Receive Operation

The X550 can offload the following tasks from the CPU while processing FCoE receive traffic: FC CRC check, receive coalescing and Direct Data placement (DDP). These offload options are described in the sections that follow.

DDP functionality is not provided for control packets or data packets that do not meet DDP criteria (described later in the sections that follow). In those cases, hardware posts the packets to the legacy Rx queues as is (header and trailer are not stripped including SOF, EOF, FC padding and FC CRC bytes). When DDP functionality is enabled, only the FC payload is posted to the user buffers. If the packet's header should be indicated to the legacy Rx queues, all bytes starting at the destination Ethernet MAC Address until the FC header and optionally FC header(s) inclusive are posted to the legacy buffer.

7.10.3.1 FCoE Receive Cross Functionality

FCoE receive offload capabilities coexist with other functions in the X550 are listed as follows:

Table 7-81. FCoE Receive Cross Functionality

Cross Function	Requirements
Ethernet CRC check	There is no enforcement on save bad frames policy. In the case of save bad frames, packets with bad Ethernet CRC are posted to the legacy receive queue even if DDP is enabled. FC payload of bad packets are never posted directly to the user buffers.
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.
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.
SNAP packet	The X550 does not provide FCoE offload for FCoE frame over SNAP.
FC and PFC	FC traffic relies on a high-quality link that guarantees no packet loss. It is expected that any lost traffic protocols supported by the network is enabled by the X550 as well.
Virtualization	It is expected that VM(s) generate FC write requests to the VMM. FCoE setting and FCoE traffic is expected only by the VMM accessing the physical function.

Table 7-81. FCoE Receive Cross Functionality [continued]

Cross Function	Requirements
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.
Jumbo frames	Maximum expected clear text FC frame size is 2140 bytes (FC header + FC payload + FC CRC). Adding optional FC crypto, plus FCoE encapsulation, packet might exceed the 2200 bytes. To enable FCoE traffic, jumbo packet reception should be enabled.
Receive descriptors in the legacy Rx queues	When FC CRC offload or DDP functionality are enabled, software must use the advanced descriptors in the associated legacy Rx queues (SRRCTL.*DESCTYPE* = 001b). The legacy Rx buffers must be larger than the maximum expected packet size so any Rx packets span on a single buffer.

7.10.3.2 FC Receive CRC Offload

FC CRC calculation is one of the most CPU intensive tasks in TSO transactions. The X550 offloads the receive FC CRC integrity check while trashing the CRC bytes and FC padding bytes.

The X550 recognizes FCoE frames in the receive data path by their FCoE Ethernet type and the FCoE version in the FCoE header. The Ethernet type that hardware associates with FCoE is defined in the ETQF register by setting the *FCoE* bit with a specific Ethernet type value. The supported FCoE versions by the Rx offload logic are defined by FCRXCTRL.*FCOEVER.* FCoE packets that do not match the previously described Ethernet type and FCoE versions are ignored by the Rx FCoE logic.

The X550 reconstructs the FC CRC while processing the incoming bytes and compares it against the received FC CRC. The frame is considered a good FC packet if the previous comparison matches and it is considered as a bad FC packet otherwise.

The FC CRC integrity check is meaningful only if all the following conditions are met:

- The received frame contains a correct Ethernet CRC

The length of the FC padding bytes that hardware trashes are defined in the *Fill Bytes* field in the FC frame control (F_CTL). The *Fill Bytes* field can have any value between zero to three that makes the FC frame a whole number of DWords. It is expected that the *Fill Bytes* field would be zero except for last data frames within a sequence.

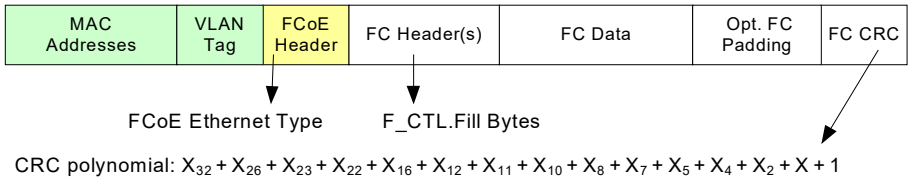


Figure 7-48. Relevant FCoE and FC Fields for CRC Receive Offload

7.10.3.3 Large FC Receive

Large FC receive includes two types of offloads. The X550 can save a data copy by posting the received FC payload directly to the kernel storage cache or the user application space (in the remainder of the document there is no difference between the two cases and it is named as user buffers). When the packet's payload are posted directly to the user buffers their headers can be posted to the legacy receive queues. The X550 saves CPU cycles by reducing the data copy and also minimize CPU processing by posting only the packet's headers that are required for software.

Figure 7-49 shows the mapping of received FCoE frames to the legacy Rx queue and the user buffers. Figure 7-50 shows a top level overview of the large FC receive flow. The remaining sections detail the large FC receive functionality as follows:

- Enabling large FC receive — Section 7.10.3.3.1
- FC read exchange flow — Section 7.10.3.3.2
- FC write exchange flow — Section 7.10.3.3.3
- FCoE receive filtering (Frame types and rules) — Section 7.10.3.3.5, Section 7.10.3.3.10 and Section 7.10.3.3.12
- User descriptors — Section 7.10.3.3.9
- Header posting to the legacy receive queues and FC exceptions — Section 7.10.3.3.13 and Section 7.10.3.3.14
- Interrupts — Section 7.10.3.3.15

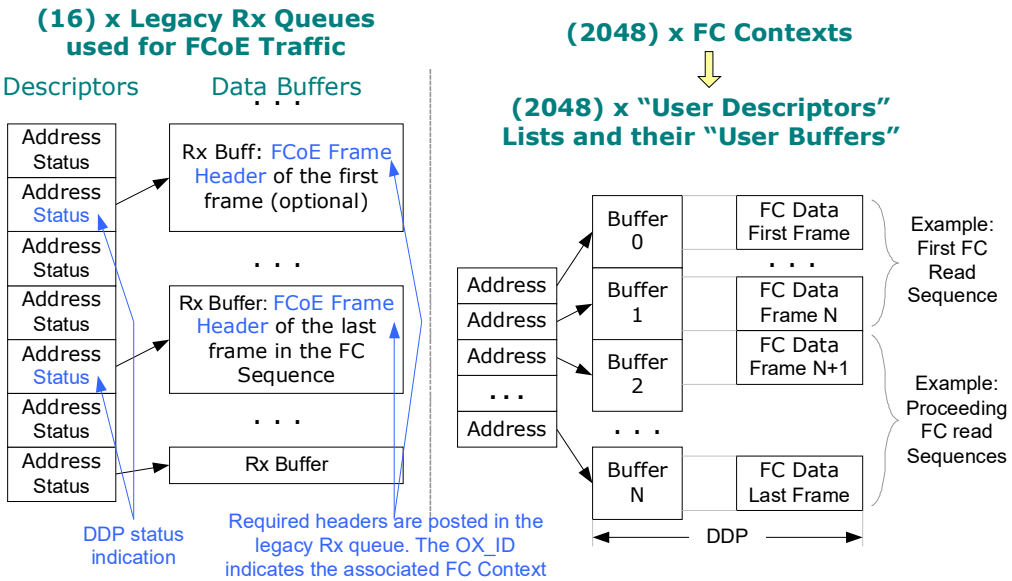


Figure 7-49. Large FC Reception to User Buffers and Legacy Rx Queue

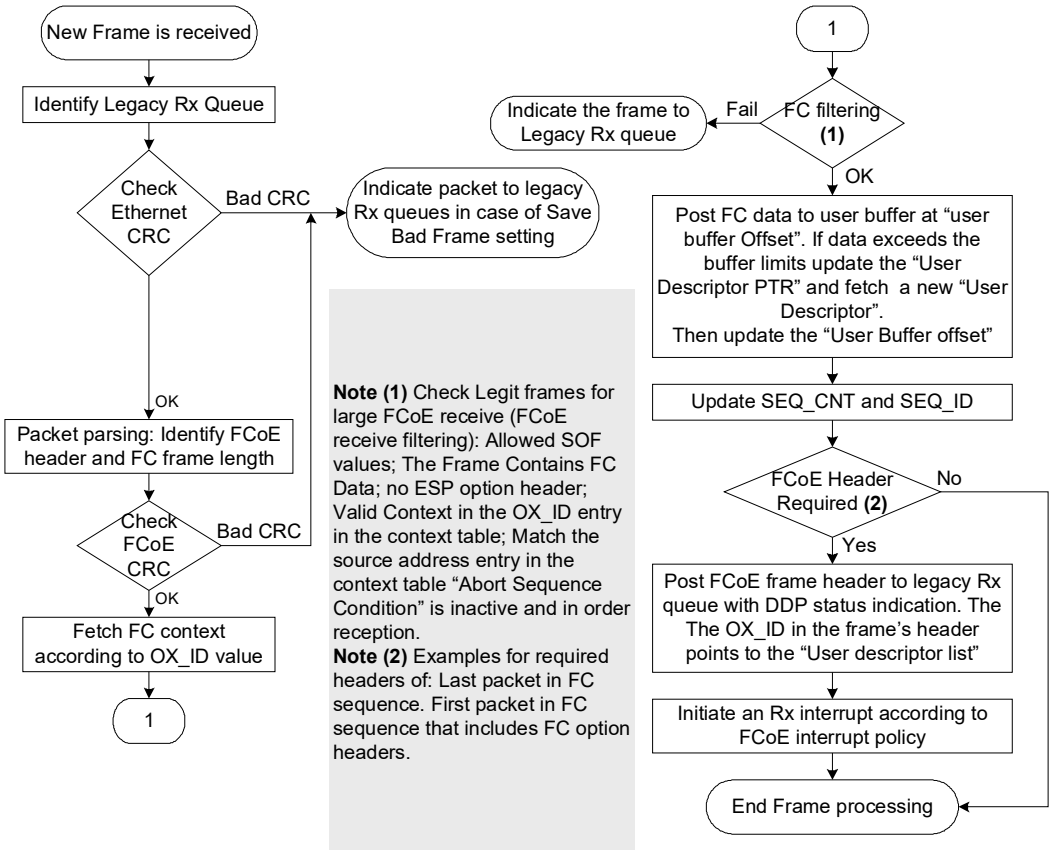


Figure 7-50. FCoE Large Receive Flow Diagram

7.10.3.3.1 Enabling Large FC Receive

Large FC receive offload is enabled per each outstanding read or write exchange by programming the FCoE context table with the flow parameters. Setting the FC context for read or write exchange is done at run time. It is expected that a read context is programmed before the read request is initiated to the remote target and write context is programmed before the target sends the ready indication to the initiator. Unless the FC context is invalidated, software must not modify it in the middle of a transaction (see Section 7.10.3.3.5 for details on context invalidation). For more details on FCoE initialization flow see Section 4.6.9.

7.10.3.3.2 FC Read Exchange Flow

Figure 7-51 shows an example of an FC (class 3) read request. This flow is detailed in this section.

1. The software checks if the read request can use large FC receive offload depending on FC context resources and some criteria as listed in Section 7.10.3.3.5. Section 7.10.3.3.12 describes a proposed software flow to manage the FC contexts.
2. If the previous conditions are not met, software can still initiate the FC read request according the flow described in Figure 7-51 while the received frames are posted to the legacy receive queues.

- If the previous conditions are met, software locks the relevant user buffers (the target buffers for the FC read request) and program the FC context table. It then initiates the FC read request according the flow shown in Figure 7-51. The payload of the received frames is posted directly to the user buffers. Some of the packet's headers (only the required ones) are posted to the legacy receive queues. The FC header in the packet's header contains the *OX_ID* field. This field indicates to software its context and its user buffer list. During nominal operation, all packets' headers except packets with FC optional headers are trashed by the hardware minimizing software overhead.

3. The target sends the FCP_RSP frame type indicating the completion of the read exchange. As a response, the hardware invalidates the FC read context (if it was used) and indicates the number of bytes posted directly to the user buffers in the receive descriptor (see Section 7.10.3.3.13). Software indicates the read completion to the application.

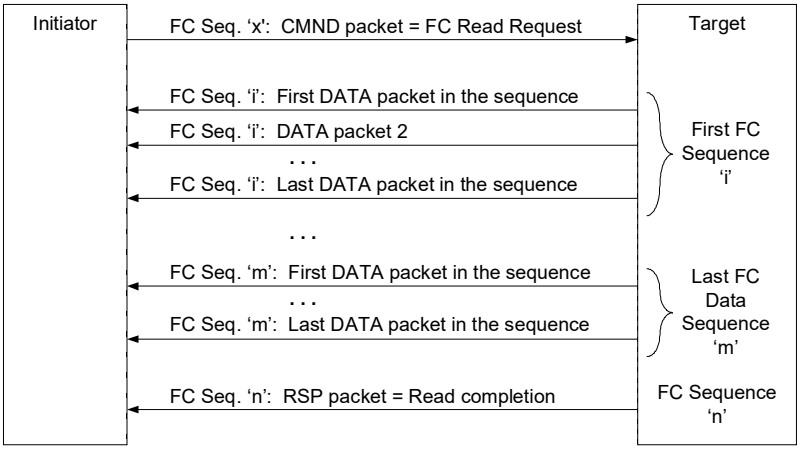


Figure 7-51. Example for FC Class 3 Read Exchange Flow

7.10.3.3.3 FC Write Exchange Flow

Figure 7-52 shows an example of an FC (class 3) write request (on which the Seq_CNT starts from zero on each new sequence). This flow is detailed in the sections that follow.

1. The host (originator) sends an FC write request to the target (responder).
2. Software in the target checks if the write request can use large FC receive offload depending on FC context resources and some criteria as listed in Section 7.10.3.3.5.
3. If the previous conditions are met, software can use DDP for this FC write exchange.
4. The target software locks the relevant user buffers (the target buffers for the FC write request) and program the FC context table. It then initiates the FC ready indication to the host.
5. As a response, the host sends the data frames to be written to the target. The frames are received in the target. If DDP is used, the FC payload is posted directly to the user buffers while "most" (see additional details below) packet's headers are trashed minimizing software overhead.
6. The host marks the last data frame it was requested to send by setting the *Sequence Initiative* bit in the *F_CTL* field.

7. The target identifies the last data frame and invalidates the DDP context. As indicated above, during nominal operation, "most" packet's headers are trashed. Only headers that have meaningful content are posted to host memory as: Headers of packets with FC optional headers and the header of the last packet in a sequence with active sequence initiative bit are posted to the legacy receive queues. The hardware indicates the number of bytes posted directly to the user buffers in the receive descriptor (see Section 7.10.3.3.13). Note that the FC header contains the *RX_ID* field that can be used by software to identifies its associated DDP context and user buffer list.
8. The target may repeat Step 4, which is followed by Step 5 until the entire requested data is transferred.
9. The target sends the FCP_RSP frame indicating to the initiator the completion of the write exchange.

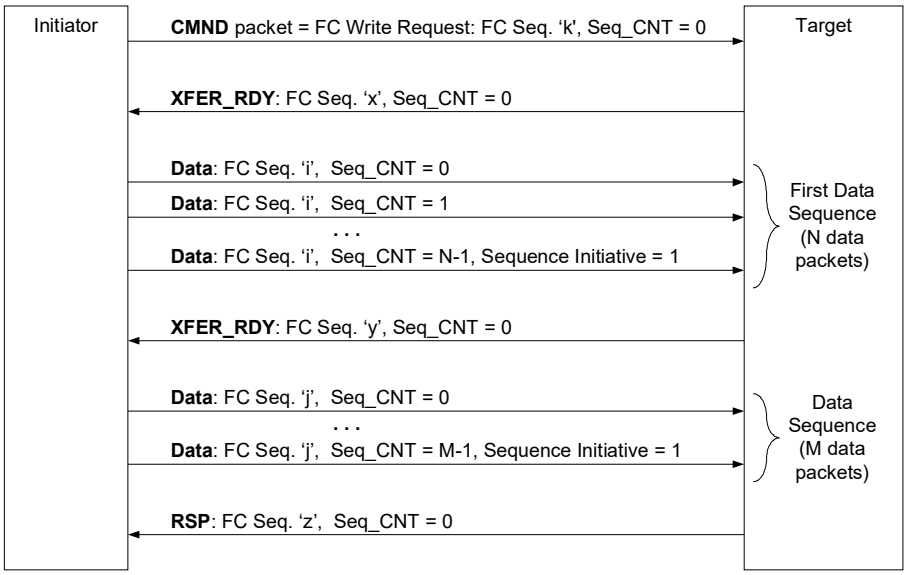


Figure 7-52. Example for FC Class 3 Write Exchange Flow

7.10.3.3.4 EOF and SOF Flags identification

As part of the DDP functionality, hardware identifies the SOF and EOF flags in the received packets. The flags identification is based on a setting of the RSOFF and REOFF registers. These registers are identical to the TSOFF and TEOFF registers and should be programmed by software to the same values.

7.10.3.3.5 FCoE Receive Filtering

Received FCoE frames are associated to one of the legacy receive queues according to the scheme described in Section 7.1.3. When the legacy receive queue is enabled, large FC receive functionality is enabled as well if a matched FC receive context is defined. The data is posted to the user buffers that are pointed to by the FC receive context. Some of the headers of these frames that are required for the data processing are posted to the legacy receive queue (see Section 7.10.3.3.13).

FCoE frames that carry FC class 3 or class 2 data can be posted to large receive buffers if they meet the following conditions:

- The FC context table contains valid context that matches the exchange ID in the received frame. Hardware checks the RX_ID for write data packets sent by the initiator. These packets are identified by the *Exchange Context* bit in the F_CTL header equals zero (originator of exchange). Hardware checks the OX_ID for read response data packets sent by the target. These packets are identified by the *Exchange Context* bit in the F_CTL header equals one (responder of exchange).
- The FC frame source address match the source MAC Address field in the in the same valid exchange ID context.
- Frames are identified as FCoE frame type according to the Ethernet type in the FCoE header. The Ethernet type that hardware associates with FCoE is defined in the ETQF registers by setting the *FCoE* bit with a specific Ethernet type value.
- The FC frame carry class 2 or class 3 content as defined by the SOF flag. The SOF in the FCoE header equals SOFi2 or SOFn2 or SOFi3 or SOFn3.
- The FCoE version in the received frame is equal or lower than FCRXCTRL.*FCOEVER*.
- The frame contains data content (with data payload) as defined in the *Routing Control* field (R_CTL) in the FC header:
  - R_CTL.*Information* (least significant four bits) equals 0x1 (solicited data).
  - R_CTL.*Routing* (most significant four bits) equals 0x0 (device data).
  - Frames that do not contain device data are not posted to the user buffers. Still these frames are compared against the expected SEQ_ID and SEQ_CNT in the FC context and update these parameters as described in Section 7.10.3.3.5.
- The FC frame does not include ESP header (bit 6 in the *DF_CTL* field within the FC header is cleared). Frames that include ESP option headers are posted to the legacy receive queue. For good use of hardware resources, software should not program the large FC receive context table with flows that carry an ESP header.
- The FC frame does not include FC extended headers. For good use of hardware resources, software should not program the large FC receive context table with flows that carry extended headers.
- The first packet received to a new context is identified as the first FC frame in the exchange. This packet is expected to have the SOFi2 or SOFi3 codes. The SEQ_ID on the first packet may have any value.
- The frame is received in order as defined in Section 7.10.3.3.7 and does not carry any exception errors as defined in Section 7.10.3.3.14.
- The first frame on each FC sequence is identified by the SOFi2 or SOFi3 codes in the *SOF* field in the FCoE header. It is expected that the SEQ_ID is changed for any new sequence.
- The last frame on each FC sequence is identified by an active *End Sequence* flag in the *F_CTL* field in the FC header. It is expected to receive the EOFt code in the *EOF* field; however, hardware does not check this rule.

Other frames (that do not meet the previous conditions) are posted to the legacy receive queues according to the generic Rx filtering rules.

7.10.3.3.6 DDP Context

Hardware can provide DDP offload for up to 2048 concurrent outstanding FC read or write exchanges. Each exchange has an associated FC context in hardware. Contexts are identified by the exchange ID (OX_ID for FC read and RX_ID for FC write). The exchange ID is a 16-bit field so that a system could theoretically generate up to 64 K concurrent outstanding FC read requests and 64 K concurrent outstanding FC write requests. Hardware contains 2048 contexts for the 2048 concurrent outstanding exchanges. Using exchange ID values between 0 to 2047, software can benefit from the DDP offload. Each entry in the DDP table can be used to offload either a read exchange (as the initiator) or a write exchange (as the target).

Mapping an exchange to a DDP context is done by the exchange ID. The 11 LS bits of the exchange ID is the DDP context Index. In addition, read DDP context is defined by the 11 LS bits of the OX_ID and write DDP context is defined by the 11 LS bits of the RX_ID. Therefore, if a specific OX_ID is offloaded by read DDP, the same number of RX_ID can't be offloaded by write DDP at the same time.

The FC context is a set of parameters used to identify a frame and its user buffers in host memory. The context parameters are split into two categories (according to the internal hardware implementation): DMA context (FCPTRL, FCPTRH, FCBUFF and FCDMARW registers) and filter context (FCFLT, FCPARAM, SMAC, FCD_ID, and FCFLTRW registers) as listed in Table 7-82 and shown in Figure 7-53.

Software should program both the DMA context and filter context making the context usable. During reception, hardware updates some of the parameters if the packet matches all criteria detailed in Section 7.10.3.3.5. Initialization values and the updated ones are listed in this section.

Table 7-82. Large FC Context Table

Exchange ID	DMA Context (FCPTRL, FCPTRH, FCBUFF, FCDMARW)		Filter Context (FCFL, FCFLTRW, FCBUFF)	
	DMA Flags	User Descriptor	Filter Flags	In Order Reception
0	Valid, First, Last, Count	Size, Offset, Pointer	Valid, First	Seq_ID, Seq_CNT, PARAM,SMAC[1], D_ID[2]
1	Valid, First, Last, Count	Size, Offset, Pointer	Valid, First	Seq_ID, Seq_CNT, PARAM, SMAC[1], D_ID[2]
2	Valid, First, Last, Count	Size, Offset, Pointer	Valid, First	Seq_ID, Seq_CNT, PARAM, SMAC[1], D_ID[2]
...	...		...	
2047	Valid, First, Last, Count	Size, Offset, Pointer	Valid, First	Seq_ID, Seq_CNT, PARAM, SMAC[1], D_ID[2]

1. If *FCRXCTRL.SMAC_EN* is set
2. If *FCRXCTRL.DID_EN* is set

Note: The *WRCONTX* field differentiating read and write exchanges is not part of the filter. Thus software should not request a DDP offload of a read flow if a write flow with the same Exchange ID may be received and vice versa.

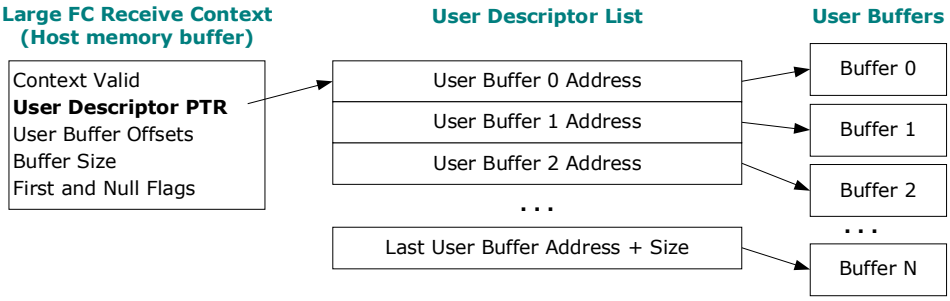


Figure 7-53. Large FC Receive Context Related to the User Buffers

Table 7-83. FCoE DMA and Filter Contexts

Field	Width (Bits)	Description	Initialization Value	Updates by Hardware
DMA Context Valid[1]	1	DMA context is valid.	1	Cleared when context is invalidated. See Section 7.10.3.3.10 and Section 7.10.3.3.11
Filter Context Valid	1	Filter Context is valid.	1	
Filter First	1	The first received frame that matches an active context in the filter unit is marked by the filter. This marking is used by the DMA unit as an indication that reception to this context has been started. The DMA context does not accept packets from the filter unit unless it received successfully the packet that was marked as the first one (see the section on exception handling in Section 7.10.3.3.14).	0	Hardware sets this bit when the filter unit recognizes the first packet that matches a valid context.
DMA First	1		0	Hardware sets this bit when the DMA unit received packet that matches a valid context and marked as first by the filter unit.
DMA Last	1		0	Hardware sets this bit when the DMA unit received packet that matches a valid context and marked as last by the filter unit.
Buffer Count	8	Defines the number of remaining user buffers in the list. A value of 0x00 means 1024 buffers	Number of the allocated user buffers.	During reception, hardware decrements the buffer count as each of them completes.
Buffer Size	2	Defines the user buffer size used in this context. 00b = 4 KB 01b = 8 KB 10b = 16 KB 11b = 64 KB. All buffers except the first one and the last one are full size. The address of all buffers is aligned to the buffer size in the context. The first buffer can start at a non-zero offset. The size of the last buffer can be smaller than the buffer size as defined by the last buffer size parameter.	Buffer Size	Static Parameter
User Buffer Offset	16	Byte offset within the current buffer to which the next packet should be posted.	Beginning of the first buffer.	Next received packet offset.

Table 7-83. FCoE DMA and Filter Contexts [continued]

Field	Width (Bits)	Description	Initialization Value	Updates by Hardware
Last User Buffer Size	16	Size of the last user buffer in byte units.	Last user buffer size.	Static Parameter
User Descriptor PTR	8	The user buffers are indicated by a list of pointers named as user descriptors (see Section 7.10.3.3.9 for a description of the user descriptors). The user descriptor PTR in the FC context is a pointer to the user descriptor list.	Beginning of the user descriptor list.	Hardware increments the user descriptor PTR by eight (the size of the user descriptor) when it completes a buffer and requires the next one.
SEQ_ID	8	The sequence ID identifies the received sequence number. An FC read or write exchange can be composed of multiple sequences depending on the target implementation. The SEQ_ID has a different value for each sequence and does not necessarily increment sequentially. Hardware uses the SEQ_CNT and SEC_ID for checking in-order reception as described in Section 7.10.3.3.7.	0	Hardware updates the SEQ_ID in the context table according to the value of the SEQ_ID in the incoming frame.
SEQ__CNT	16	SEQ_CNT is an index of the expected FC frames within a sequence or within the entire exchange. Hardware uses the SEQ_CNT and SEC_ID for checking in-order reception as described in Section 7.10.3.3.7.	Read context: 0 Write context: SEQ_CNT + 1 of the last packet of the same exchange received from the initiator.	For each in-order reception, hardware sets SEQ_CNT in the context to the value of the received SEQ_CNT + 1.
PARAM	32	The *PARAM* field in the FC header may indicate the data offset within the FC IO exchange. It is indicated as an offset by the *Relative Offset Present* bit in the *F_CTL* field in the FC header. In this case, the PARAM field indicates the expected value of the next received packet.	Read context: 0 Write context: the expected received value	Hardware increments the PARAM by the size of the FC payload if it is used as an offset. The FC payload size equals the packet size minus the length of its header and trailer. While the header for this purpose includes all bytes starting at the Ethernet destination address up to and including the basic FC header, the trailer includes the FC CRC, FC padding, EOF including the three reserved bytes, and the Ethernet CRC.
SMAC	48	Source MAC Address of the FCoE sender. Used to match the source MAC Address of the incoming FCoE packets. The comparison of the *SMAC* field is enabled by the FCRXCTRL.*SMAC_EN* bit	Source MAC	Static Parameter
D_ID	24	The *D_ID* field is configured with the FC Destination Identification address of the FCoE receiver and is used to match D_ID value of the incoming FCoE packets. The comparison of the *D_ID* field is enabled by the FCRXCTRL.*DID_EN* bit	D_ID	Static Parameter
WRCONTX	1	Write DDP Context. This bit should be set to 1 for write exchange context aimed for target (responder) usage. This bit should be set to 0 for read exchange context aimed for initiator (originator) usage.	WRCONTX	Static Parameter

1. During programming time, software should enable first the DMA context. When software disables a context it should invalidate first the filter context. See more details on context invalidation in Section 7.10.3.3.10.

7.10.3.3.7 In Order Reception Checking

Hardware checks in-order reception by *SEQ_ID*, *SEQ_CNT* and *PARAM* fields. These parameters should meet the expected values (as follows) to pass in-order reception's criteria.

- **PARAM** — When the *PARAM* field is used as an offset (as indicated by the *Relative Offset Present* bit in the *F_CTL* field in the FC header), the *PARAM* field in the received packet should be the same as the *PARAM* field in the FC context. Software should initialize this parameter to the expected received value (equals to zero in read exchanges).
- **SEQ_ID, SEQ_CNT** — SEQ_ID identifies the FC sequence and SEQ_CNT is the FC frame index within the entire exchange or within the sequence (according to specific vendor preference). SEQ_CNT in the received packet could be either the same as the SEQ_CNT in the FC context or it could start from zero for new SEQ_ID, which is different than the SEQ_ID in the context. Software should initialize SEQ_CNT to the expected received value (equals zero in read exchanges). SEQ_ID on the first packet is always assumed to be a new value even if by chance it equals to the initial value in the context.

7.10.3.3.8 Accessing the Large FC Receive Context

The X550 supports a large number of FC contexts while each context contains about 16 bytes. To save consumed memory space, the FC context is accessed by indirect mapping. This section describes how the DMA and filter contexts are accessed. The DMA context is consist of the FCPTRL, FCPTRH and FCBUFF registers while read and write accesses are controlled by the FCDMARW register. The filter context is consist of the FCFLT register while read and write accesses are controlled by the FCFLTRW register.

- **Indirect DMA Context Programming** — Software should program the FCPTRL, FCPTRH and FCBUFF registers by the required setting. It then programs the FCDMARW register with the following content:
  - FCOESEL should be set by the required context index (OX_ID or RX_ID values)
  - The *WE* bit is set to 1b for write access while the *RE* bit is set to 0b. *LASTSIZE* should be set to the relevant value for the context
- **Indirect DMA Context Read** — Software should program the FCDMARW register as follows and then read the context on the FCPTRL, FCPTRH, FCBUFF and FCDMARW registers
  - FCOESEL should be set by the required context index (OX_ID or RX_ID values).
  - *RE* bit should be set to 1b for read access while *WE*, and *LASTSIZE* fields are set to 0b. *LASTSIZE* should be set to 0b. It is ignored by hardware when the *RE* bit is set to 1b. When reading FCDMARW the *LASTSIZE* field reflects the context content.

  The FC context may handle up to 1024 user buffers in a DDP request.
- **Direct DMA Context Access** — Software can access the FCoE DMA context directly for reading or programming through the FCDDC registers, each 4 consecutive DWs represent a separate FCoE Context:
  - DW[4*n]: *FCPTRL[n]*
  - DW[4*n+1]: *FCPTRH[n]*
  - DW[4*n+2]: *FCBUFF[n]*
  - DW[4*n+3]: *FCDMARW[n]*

- **Indirect Filter Context Programming** — Software should program the FCFLT register by the required setting. It then programs the FCFLTRW register with the following content:
  - FCOESEL should be set by the required context index (OX_ID or RX_ID values).
  - *WE* bit is set to 1b for a write access while *RE* bit is set to 0b.
- **Direct Filter Context Access** — Software can access the FCoE Filter Context directly for reading or programming through the FCDFC, FCDFCD registers. FCDFCD has a DW per context while FCDFC is structured such that each 4 consecutive DWs represent a separate FCoE context:
  - *FCDFC:*
    - DW[4*n]: *FCFLT[n]*
    - DW[4*n+1]: *FCPARAM[n]*
    - DW[4*n+2]: *FCSMAC[n]*
    - DW[4*n+3]: *FCFLTRW[n]*
  - *FCDFCD*:
    - DW[n]: *FC D_ID*
- **Indirect Filter Context Read** — Software should program the FCFLTRW register as follows and then read the context on the FCFLT register:
  - FCOESEL should be set by the required context index (OX_ID or RX_ID values).
  - *RE* bit should be set to 1b for a read access while *WE* bit is set to 0b.

  Software should access the FCoE context either through the direct access type or indirect access type. Software mixing the access type accesses results in un-defined outcome.

7.10.3.3.9 User Descriptor Structure and User Descriptor List

The buffers in host memory could be either user application memory or storage cache named as user buffers. In both cases the buffers must be locked (against software) and converted to physical memory up front.

The user descriptor list is a contiguous list of pointers to the user buffers. The buffers are aligned to their size as defined in the FC context. The first buffer can start a non-zero offset as the software defines it in the FC context. All other buffers start at a zero offset. The last buffer can be smaller than the full size as defined in the FC context.

Table 7-84. FC User Descriptor

63	0
User buffer address defined in byte units. N LS bits must be set to zero while N equals 12 for a 4 KB buffer size, 13 for 8 KB buffer size, 14 for 16 KB buffer size and 16 for 64 KB buffer size.	

7.10.3.3.10 Invalidating FC Read Context

During nominal activity, hardware invalidates autonomously the FC contexts. The target indicates a completion of an FC read by sending the FCP_RSP frame. Hardware identifies the FCP_RSP frame and invalidates the FC context that matches the OX_ID in the incoming frame. The FCP_RSP frame is posted to the legacy Rx queues with appropriate status indication. Hardware identifies the FCP_RSP frame by the following criteria:

- The frame is identified as FCoE frame by its Ethernet type
- R_CTL.Information (least significant four bits) equals 0x7 (command status)
- R_CTL.Routing (most significant four bits) equals 0x0 (device data)

Context that is invalidated autonomously by hardware is indicated by setting the *FCSTAT* field in the receive descriptor to 10b. When software gets this indication it can unlock the user buffers instantly and re-use the context for a new FC exchange.

In some erroneous cases software might invalidate a context before a read exchange completes (such as a time out event). In such cases, software should clear the *Filter Context Valid* bit and then the *DMA Context Valid* bit. Hardware invalidates the context at a packet's boundaries. Therefore, after software clears the *DMA Context Valid* bit, software should either poll it until it is granted (cleared) by hardware or optionally software could wait ~100 µs (guaranteed time for any associated DMA cycles to complete). In addition, software should also ensure that the receive packet buffer does not contain any residual packets of the same flow. See Section 4.6.7.1.1 for the required software flow. Only then the software can unlock the user buffers and re-use the context for a new FC exchange.

Note: While RXCTRL.*RXEN* is in the disabled state, frames from the DDP filter logic may be dropped silently by the device. Therefore, prior to clearing RXCTRL.*RXEN* bit, any outstanding or active DDP context should be invalidated and those associated DDP I/O read requests retried.

7.10.3.3.11 Invalidating FC Write Context

During nominal activity, hardware invalidates autonomously the FC contexts. The initiator indicates a completion of a granted portion of an FC write by sending a data frame with active sequence initiative flag. After receiving this type of frame, hardware invalidates the matched FC context. The header of this frame is posted to the legacy Rx queues with appropriate status indication. Hardware identifies this frame by the following criteria:

- The frame is identified as FCoE frame by its Ethernet type
- R_CTL — Information (least significant four bits) equals 0x1 (solicited data).
- R_CTL — Routing (most significant four bits) equals 0x0 (device data).
- F_CTL — Sequence initiative equals 1b indicating transfer initiative to the target.
- F_CTL — End sequence equals 1b, indicating last frame in a sequence.

Context that is invalidated autonomously by hardware is indicated by setting the *FCSTAT* field in the receive descriptor to 10b. When software gets this indication, it can unlock the user buffers instantly and re-use the context for a new FC exchange. If the FC write is not complete, software can re-use the same context for the completion of the exchange. It can also define a new user buffer list and indicate it to hardware by programming the DMA context. It then can enable the filter context by setting the *Re-Validation* bit the *WE* bit and the *FCOESEL* field in the FCFLTRW register.

Software can also invalidate a context in case of a time out event or other reasons. Software invalidation flow is described in Section 7.10.3.3.10.

7.10.3.3.12 OX_ID and RX_ID Pool Management

As previously indicated, hardware enables Large FC receive offload for up to 2048 concurrent outstanding read or write requests. In some cases more than 2048 concurrent outstanding requests are generated by the FCoE stack. Therefore, software might need to manage two separate queues for the requests: one queue for those FC requested supported by the large FC receive offload and another one for those requests that do not gain the large FC receive offload. Software should claim an entry in the context table, and its associated OX_ID or RX_ID for the duration of the read or write requests, respectively. Once a request completes and its context is invalidated, software can re-use its context entry for a new request.

Table 7-85 defines an example for an OX_ID list that can be used for new FC read requests managed by software at initialization time and during run time. Similarly, this table could be helpful for write requests and their RX_ID or shared pool for both read and write requests.

Table 7-85. Software OX_ID Table

Init State of the OX_ID Table	Run-Time Events	Updated State of the OX_ID Table	Run-Time Events	Updated State of the OX_ID Table	Run-Time Events	Updated State of the OX_ID Table
0	Software is using 50 OX_ID values supported by large FC receive.	50	The following FC read requests are completed (and released by software) in the following order: 44, 21, 9, 0.	50	Software is using additional 50 OX_ID values supported by large FC receive. The following FC read requests are completed (and released by software) in the following order: 75, 10, 38. Ordering between software requests and releases does not matter in this example.	100
1		51		51		101
. . .		. . .		. . .		. . .
. . .		. . .		. . .		2046
. . .		. . .		. . .		2047
. . .		2046		2046		44
. . .		2047		2047		21
. . .				44		9
. . .				21		0
. . .				9		75
2046				0		10
2047						38

Note: Software is aware of which read requests can be offloaded by the large FC receive and use OX_IDs in the hardware range (0 to 2047) only for those ones.

7.10.3.3.13 Packets and Headers Indication in the Legacy Receive Queue

The following packets or packet's headers are posted to the legacy receive queues:

- All FCoE frames that are not offloaded by the DDP logic
- Any packet with exception errors as described in Section 7.10.3.3.14
- Headers of packets posted to the user buffers by the DDP logic that contain meaningful data (as detailed in Section 7.10.3.3.2 and Section 7.10.3.3.3)

There are a few new fields in the receive descriptor dedicated to FCoE described in Section 7.1.5.2.2:

- **Packet Type** — FCoE packets are identified by their Ethernet type that is programmed in the ETQF registers.

- **FCoE_PARAM** — Reflects the value of the *PARAM* field in the DDP context.
- **FCSTAT** — FCoE DDP context indication.
- **FCERR** — FCoE Error indication. DDP offload is provided only when no errors are found.
- **FCEOFs and FCEOFe** — Status indication on the *EOF* and *SOF* flags in the Rx packet.

7.10.3.3.14 Exception Handling

Table 7-86 lists the exception errors related to FC receive functionality. Packets with any of the following exception errors are posted to the legacy receive queues with no DDP unless specified differently. In these cases, the exception error is indicated in the *Extended Error* field in the receive descriptor. The exceptions are listed in priority order in the table with highest priority first. Other than the EOF exception, any high priority exception hides all other ones with a lower priority.

Table 7-86. Exception Error Table

Event Description	Actions and Indications
Unsupported FCoE version (Rx Version > FCRXCTRL.*FCOEVER*)	The packet is identified as an FCoE packet type. DDP context parameters are left intact. Speculative CRC check is done. The packet is posted to legacy Rx queue (unless CRC is incorrect and FCRXCTRL.*SAVBAD* is not set). If the packet matches the FCoE redirection table, the packet is posted to Rx queue index defined by its OX_ID. RDESC.STATUS.FCSTAT = 00b. RDESC.ERRORS.FCERR = 100b.
Rx packet with ESP option header.	If it matches the DDP context, auto invalidate the filter context while keeping the parameters intact. Note that this exception is not expected since software should not enable a context to an exchange that uses ESP encapsulation. RDESC.STATUS.FCSTAT = 00b / 01b / 10b. RDESC.ERRORS.FCERR = 000b.
Received EOFa or EOFni or any unrecognized EOF or SOF flags.	If it matches the DDP context, auto invalidate the filter context while keeping the parameters intact. RDESC.STATUS.FCSTAT = 00b / 01b / 10b. RDESC.ERRORS.FCERR = 010b (even if no DDP match). RDESC.ERRORS.FCEOFe = 1b. RDESC.STATUS.FCEOFs = 1b.
Received non-zero abort sequence condition in FC read exchange.	If it matches the DDP context, auto invalidate the filter context while keeping the parameters intact. RDESC.STATUS.FCSTAT = 00b / 01b / 10b. RDESC.ERRORS.FCERR = 010b (even if no DDP match).
Out of order reception of packet that matches a DDP context[1].	Auto invalidate the filter context while keeping the parameters intact. RDESC.STATUS.FCSTAT = 01b. RDESC.ERRORS.FCERR = 100b.
Received unexpected EOF / SOF: 1) New sequence ID and SOF is not SOFi. 2) Last packet in a sequence and EOF is not EOFt.	No DDP while filter context is updated (if matched and other parameters are in order). RDESC.STATUS.FCSTAT = 00b / 01b / 10b. RDESC.ERRORS.FCERR = 000b. RDESC.ERRORS.FCEOFe = 1b. RDESC.STATUS.FCEOFs = 0b.
The DMA unit gets FCoE packets while it missed the packet that was marked as first by the filter unit[2].	Filter context parameters are updated while DMA context parameters are left intact. RDESC.STATUS.FCSTAT = 01b / 10b / 11b. RDESC.ERRORS.FCERR = 011b or 101b.
Last user buffer is exhausted (not enough space for the FC payload).	The filter context is updated while DMA context is auto invalidated. RDESC.STATUS.FCSTAT = 01b / 10b / 11b. RDESC.ERRORS.FCERR = 101b.

Table 7-86. Exception Error Table [continued]

Event Description	Actions and Indications
Legacy receive queue is not enabled or no legacy receive descriptor.	The entire packet is dropped. Auto invalidates the DMA context while the filter context remains active and continues to be updated regularly. Once legacy descriptors become valid again, packets are posted to the legacy queues with the following indication. RDESC.STATUS.FCSTAT = 01b / 10b / 11b. RDESC.ERRORS.FCERR = 101b.
Packet missed (lost) by the Rx packet buffer. Normally a case when flow control is not enabled or flow control does not work properly.	The entire packet is dropped (by the Rx packet buffer). Auto invalidate the DMA context while the filter context remains active and continues to be updated regularly. Once the Rx packet buffer gets free, further Rx packets are posted to the legacy queues with the following indication. RDESC.STATUS.FCSTAT = 01b / 10b / 11b. RDESC.ERRORS.FCERR = 110b. Note that software might ignore this error when FCSTAT equals 00b.

1. Out of order might be one of the following cases. SEQ_CNT does not meet expected value. The *PARAM* field in the Rx packet does not match the DDP context. SEQ_ID keeps the same value as the previous packet in a new sequence identified by the presence of SOFi code in the *SOF* field. The SOFi is not found in the first FC frame in a context.
2. Lost sync between Filter and DMA contexts could be a result of context invalidation by software together with misbehaved target that sends packet with no host request.

7.10.3.3.15 FC Exchange Completion Interrupt

One of the performance indicators of an initiator is measured by the number of I/O operations per second it can generate. The number of FC exchanges per second is affected mainly by the CPU overhead associated with the FC exchange processing and software latencies. The number of concurrent outstanding FC exchanges supported by large FC receive is limited by hardware resources. Reducing the latency associated with processing completions increases the number of FC exchanges per second that the system supports.

7.11 Reliability

7.11.1 Memory Integrity Protection

All X550 internal memories are protected against soft errors. Most of them are covered by ECC that correct single error per memory line and detect double errors per memory line. Few of the smaller memories are covered by parity protection that detects a single error per memory line.

Single errors in memories with ECC protection are named also as correctable errors. Such errors are silently corrected. Two errors in memories with ECC protection or single error in memories with parity protection are also named as un-correctable errors. Un-correctable errors are considered as fatal errors. If an un-correctable error is detected in Tx packet data, the packet is transmitted with a CRC error. If un-correctable error is detected in Rx packet data, the packet is reported to the host (or manageability) with a CRC error. If an un-correctable error is detected anywhere else, the X550 halts the traffic and sets the ECC error interrupt. Software is then required to initiate a complete initialization cycle to resume normal operation.

7.11.2 PCIe Error Handling

For PCIe error events and error reporting see Section 3.1.5.

7.12 IPsec Support

7.12.1 Overview

This section defines the hardware requirements for the IPsec offload ability included in the X550. IPsec offload is the ability to handle (in hardware) a certain amount of the total number of IPsec flows, while the remaining are still handled by the operating system. It is the operating system's responsibility to submit to hardware the most loaded flows, to take maximum benefits of the IPsec offload in terms of CPU utilization savings. Establishing IPsec Security Associations between peers is outside the scope of this document, since it is handled by the operating system. In general, the requirements on the driver or on the operating system for enabling IPsec offload are not detailed here.

When an IPsec flow is handled in software, since the packet might be encrypted and the integrity check field already valid, and as IPv4 options might be present in the packet together with IPsec headers, the X550 processes it like it does for any other unsupported Layer4 protocol, and without performing on it any layer4 offload.

Refer to Section 4.6.13, for security offload enablement.

7.12.2 Hardware Features List

7.12.2.1 Main Features

- Offload IPsec for up to 1024 Security Associations (SA) for each of Tx and Rx:
  - On-chip storage for both Tx and Rx SA tables.
  - Tx SA index is conveyed to hardware via Tx context descriptor.
  - Deterministic Rx SA lookup according to a search key made of SPI, destination IP Address, and IP version type (IPv6 or IPv4).
- IPsec protocols:
  - IP Authentication Header (AH) protocol for authentication.
  - IP Encapsulating Security Payload (ESP) for authentication only.
  - IP ESP for both authentication and encryption, only if using the same key for both.
- Crypto engines:
  - For AH or ESP authentication only: AES-128-GMAC (128-bit key).
  - For ESP encryption and authentication: AES-128-GCM (128-bit key).
- IPsec encapsulation mode: Transport mode, with tunnel mode only in receive.
  - In Tx, packets are provided by software already encapsulated with a valid IPsec header, and:
    - for AH with blank ICV inside.
    - for ESP single send, with a valid ESP trailer and ESP ICV (blank ICV).
    - for ESP large send, without ESP trailer and without ESP ICV.

  - In Rx, packets are provided to software encapsulated with their IPsec header and for ESP with the ESP trailer and ESP ICV.
    - where up to 255 bytes of incoming ESP padding is supported, for peers that prefer hiding the packet length.
- IP versions:
  - IPv4 packets that do not include any IP option.
  - IPv6 packets that do not include any extension header (other than AH/ESP extension header).
- Rx status reported to software via Rx descriptor:
  - Packet type: AH/ESP (in the *SECSTAT* field).
  - IPsec offload done (SA match), in the *IPSSA* field.
- One Rx error reported to software via Rx descriptor in the following precedence order: no error, invalid IPsec protocol, packet length error, authentication failed (*SECERR* field).

7.12.2.2 Cross Features

- When IPsec offload is enabled, Ethernet CRC must be enabled as well by setting both TXCRCEN and RXCRCSTRP bits in the HLREG0 register
- Segmentation: Full coexistence (TCP/UDP packets only).
  - Increment IPsec Sequence Number (SN) and Initialization Vector (IV) on each additional segment.
- Checksum offload: Full coexistence (Tx and Rx)
  - IP header checksum.
  - TCP/UDP checksum.
- IP fragmentation: No IPsec offload done on IP fragments.
- RSS: Full coexistence, hash on the same fields used without IPsec (either 4-tuples or 2-tuples).
- Virtualization:
  - Full coexistence in VMDq mode.
  - in IOV mode, all IPsec registers are owned by the VMM/PF. For example, IPsec can be used for VMotion traffic.
  - No coexistence with VM-to-VM switch, IPsec packets handled in hardware are not looped back by the X550 to another VM. Tx IPsec packets destined to a local VM must be handled in software and looped back via the software switch. However, an anti-spoofing check is performed on any IPsec packet.
- DCB: Full coexistence.
  - Priority flow control, with special care to respect timing considerations.
  - Bandwidth allocation scheme enforced on IPsec packets since 802.1p field is always sent in clear text.
- FCoE: No interaction as FCoE packets are not IP packets.
- RSC: No coexistence.
- Flow Director, L3/L4 5-tuple filters, TCP SYN Filter: Full coexistence.

- Jumbo frames: When the SECTXCTRL.*STORE_FORWARD* bit is set (as required for IPsec offload), the maximum supported jumbo packet size is 9.5 KB. This limitation is valid for all packets regardless if they are offloaded by hardware or carry IPsec encapsulation altogether.
- 802.1x: No interaction.
- TimeSync:
  - TimeSync IEEE 1588v1 UDP packets must not be encapsulated as IPsec packets.
  - No interaction with TimeSync 1588v2 Layer2 packets.
- Layer2 encapsulation modes:
  - IPsec offload is not supported for flows with SNAP header.
  - IPsec offload coexists with double VLAN encapsulations.
- Tunneled IPsec packets in receive: IPsec offload supported, but no other Layer4 offload performed.
- NFS and any other Layer5 Rx filter: NFS or Layer5 packets encapsulated over ESP (whether IPsec is offloaded in hardware or not) and over a Layer4 protocol other than TCP are not parsed, nor recognized.
- SCTP Rx offload: Partial coexistence with SCTP CRC32 offload for IPsec-AH packets only.
- SCTP Tx offload: Full coexistence with SCTP CRC32 offload for both IPsec-AH and IPsec-ESP packets.
- Manageability traffic: IPsec offload ability is controlled exclusively by the host, and thus manageability traffic could use IPsec offload only if it is coordinated/configured with/by the host. For IPsec flows handled by software:
  - If manageability and host entities share some IP Address(es), manageability should coordinate any use of IPsec protocol with the host. Note it should be true for previous devices that do not offer IPsec offload.
  - If manageability and host entities have totally separate IP Addresses, manageability can use IPsec protocol (as long as it is handled by the manageability controller software)
- Header split:
  - Supported for SAs handled in hardware, IP boundary split includes the IPsec header.
  - For SAs handled in software, no header split done.
- OS2BMC:
  - As OS2BMC flows that use IPsec can not be offloaded by X550, the driver should make sure it does not accept offload requests for flows to the BMC. To support this, a BMC that uses IPsec should update the *BMCIP* and *BMCIPVAL* registers with the value of the IP Address used by the BMC. After each update, an *EICR.MNG* event is set to indicate to the driver it should read an updated value of the BMC IP Address.
  - Any IPsec offload request, whose destination IP Address matches the BMC IP Address, should be rejected.

7.12.3 Software/Hardware Demarcation

The followings items are not supported by hardware but might be supported by operating system/ driver:

- Multicast SAs.
- IPsec protocols:
  - Both AH and ESP protocols on the same SA or packet.
  - ESP for encryption only.
  - ESP for authentication and ESP for encryption, each of them using different keys and/or different crypto engines.
- Crypto engines:
  - AES-256, SHA-1, AES-128-CBC, or any other crypto algorithm.
- Tx IPsec packets encapsulated in tunnel mode
- Extended Sequence Number (ESN)
- IP versions:
  - IPv4 packets that include IP option.
  - IPv6 packets that include extension headers other than the AH/ESP extension headers.
- Anti-replay check and discard of incoming replayed packets.
- Discard of incoming dummy ESP packets (packets with protocol value 59).
- IPsec packets that are IP fragments.
- ESP padding content check.
- IPsec statistics.
- IPsec for flows with SNAP header.

Note: For SCTP and other Layer4 header types, or for tunneled packets, hardware does not care what is there when doing Rx IPsec processing. Everything after the IP/IPsec headers can be opaque to hardware (consider as IP payload). IPsec processing can be done on any packet that has a matching SA and appropriate IP options/extension headers. There is no expectation that hardware determines what is in the packet beyond the IP/IPsec headers before decrypting/authenticating the packet. The most important point is that hardware should not corrupt or drop incoming IPsec packets — in any situation. When hardware decides and starts performing IPsec offload on a packet, it should pursue the offload until the packet's end — at the price of eventually not doing other Layer3/4 offloads on it. It is always acceptable for hardware not to start doing the IPsec offload on a matched SA, if it knows it is an unsupported encapsulation. For example, one of the three cases: IPv4 option, IPv6 extensions, or SNAP.

7.12.4 IPsec Formats Exchanged Between Hardware and Software

This section describes the IPsec packet encapsulation formats used between software and hardware by an IPsec packet concerned with the offload in either Tx or Rx direction.

In Rx direction, the IPsec packets are delivered by hardware to software encapsulated as they were received from the line, whether IPsec offload was done or not, and when it was done, whether authentication/decrypting has succeeded or failed. Refer to the formats described in Section A.3.

7.12.4.1 Single Send

In Tx direction, single-send IPsec packets are delivered by software to hardware already encapsulated and formatted with their valid IPsec header and trailer contents, as they should be run over the wire — except for the *ICV* field that is filled with zeros, and the ESP payload destined to be encrypted that is provided in clear text before IPsec encryption.

7.12.4.2 Single Send with TCP/UDP Checksum Offload

For single-send ESP packets with TCP/UDP checksum offload, the checksum computing includes the TCP/UDP header and payload before hardware encryption occurred and without the ESP trailer and ESP ICV provided by software. Software provides the length of the ESP trailer plus ESP ICV in a dedicated field of the Tx context descriptor (*IPS_ESP_LEN* field) to signal hardware when to stop TCP/UDP checksum computing.

Software calculates a full checksum for the IP pseudo-header as in the usual case. The protocol value used in the IP pseudo-header must be the TCP/UDP protocol value and not the AH/ESP protocol value that appears in the IP header. This full checksum of the pseudo-header is placed in the packet data buffer at the appropriate offset for the checksum calculation.

The byte offset from the start of the DMA'ed data to the first byte to be included in the TCP/UDP checksum (the start of the TCP header) is computed as in the usual case: MACLEN+IPLEN. It assumes that IPLEN provided by software in the Tx context descriptor is the sum of the IP header length with the IPsec header length.

Note: For the IPv4 header checksum offload, hardware cannot rely on the *IPLEN* field provided by software in the Tx context descriptor, but should rely on the fact that no IPv4 options are present in the packet. Consequently, for IPsec offload packets, hardware always computes IP header checksum over a fixed amount of 20 bytes.

7.12.4.3 TSO TCP/UDP

In Tx direction, TSO IPsec packets are delivered by software to the X550 already encapsulated and formatted with only their valid IPsec header contents — except for the *ICV* field included in AH packets headers that is filled with zeros, and to the ESP payload destined to be encrypted that is provided in clear text before any encryption. No ESP trailer or ESP ICV are appended to TSO by software. It means that hardware has to append the ESP trailer and ESP ICV on each segment by itself, and to update IP total length / IP payload length accordingly.

The next header of the ESP trailer to be appended is taken from *TUCMD.L4T* field of the Tx context descriptor.

By definition TSO requires on each segment that the IP total length / IP payload length be updated, and the IP header checksum and TCP/UDP checksum be re-computed. But for the TSO of IPsec packets, the *SN* and the *IV* fields must be increased by one in hardware on each new segment (after the first one) as well.

Software calculates a partial checksum for the IP pseudo-header as in the usual case. The protocol value used in the IP pseudo-header must be the TCP/UDP protocol value and not the AH/ESP protocol value that appears in the IP header. This partial checksum of the pseudo header is placed in the packet data buffer at the appropriate offset for the checksum calculation.

The byte offset from the start of the DMA'ed data to the first byte to be included in the TCP checksum (the start of the TCP/UDP header) is computed as in the usual case: MACLEN+IPLEN. It assumes that IPLEN provided by software in the Tx context descriptor is the sum of the IP header length with the IPsec header length.

For TSO ESP packets, the TCP/UDP checksum computing includes the TCP/UDP header and payload before hardware encryption occurred and without the ESP trailer and ESP ICV appended by hardware. The X550 thus stops TCP/UDP checksum computing after the amount of bytes given by L4LEN + MSS. It is assumed that the MSS value placed by software in the Tx context descriptor specifies the maximum TCP/UDP payload segment sent per frame, not including any IPsec header or trailer — and not including the TCP/UDP header.

Note: For IPv4 header checksum computing, refer to the note in Section 7.12.4.2.

Shaded fields in the tables that follow correspond to fields that need to be updated per each segment.

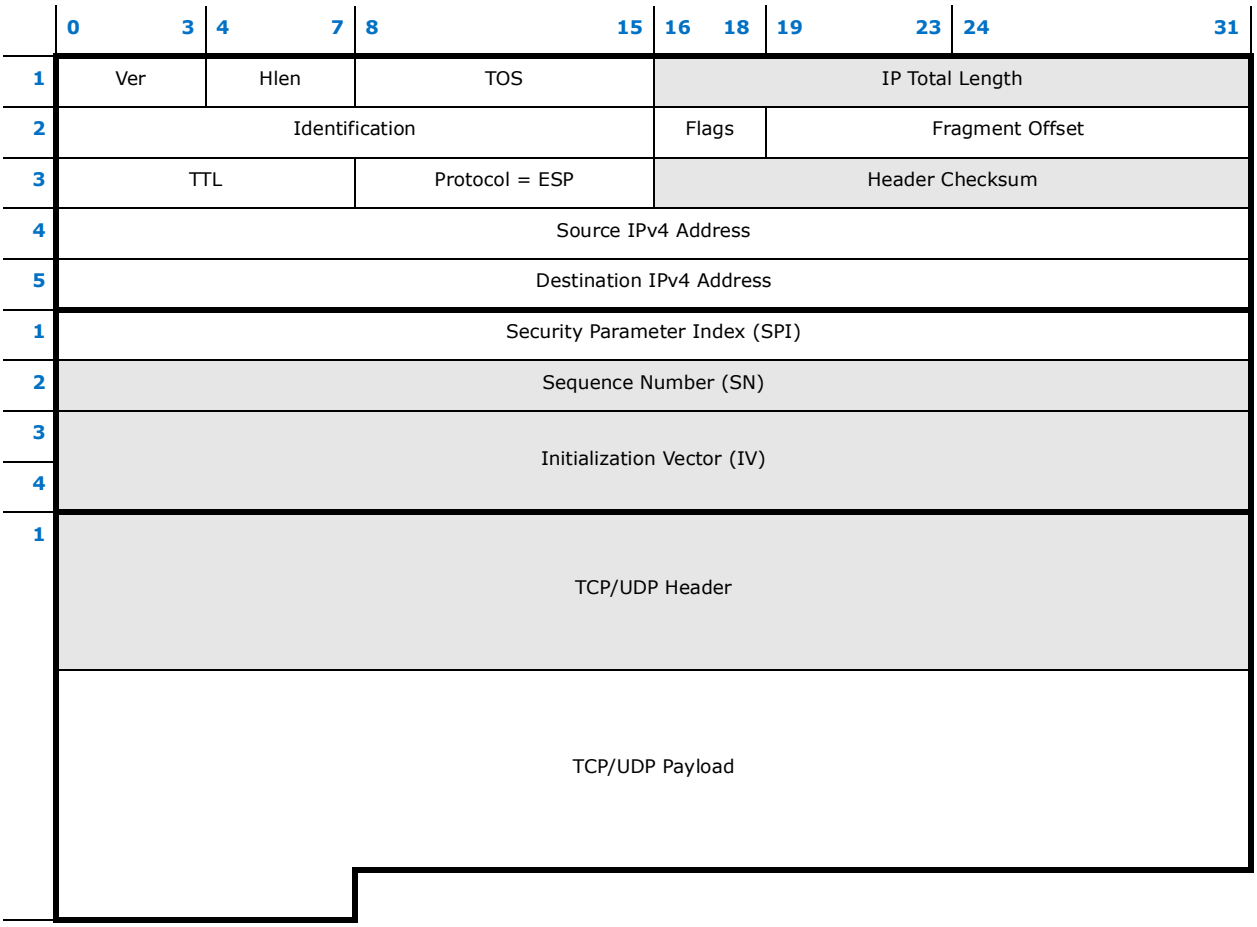


Figure 7-54. IPv4 TSO ESP Packet Provided by Software

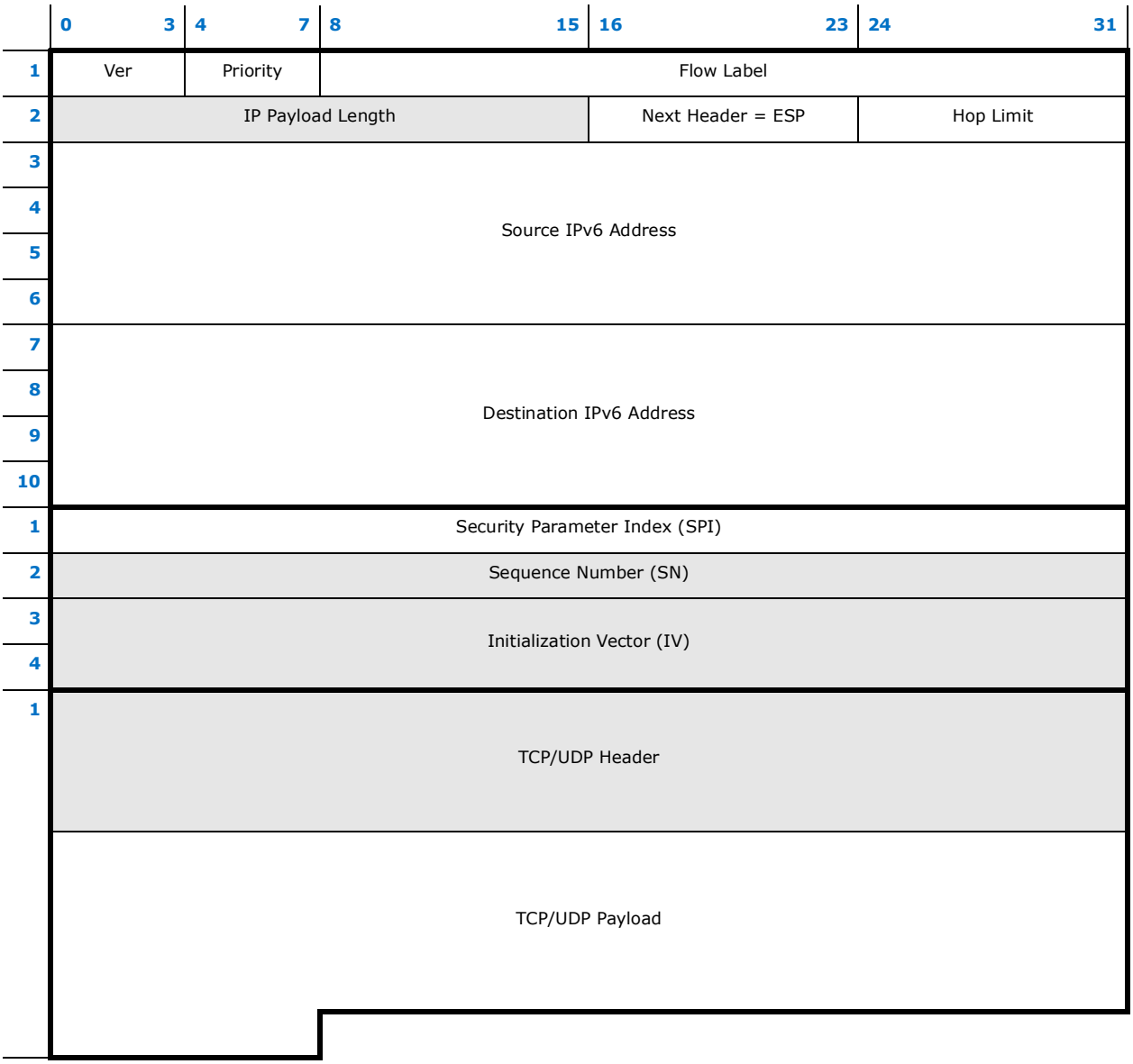


Figure 7-55. IPv6 TSO ESP Packet Provided by Software

7.12.5 Tx SA Table

IPsec offload is supported only via advanced transmit descriptors. See Section 7.2.3.2.4 for details.

7.12.5.1 Tx SA Table Structure

The Tx SA table contains additional information required by the AES-128 crypto engine to authenticate and encrypt data. This information is not run over the wire together with the IPsec packets, but is exchanged between the IPsec peers' operating system during the SA establishment process. When the IKE software does a key computation it computes four extra bytes using a pseudo-random function (it generates 20 bytes instead of 16 bytes that it needs to use as a key) and the last four bytes are used as a SALT value.

The SA table in Tx is a 1024 x 20-byte table loaded by software. Each line in the table contains the following fields:

Table 7-87. Tx SA Table

AES-128 KEY	AES-128 SALT
16 bytes	4 bytes

Refer to Section 7.12.7 for a description of the way these fields are used by the AES-128 crypto engine.

Each time an unrecoverable memory error occurs when accessing the Tx SA tables, an interrupt is generated and the transmit path is stopped until the host resets X550. Packets that have already started to be transmitted on the wire are sent with a wrong CRC.

Upon reset, the X550 clears the contents of the Tx SA table. Note that access to Tx SA table is not guaranteed for 10 µs after the reset command.

7.12.5.2 Access to Tx SA Table

7.12.5.2.1 Write Access

1. Software writes the IPSTXKEY 0...3 and/or IPSTXSALT register(s).
2. Software writes the IPSTXIDX register with the *SA_IDX* field carrying the index of the SA entry to be written, and with the *Write* bit set (*Read* bit cleared).
3. Hardware issues a Write command into the SA table, copying the IPSTXKEY (16 bytes) and the IPSTXSALT (4 bytes) registers into the table entry pointed by the *SA_IDX* field configured in IPSTXIDX register. It then clears the *Write* bit in IPSTXIDX register.
4. Software starts/resumes sending IPsec offload packets with the *IPsec SA IDX* field in the Tx context descriptor pointing to valid/invalid SA entries. A valid SA entry contains updated *KEY* and *SALT* fields currently in use by the IPsec peers.

7.12.5.2.2 Read Access

1. Software writes the IPSTXIDX register with the index of the SA entry to be read, and with the *Read* bit set (*Write* bit cleared).
2. Hardware issues a Read command from the SA table, copying into the registers the IPSTXKEY (16 bytes) and the IPSTXSALT (4 bytes) values from the table entry pointed by the *SA_IDX* field configured in the IPSTXIDX register. It then clears the *Read* bit in IPSTXIDX register.
3. Software reads the IPSTXKEY 0...3 and/or IPSTXSALT register(s).

7.12.6 Tx Hardware Flow

7.12.6.1 Single Send Without TCP/UDP Checksum Offload

1. Extract IPsec offload request from the *IPSEC* bit of the *POPTS* field in the advanced Tx transmit data descriptor.
2. If IPsec offload is required for the packet (*IPSEC* bit was set), extract the *SA_IDX*, *Encryption*, and *IPSEC_TYPE* fields from the Tx context descriptor associated to that flow.
3. Fetch the AES-128 *KEY* and *SALT* from the Tx SA entry indexed by *SA_IDX* and according to the *Encryption* and *IPSEC_TYPE* bits to determine which IPsec offload to perform.
4. For AH, zero the mutable fields.
5. Compute ICV and encryption data (if required for ESP) over the appropriate fields as specified in Section A.3, according to the operating rules described in Section 7.12.7, and making use of the AES-128 *KEY* and *SALT* fields fetched in Step 3.
6. Insert ICV at its appropriate location and replace the plaintext with the ciphertext (if required for ESP), as specified in Section A.3.

7.12.6.2 Single Send With TCP/UDP Checksum Offload

1. Extract the IPsec offload command from the *IPSEC* bit of the *POPTS* field in the advanced Tx transmit data descriptor.
2. If IPsec offload is required for the packet (*IPSEC* bit was set), extract the *SA_IDX*, *Encryption*, *IPSEC_TYPE*, and *IPS_ESP_LEN* fields from the Tx context descriptor associated to that flow.
3. Fetch the AES-128 *KEY* and *SALT* from the Tx SA entry indexed by *SA_IDX*, and according to the *Encryption* and *IPSEC_TYPE* bits to determine which IPsec offload to perform.
4. Compute the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum (the start of the TCP header) as specified in Section 7.12.4.2.
5. Compute TCP/UDP checksum until either the last byte of the DMA data or for ESP packets, up to IPS_ESP_LEN bytes before it. As in the usual case, *implicitly* pad out the data by one zeroed byte if its length is an odd number.
6. Sum the full checksum of the IP pseudo header placed by software at its appropriate location with the TCP/UDP checksum computed in Step 5. Overwrite the checksum location with the 1’s complement of the sum.
7. For AH, zero the mutable fields.

8. Compute ICV and encrypt data (if required for ESP) over the appropriate fields as specified in Section A.3, according to the operating rules described in Section 7.12.7, and making use of the AES-128 *KEY* and *SALT* fields fetched in Step 3.
9. Insert ICV at its appropriate location and replace the plaintext with the ciphertext (if required for ESP), as specified in Section A.3.

7.12.6.3 TSO TCP/UDP

1. Extract the IPsec offload command from the *IPSEC* bit of the *POPTS* field in the advanced Tx transmit data descriptor.
2. If IPsec offload is required for the packet (*IPSEC* bit was set), extract the *SA_IDX*, *Encryption*, and *IPSEC_TYPE* fields from the Tx context descriptor associated to that flow.
3. Fetch the AES-128 *KEY* and *SALT* from the Tx SA entry indexed by SA_IDX, and according to the *Encryption* and *IPSEC_TYPE* bits to determine which IPsec offload to perform.
4. Fetch the packet header from system memory, up to IPLEN+L4LEN bytes from the start of the DMA'ed data.
5. Overwrite the TCP SN with the stored accumulated TCP SN (if it is not the first segment).
6. Fetch (next) MSS bytes (or the remaining bytes up to PAYLEN for the last segment) from system memory and from the segment formed by packet header and data bytes, while storing the accumulated TCP SN.
7. Compute the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum (the start of the TCP header) as specified in Section 7.12.4.3.
8. Compute TCP/UDP checksum until the last byte of the DMA data. As in the usual case, *implicitly* pad out the data by one zeroed byte if its length is an odd number.
9. For both IPv4 and IPv6, hardware needs to factor in the TCP/UDP length (typically L4LEN+MSS) to the software-supplied pseudo header partial checksum. It then sums to obtain a full checksum of the IP pseudo header with the TCP/UDP checksum computed in Step 7. Overwrite the TCP/UDP checksum location with the 1's complement of the sum.
10. Increment by one the *AH/ESP SN* and *IV* fields on every segment (except the first segment), and store the updated *SN* and *IV* fields with other temporary statuses stored for that TSO (one TSO set of statuses per Tx queue).
11. For ESP, append the ESP trailer: 0-3 padding bytes, padding length, and next header = TCP/UDP protocol value, in a way to get the 4-byte alignment as described in Section 7.12.4.3.
12. Compute the IP total length / IP payload length and compute IPv4 header checksum as described in the note of Section 7.12.4.1. Place the results in their appropriate location.
13. For AH, zero the mutable fields.
14. Compute ICV and encryption data (if required for ESP) over the appropriate fields as specified in Section A.3, according to the operating rules described in Section 7.12.7, and making use of the AES-128 *KEY* and *SALT* field fetched in Step 3.
15. Insert ICV at its appropriate location and replace the plaintext with the ciphertext (if required for ESP), as specified in Section A.3.
16. Go back to Step 4 to process the next segment (if necessary).

7.12.7 AES-128 Operation in Tx

The AES-128-GCM crypto engine used for IPsec is referred throughout the document as an AES-128 black box, with 4-bit string inputs and 2-bit string outputs, as shown in Figure 7-56. Refer to the GCM specification for the internal details of the engine. The difference between IPsec modes resides in the set of inputs presented to the box.

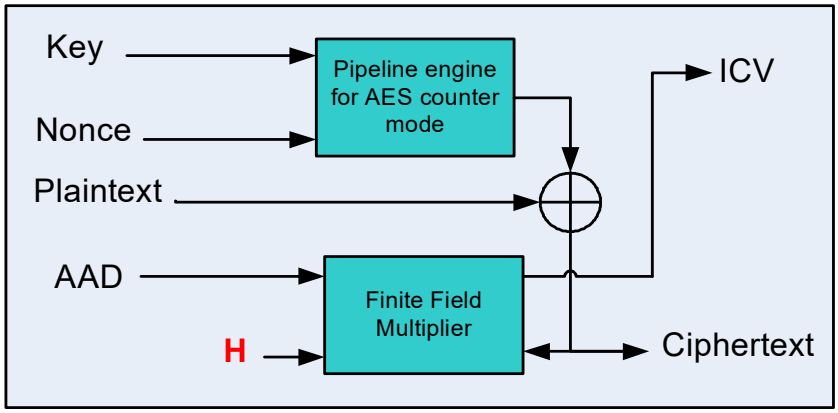


Figure 7-56. AES-128 Crypto Engine Box

- **Key** — 128-bits AES-128 *KEY* field (secret key) stored for that IPsec flow in the Tx SA table:

  Key = AES-128 KEY

- **Nonce** — 96-bits initialization vector used by the AES-128 engine, which is distinct for each invocation of the encryption operation for a fixed key. It is formed by the AES-128 *SALT* field stored for that IPsec flow in the Tx SA table, appended with the *Initialization Vector* (*IV*) field included in the IPsec packet:

  Nonce = [AES-128 SALT, IV]

  The nonce, also confusingly referred as IV in the GCM specification, is broken into two pieces — a fixed random part salt and increasing counter part IV, so the salt value goes with the packet as the fixed part. The purpose behind using the salt value is to prevent offline dictionary-type attacks in hashing case, to prevent predictable patterns in the hash.

- **AAD** — Additional Authentication Data input, which is authenticated data that must be left un-encrypted.
- **Plaintext** — Data to be both authenticated and encrypted.
- **Ciphertext** — Encrypted data, whose length is exactly that of the plaintext.
- **ICV** — 128-bit Integrity Check Value (referred also as authentication tag).
- **H** — is internally derived from the key.

Note: The square brackets in the formulas is used as a notation for concatenated fields.

7.12.7.1 AES-128-GCM for ESP — Both Authenticate and Encryption

AAD = [SPI, SN]

Plaintext = [TCP/UDP header, TCP/UDP payload, ESP trailer]

Note: Unlike other IPsec modes, in this mode, the *IV* field is used only in the nonce, and it is not included in either the plaintext or the AAD inputs.

ESP trailer does not include the *ICV* field. Refer to Section A.3.2.

7.12.7.2 AES-128-GMAC for ESP — Authenticate Only

AAD = [SPI, SN, IV, TCP/UDP header, TCP/UDP payload, ESP trailer]

Plaintext = [] = empty string, no plaintext input in this mode

Note: ESP trailer does not include the *ICV* field. Refer to Section A.3.2.

7.12.7.3 AES-128-GMAC for AH — Authenticate Only

AAD = [IP header, AH header, TCP/UDP header, TCP/UDP payload]

Plaintext = []= empty string, no plaintext input in this mode

Note: Both IP header and AH header contain mutable fields that must be zeroed prior to be entered into the engine. Refer to Section A.3.1. Among other fields, the AH header includes *SPI*, *SN*, and *IV* fields.

7.12.8 Rx Descriptors

IPsec offload is supported only via advanced receive descriptors. See Section 7.1.5.2 for details.

7.12.9 Rx SA Tables

7.12.9.1 Rx SA Tables Structure

The Rx SA tables contain additional information required by the AES-128 crypto engine to authenticate and decrypt the data. This information is not run over the wire together with the IPsec packets, but is exchanged between the IPsec peers' operating system during the SA establishment process. When the IKE software does a key computation it computes four extra bytes using a pseudo-random function (it generates 20 bytes instead of 16 bytes that it needs to use as a key) and the last four bytes are used as a salt value.

SPI is allocated by the receiving operating system in a unique manner. However, in a virtualized context, guest operating systems can allocate SPI values that collide with the SPI values allocated by the VMM/PF. Consequently, the SPI search must be completed by comparing the destination IP Address with the IP Addresses of the VMM/PF, which are stored in a separate table. Guest operating systems could thus use the proposed IPsec offload as long as their SAs are configured via the VMM/PF. It is assumed that refreshing the SAs would be done once every several minutes, and would thus not overload the VMM/PF.

There are three Rx SA tables in the X550:

- IP Address table — 128 entries
- SPI table — 1K entries
- KEY table — 1K entries

They are loaded by software via indirectly addressed CSRs, as described in Section 7.12.9.2.

Table 7-88. IP Address Table

IP Address
16 bytes

Table 7-89. SPI Table

SPI	IP Index (points to IP Address table)
4 bytes	1 bytes

Table 7-90. KEY Table

IPsec Mode	AES-128 KEY	AES-128 SALT
1 byte	16 bytes	4 bytes

The *IPsec Mode* field contains the following bits:

- VALID
- IPv6
- PROTO
- DECRYPT

It is assumed that the SPI and IP Address tables are implemented internally in CAM cells, while the KEY table uses RAM cells. When an incoming IPsec packet (which does not includes option in IPv4 or another extension header in IPv6) is detected, hardware first looks up for the destination IP Address to match one of the IP Addresses stored in the IP Address table. If there is a match, the index of that IP Address match is used together with the *SPI* field extracted from the packet for a second lookup into the SPI table. If there is again a match, the index of that SPI+IP Index match is used to retrieve the SA parameters from the KEY table. The packet is finally considered to get an SA match only after inspecting the corresponding entry in the KEY table, as long as all the following conditions are met:

- *Valid* bit is set
- *IPv6* bit match with the IP version (IPv6/IPv4) of the incoming IPsec packet
- *Proto* bit match with the AH/ESP type of the incoming IPsec packet

Each time an unrecoverable memory ECC error occurs when accessing one of the Rx SA tables, an interrupt is generated and the receive path is stopped until the host resets X550.

Upon reset, the X550 clears the contents of the Rx KEY table and software is required to invalidate the entire IP Address and SPI CAM tables by clearing their contents. Access to Rx SA tables is not guaranteed for 10 µs after the Reset command.

7.12.9.2 Access to Rx SA Tables

7.12.9.2.1 Write Access

1. Software writes the IP Address table via the IPSRXIPADDR 0...3 registers
2. Software writes the IPSRXIDX register with the following:
   a. *Table* bits combination corresponding to the Rx SA table to be written (such as 01b for IP Address table)
   b. *TB_IDX* field pointing to the index to be written within the table
   c. *Write* bit set (*Read* bit cleared)
3. Hardware issues a Write command into the Rx SA table pointed by the *Table* bits combination, copying the concerned register(s) into the entry pointed by the *TB_IDX* field configured in the IPSRXIDX register. It then clears the *Write* bit in IPSRXIDX register.
4. Software performs steps 1 to 3 twice, first for writing the SPI table via IPSRXSPI, IPSRXIPIDX registers, and second for writing the KEY table via IPSRXKEY 0...3, IPSRXSALT, and IPSRXMOD registers.

Each time an entry in the IP Address or SPI table is not valid/in-use anymore, software is required to invalidate its content by clearing it. For the IP table, an entry must be invalidated by software each time there is no more SPI entry that points to it; while for the SPI table, software must invalidate any entry as soon as it is not valid/not used anymore.

7.12.9.2.2 Read Access

1. Software writes the IPSRXIDX register with the *Table* and *TB_IDX* fields corresponding to the Rx SA table and entry to be read, and with the *Read* bit set (*Write* bit cleared).
2. Hardware issues a Read command from the Rx SA table and entry pointed by *Table* bits combination and *TB_IDX* field, copying each field into its corresponding register. It then clears the *Read* bit in IPSRXIDX register.
3. Software reads the corresponding register(s).

Caution: There is an internal limitation in that only one single Rx SA table can be read accessed by software at a time. Hence, it is recommended that the entire read process, from steps 1 to 3, be repeated successively for each Rx SA table separately.

7.12.10 Rx Hardware Flow without TCP/UDP Checksum Offload

1. Detect an IPsec header not encapsulated over a SNAP header is present without any IPv4 option or other IPv6 extension header encapsulated before it, and determine its type AH/ESP.
2. If such an IPsec header is present (as announced by the IP protocol field for IPv4 or by the next header for IPv6), extract the SPI, destination IP Address, and IP version (IPv4 or IPv6), and use these fields for the lookups into the Rx SA tables as described in Section 7.12.9.1. Also report the IPsec protocol found in the *Security* bits of the *Extended Status* field in the advanced Rx descriptor.

3. If there is a SA match for that packet, fetch the IPsec Rx mode from the SA entry, and according to the *Proto* and *Decrypt* bits determine which IPsec offload to perform. Also, set the *IPSSA* bit of the *Extended Status* field in the advanced Rx descriptor. If there was no SA match, clear the *IPSSA* bit, report no error in *Security Error* bits of the *Extended Errors* field in the advanced Rx descriptor, and stop processing the packet for IPsec.
4. If the *Proto* field recorded in the Rx SA table does not match the *IP Protocol* field (next header for IPv6) seen in the packet, report it via the *Security Error* bits of the *Extended Errors* field in the advanced Rx descriptor, and stop processing the packet for IPsec.
5. Fetch the AES-128 KEY and Salt from the matched Rx SA entry.
6. For AH, zero the mutable fields.
7. Make sure the AH/ESP header is not truncated, and for ESP, make sure the packet is 4-bytes aligned. If not, report it via the *Security* error bits of the *Extended Errors* field in the advanced Rx descriptor, but processing of the packet for IPsec might be completed (if it has already started). A truncated IPsec packet is a valid Ethernet frame (at least 64 bytes) shorter than:
   a. ESP – at least 40 bytes following the IP header (16 [ESP header] + 4 [min. padding, pad_len, NH] + 16 [ICV] + 4 [CRC])
   b. AH over IPv4 – at least 40 bytes following the IP header (20 [AH header] + 16 [ICV] + 4 [CRC])
   c. AH over IPv6 – at least 44 bytes following the IP header (20 [AH header] + 4 [ICV padding] + 16 [ICV] + 4 [CRC])
8. Compute ICV and decrypt data (if required for ESP) over the appropriate fields as specified in Section A.3, according to the operating rules described in Section 7.12.12, and making use of the AES-128 *KEY* and *SALT* fields fetched in Step 5.
9. Compare the computed ICV with the *ICV* field included in the packet at its appropriate location as specified in Section A.3, and report the comparison status match/fail via the *Security Error* bits of the *Extended Errors* field in the advanced Rx descriptor.

7.12.11 Rx Hardware Flow with TCP/UDP Checksum Offload

Perform the Rx hardware flow described in Section 7.12.10 and add the following steps:

1. Start computing the checksum from the TCP/UDP header's beginning — found according to the Rx parser logic updated for IPsec formats described in Section A.3. Do not perform Layer4 offloads if unsupported IPsec encapsulation is detected. For example, tunneled IPsec, IPv4 options or IPv6 extensions after the IPsec header.
2. For ESP, stop checksum computing before the beginning of the ESP trailer — found from the end of packet according to the padding length field content, and to the formats described in Section A.3.2. As in the usual case, *implicitly* pad out the data by one zeroed byte if its length is an odd number.
3. Store the next header extracted from the AH header/ESP trailer into the *Packet Type* field of the advanced Rx descriptor, but use the TCP/UDP protocol value in the IP pseudo header used for the TCP/UDP checksum. Also compute the TCP/UDP packet length to be inserted in the IP pseudo header (excluding any IPsec header or trailer).
4. Compare the computed checksum value with the TCP/UDP checksum included in the packet. Report the comparison status in the *Extended Errors* field of the advanced Rx descriptor.

7.12.12 AES-128 Operation in Rx

The AES-128 operation in Rx is similar to the operation in Tx, while for decryption, the encrypted payload is fed into the plaintext input, and the resulted ciphertext stands for the decrypted payload. Refer to Section 7.12.7 for the proper inputs to use in every IPsec mode.

7.12.12.1 Handling IPsec Packets in Rx

Table 7-91 lists how IPsec packets are handled according to some of their characteristics.

Table 7-91. Summary of IPsec Packets Handling in Rx

IP Fragment	IPv4 Option or IPv6 Extensions or SNAP	IP Version	SA Match	IPsec Offload in Hardware	Layer4/3 Offload in Hardware	Header Split	AH/ESP Reported in Rx Desc.
Yes	Yes	v4	Don't care	No	IP checksum only	Up to IPsec header included	Yes
Yes	Yes	v6	Don't care	No	No	Up to IP fragment extension included	No
Yes	No	v4	Don't care	No	IP checksum only	Up to IPsec header included	Yes
No	Yes	v4	Don't care	No	IP checksum only	No	Yes
No	Yes	v6	Don't care	No	No	Up to first unknown or IPsec extension header, excluded	No[1]
No	No	v4	Yes	Yes	Yes[2]	Yes[3]	Yes
No	No	v4	No	No	IP checksum only	No	Yes
No	No	v6	Yes	Yes	Yes[4]	Yes[3]	Yes
No	No	v6	No	No	No	No	Yes

1. Exception to SNAP IPsec packets that are reported as AH/ESP in Rx descriptor.
2. No Layer4 offload done on packets with IPsec error.
3. According to definition made in PSRTYPE[n] registers.
4. No Layer4 offload done on packets with IPsec error.

Chapter 8 Programming Interface

8.1 General

The X550's address space is mapped into four regions with PCI Base Address Registers listed in Table 8-1 and explained more in Section 9.2.2.16 and Section 9.2.2.17.

Table 8-1. X550 Address Regions

Addressable Content	Mapping Style	Region Size
Internal registers memories and NVM (Memory BAR)	Direct memory mapped	128 KB + NVM Size[1]
FLASH (optional)[2]	Direct memory mapped	2 MB - 4 MB
Expansion ROM (optional)	Direct memory mapped	64 KB - 512 KB
Internal registers and memories (optional)[3]	I/O window mapped	32 bytes
MSI-X (optional)	Direct memory mapped	16 KB[4]

1. The Flash size is defined by the BARCTRL register.
2. The Flash space in the Memory CSR and Expansion ROM Base Address are mapped to different Flash memory regions. Accesses to the Memory BAR at offset 256 KB are mapped to the Flash device at offset 0x0, while accesses to the Expansion ROM at offset 0x0 are mapped to the Flash region pointed by the PXE Driver Module Pointer (read from NVM word address 0x05). See Section 6.1. The Expansion ROM region has a size limited to 512 KB.
3. The internal registers and memories can be accessed though I/O space as explained in the sections that follow.
4. See Section 9.2.2.16 for the MSI-X BAR offset in 32-bit and 64-bit BAR options.

8.1.1 Memory-Mapped Access

8.1.1.1 Memory-Mapped Access to Internal Registers and Memories

The internal registers and memories can be accessed as direct memory-mapped offsets from the Memory CSR BAR. See the following sections for detailed description of the Device registers.

In IOV mode, this area is partially duplicated per VF. All replications contain only the subset of the register set that is available for VF programming.

8.1.1.2 Memory-Mapped Accesses to Flash

The external Flash can be accessed using direct memory-mapped offsets from the CSR base address register (BAR0/BAR1).

8.1.1.3 Memory-Mapped Access to MSI-X Tables

The MSI-X tables can be accessed as direct memory-mapped offsets from the base address register (BAR3). The MSI-X registers are described in Section 8.2.4.1, “PF - MSI-X Table Registers”.

In IOV mode, this area is duplicated per VF.

8.1.1.4 Memory-Mapped Access to Expansion ROM

The option ROM module located in the external Flash can also be accessed as a memory-mapped expansion ROM. Accesses to offsets starting from the Expansion ROM Base address reference the Flash, provided that access is enabled through the NVM Initialization Control Word, and if the Expansion ROM Base Address register contains a valid (non-zero) base memory address.

8.1.2 I/O-Mapped Access

All internal registers and memories can be accessed using I/O operations. I/O accesses are supported only if an I/O Base Address is allocated and mapped (BAR2), the BAR contains a valid (non-zero value), and I/O address decoding is enabled in the PCIe configuration.

When an I/O BAR is mapped, the I/O address range allocated opens a 32-byte window in the system I/O address map. Within this window, two I/O addressable registers are implemented: IOADDR and IODATA. The IOADDR register is used to specify a reference to an internal register or memory, and then the IODATA register is used to access it at the address specified by IOADDR:

Table 8-2. I/O Map

Offset	Abbreviation	Name	RW	Size
0x00	IOADDR	Internal Register, Internal Memory, or Flash Location Address. 0x00000-0x1FFFF – Internal registers/memories 0x20000-0x7FFFF – Undefined	RW	4 bytes
0x04	IODATA	Data field for reads or writes to the internal register, internal memory, or Flash location as identified by the current value in IOADDR. All 32 bits of this register are read/write-able.	RW	4 bytes
0x08-0x1F	Reserved	Reserved.	O	4 bytes

8.1.2.1 IOADDR (I/O Offset 0x00; RW)

The IOADDR register must always be written as a DWord access. Writes that are less than 32 bits are ignored. Reads of any size returns a DWord of data. However, the chipset or CPU might only return a subset of that DWord.

For software programmers, the IN and OUT instructions must be used to cause I/O cycles to be used on the PCIe bus. Because writes must be to a 32-bit quantity, the source register of the OUT instruction must be EAX (the only 32-bit register supported by the OUT command). For reads, the IN instruction can have any size target register, but it is recommended that the 32-bit EAX register be used.

Bits 31 through 20 are ignored by the device and should be set to zero by the software. At hardware reset (LAN_PWR_GOOD) or PCI reset, this register value resets to 0x00000000. Once written, the value is retained until the next write or reset.

8.1.2.2 IODATA (I/O Offset 0x04; RW)

The IODATA register must always be written as a DWord access when the IOADDR register contains a value for the internal register and memories (such as 0x00000-0x1FFFC). In this case, writes that are less than 32 bits are ignored.

Writes and reads to IODATA when the IOADDR register value is in an undefined range (0x20000-0x7FFFC) should not be performed. Results cannot be determined.

Note: There are no special software timing requirements on accesses to IOADDR or IODATA. All accesses are immediate except when data is not readily available or acceptable. In this case, the X550 delays the results through normal bus methods (like split transaction or transaction retry).

Because a register/memory read or write takes two I/O cycles to complete, software must provide a guarantee that the two I/O cycles occur as an atomic operation. Otherwise, results can be non-deterministic from the software viewpoint.

8.1.2.3 Undefined I/O Offsets

I/O offsets 0x08 through 0x1F are considered to be reserved offsets with the I/O window. DWord reads from these addresses return 0xFFFF, while writes to these addresses are discarded.

8.1.3 Configuration Access to Internal Registers and Memories

To support legacy pre-boot 16-bit operating environments without requiring I/O address space, the X550 enables accessing CSRs via the configuration address space by mapping IOADDR and IODATA registers into the configuration address space. The register mapping in this case is listed in Table 8-3.

Table 8-3. IOADDR and IODATA in Configuration Address Space

Configuration Address	Abbreviation	Name	RW	Size
0x98	IOADDR	Internal register or internal memory location address. 0x00000-0x1FFFF – Internal registers and memories. 0x20000-0x7FFFFF – Undefined.	RW	4 bytes
0x9C	IODATA	Data field for reads or writes to the internal register or internal memory location as identified by the current value in IOADDR. All 32 bits of this register can be read from or written to.	RW	4 bytes

Software writes data to an internal CSR via the configuration space in the following manner:

1. CSR address is written to the IOADDR register where:
   - Bit 31 (IOADDR.*Configuration IO Access Enable*) of the IOADDR register should be set to 1b.
   - Bits 30:0 of IOADDR should hold the actual address of the internal register or memory being written to.
2. Data to be written is written into the IODATA register.
   - The IODATA register is used as a window to the register or memory address specified by IOADDR register. As a result, the data written to the IODATA register is written into the CSR pointed to by bits 30:0 of the IOADDR register.
3. IOADDR.*Configuration IO Access Enable* is cleared to avoid unintentional CSR read operations (that might cause clear by read) by other applications scanning the configuration space.

Software reads data from an internal CSR via the configuration space in the following manner:

1. CSR address is written to the IOADDR register where:
   — Bit 31 (IOADDR.*Configuration IO Access Enable*) of the IOADDR register should be set to 1b.
   - Bits 30:0 of IOADDR should hold the actual address of the internal register or memory being read.
2. CSR value is read from the IODATA register.
   - The IODATA register is used as a window to the register or memory address specified by the IOADDR register. As a result, the data read from the IODATA register is the data of the CSR pointed to by bits 30:0 of the IOADDR register.
3. IOADDR.*Configuration IO Access Enable* is cleared to avoid un-intentional CSR read operations (that might cause clear by read) by other applications scanning the configuration space.

Notes:

- When functioning in a D3 state, software should not attempt to access CSRs via the IOADDR and IODATA configuration registers.
- To enable CSR access via configuration space, software should set bit 31 to 1b (IOADDR.*Configuration IO Access Enable*) in the IOADDR register. Software should clear bit 31 of the IOADDR register after completing CSR access to avoid an unintentional clear-by-read operation by another application scanning the configuration address space.
- Bit 31 of the IOADDR register (IOADDR.*Configuration IO Access Enable*) has no effect when initiating access via I/O address space.
- Configuration access to 0x9C (IODATA) without setting bit 31 of the IOADDR register (IOADDR.Configuration Access Enable) must not result in an unsupported request.
- I/O-mapped access and CSR access via PCIe configuration space are mutually exclusive operating modes, and therefore PCIE_CNF.*IO_SUP* NVM bit must be cleared when the PCIE_CAPSUP.*CSR_CONF_EN* NVM bit is set, and vice versa.

8.1.4 Register Terminology

Type	Description
RC	**Read Clear** A register bit with this attribute is cleared after read. Writes have no effect on the bit value.
RO	**Read Only** If a register is read only, writes to this register have no effect.
ROS	**Read Only Status** Writes to ROS fields has no effect. The value of the field can change due to hardware events.
RSV	**Reserved** Field can return any value on read access and must be set to its initialized value on write access unless specified differently in the field description.
RW	**Read/Write** A register with this attribute can be read and written. If written since reset, the value read reflects the value written.
RW1C	**Read/Write Clear** A register with this attribute can be read and written. However, a write of a 1b clears (sets to 0b) the corresponding bit and a write of a 0b has no effect.
RW/RC	**Read/Write and Read Clear**
RWS	**Read Write Set** Register that is set to 1b by software by writing a 1b to the register, and cleared to 0b by hardware.
WO	**Write Only** Reading this register might not return a meaningful value.

8.1.5 VF Registers Allocated per Queue

Depending on configuration, each pool has 2, 4, or 8 queues allocated to it. Note that in IOV mode, any queues not allocated to a VF are allocated to the PF. The registers assigned to a queue are accessible both in its VF address space and in the PF address space. This section describes the address mapping of registers that belong to queues.

Section 7.7.9.7.2 defines the correspondence of queue indices between the PF and the VFs. For example, when in configuration for 32 VFs, queues 124-127 in the PF correspond to queues [3:0] of VF# 31.

Note: The queue pair of one VF should be assigned to the PF, therefore the maximum usable number of VFs is 63.

The queues are enumerated in each VF from 0 (i.e. [1:0], [3:0], or [7:0]). If a queue is allocated to a VF, its corresponding registers are accessible in the VF CSR space. Each register is allocated an address in the VF (relative to its base) according to its index in the VF space. Therefore, the registers of queue 0 in each VF are allocated the same addresses, which equal the addresses of the same registers for queue 0 in the PF. For example, RDH[0] in the VF space has the same relative address in each VF and in the PF (address 0x01010).

8.1.6 Non-Queue VF Registers

Registers that do not correspond to a specific queue are allocated addresses in the VF space according to these rules:

- Registers that are read-only by the VF (e.g. STATUS) have the same address in the VF space as in the PF space.
- Registers allocated per pool are accessed in the VF in the same location as pool [0] in the PF address space.
- Registers that are RW by the VF (e.g. CTRL) are replicated in the PF, one per VF, in adjacent addresses.

Note: Since the VF address space is limited to 16 KB, any register that resides above that address in the PF space cannot reside in the same address in the VF space and is therefore allocated in another location in the VF.

8.2 Device Registers - PF

8.2.1 BAR0 Registers Summary

Table 8-4. PF - General Control Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00000000	CTRL	Device Control Register	8.2.2.1.1
0x00000008	STATUS	Device Status Register	8.2.2.1.2
0x00000018	CTRL_EXT	Extended Device Control Register	8.2.2.1.3
0x00000020	ESDP	Extended SDP Control	8.2.2.1.4
0x00000028	PHY_GPIO	PHY GPIO Register	8.2.2.1.5
0x00000030	MAC_GPIO	MAC GPIO Register	8.2.2.1.6
0x00000100	PHYINT_STATUS0	PHY Interrupt Status Register 0	8.2.2.1.7
0x00000104	PHYINT_STATUS1	PHY Interrupt Status Register 1	8.2.2.1.8
0x00000108	PHYINT_STATUS2	PHY Interrupt Status Register 2	8.2.2.1.9
0x00000200	LEDCTL	LED Control	8.2.2.1.10
0x00005078	EXVET	Extended VLAN Ether Type - Receive	8.2.2.1.11
0x00008224	EXVET_T	Extended VLAN Ether Type - Transmit	8.2.2.1.12
0x00010150	FACTPS	Function Active and Power State to Manageability	8.2.2.1.13
0x00010200	GRC	General Receive Control	8.2.2.1.14
0x00010208	DEV_FUNC_EN	Device and Functions Enable Control	8.2.2.1.15
0x00015F58	I2CCMD	SFP I^2C Command	8.2.2.1.16
0x00015F5C	I2CPARAMS	SFP I^2C Parameters	8.2.2.1.17

Table 8-5. PF - NVM Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00010010	EEC	EEPROM Mode Control Register	8.2.2.2.1
0x0001001C	FLA	Flash Access Register	8.2.2.2.2
0x00010110	EEMNGCTL	Manageability EEPROM-Mode Control Register	8.2.2.2.3
0x00010114	EEMNGDATA	Manageability EEPROM-Mode Read/Write Data	8.2.2.2.4
0x00010118	FLMNGCTL	Manageability Flash Control Register	8.2.2.2.5
0x0001011C	FLMNGDATA	Manageability Flash Read/Write Data	8.2.2.2.6
0x00015F2C	JEDEC_ID_1	JEDEC ID 1	8.2.2.2.7

Table 8-6. PF - Flow Control Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00003200 + 0x4*n, n=0...3	FCTTVN[n]	Flow Control Transmit Timer Value n	8.2.2.3.1
0x00003220 + 0x4*n, n=0...7	FCRTL[n]	Flow Control Receive Threshold Low	8.2.2.3.2
0x00003260 + 0x4*n, n=0...7	FCRTH[n]	Flow Control Receive Threshold High	8.2.2.3.3
0x000032A0	FCRTV	Flow Control Refresh Threshold Value	8.2.2.3.4
0x00003D00	FCCFG	Flow Control Configuration	8.2.2.3.5
0x00004294	MFLCN	MAC Flow Control Register	8.2.2.3.6
0x0000431C	PFCTOP	Priority Flow Control Type Opcode	8.2.2.3.7
0x0000CE00	TFCS	Transmit Flow Control Status	8.2.2.3.8

Table 8-7. PF - PCIe Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00011028	PCI_STATUS1	PCIe Function Status 1	8.2.2.4.1
0x00011040	PCI_FWCTRL	PCIe Firmware Control	8.2.2.4.2
0x00011044	PCI_CSRTO	PCIe CSR Access Timeout	8.2.2.4.3
0x00011050	GCR_EXT	PCIe Control Extended Register	8.2.2.4.4
0x00011054	PCI_FLASHTO	PCI Flash Access Timeout	8.2.2.4.5
0x00011070	FUNC_RID	Function Requester ID Information Register	8.2.2.4.6
0x00011098	PCI_REVID	PCIe Revision ID	8.2.2.4.7
0x00011140	PCI_PCIERR	PCIe Errors Reported	8.2.2.4.8
0x00011520	PCI_ICAUSE	PCIe Interrupt Cause	8.2.2.4.9
0x00011528	PCI_IENA	PCIe Interrupts Enable	8.2.2.4.10
0x00011530	PCI_VMINDEX	PCIe VM Pending Index	8.2.2.4.11
0x00011538	PCI_VMPEND	PCIe VM Pending Status	8.2.2.4.12
0x00011540	PCI_DREVID	PCIe Default Revision ID	8.2.2.4.13
0x00011544	PCI_BYTCTH	PCIe Byte Counter High	8.2.2.4.14
0x00011548	PCI_BYTCTL	PCIe Byte Counter Low	8.2.2.4.15
0x00011734	PCI_LCBDATA	PCIe LCB Data Port	8.2.2.4.16
0x00011788	PCI_LCBADD	PCIe LCB Address Port	8.2.2.4.17
0x00011800	PCI_GSCL_1	PCIe Statistic Control Register #1	8.2.2.4.18
0x00011804	PCI_GSCL_2	PCIe Statistic Control Registers #2	8.2.2.4.19
0x00011810 + 0x4*n, n=0...3	PCI_GSCL_5_8[n]	PCIe Statistic Control Register #5...#8	8.2.2.4.20
0x00011820 + 0x4*n, n=0...3	PCI_GSCN_0_3[n]	PCIe Statistic Counter Registers #0...#3	8.2.2.4.21

Table 8-8. PF - PCIe Configuration Space Setting Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00011000	PCI_CNF	PCIe PF Configuration	8.2.2.5.1
0x00011008	PCI_PFDEVID	PCIe PF Device ID	8.2.2.5.2
0x00011010	PCI_VFDEVID	PCIe VF Device ID	8.2.2.5.3
0x00011038	PCI_CLASS	PCIe Storage Class	8.2.2.5.4
0x0001103C	PCI_VENDORID	PCIe Vendor ID	8.2.2.5.5
0x00011048	PCI_LBARCTRL	PCI BAR Control	8.2.2.5.6
0x00011058	PCI_SUBSYSID	PCIe Subsystem ID	8.2.2.5.7
0x00011060	PCI_PWRDATA	PCIe Power Data Register	8.2.2.5.8
0x00011078	PCI_SERH	PCIe Serial Number MAC Address High	8.2.2.5.9
0x00011080	PCI_CAPCTRL	PCIe Capabilities Control	8.2.2.5.10
0x00011088	PCI_CAPSUP	PCIe Capabilities Support	8.2.2.5.11
0x00011090	PCI_LINKCAP	PCIe Link Capabilities	8.2.2.5.12
0x000110A0	PCI_PMSUP	PCIe PM Support	8.2.2.5.13
0x000110A8	PCI_VFSUP	PCIe VF Capabilities Support	8.2.2.5.14
0x000110B8	PCI_GLBL_CNF	PCIe Global Config	8.2.2.5.15
0x000110E8	PCI_UPADD	PCIe Upper Address	8.2.2.5.16
0x000110F0	PCI_SERL	PCIe Serial Number MAC Address Low	8.2.2.5.17
0x000110F8	PCI_CNF2	PCIe Global Config 2	8.2.2.5.18

Table 8-9. PF - Interrupt Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00000800	EICR	Extended Interrupt Cause Register	8.2.2.6.1
0x00000808	EICS	Extended Interrupt Cause Set Register	8.2.2.6.2
0x00000810	EIAC	Extended Interrupt Auto Clear Register	8.2.2.6.3
0x00000820 + 0x4*n, n=0...23 and 0x00012300 + 0x4*(n-24), n=24...128	EITR[n]	Extended Interrupt Throttle Registers	8.2.2.6.4
0x00000880	EIMS	Extended Interrupt Mask Set/Read Register	8.2.2.6.5
0x00000888	EIMC	Extended Interrupt Mask Clear Register	8.2.2.6.6
0x00000890	EIAM	Extended Interrupt Auto Mask Enable Register	8.2.2.6.7
0x00000894	EITRSEL	MSIX to EITR Select	8.2.2.6.8
0x00000898	GPIE	General Purpose Interrupt Enable	8.2.2.6.9
0x00000900 + 0x4*n, n=0...63	IVAR[n]	Interrupt Vector Allocation Registers	8.2.2.6.10
0x00000A00	IVAR_MISC	Miscellaneous Interrupt Vector Allocation	8.2.2.6.11
0x00000A90	EICS1	Extended Interrupt Cause Set Registers 1	8.2.2.6.12
0x00000A94	EICS2	Extended Interrupt Cause Set Registers 2	8.2.2.6.13
0x00000AA0	EIMS1	Extended Interrupt Mask Set/Read Registers	8.2.2.6.14

Table 8-9. PF - Interrupt Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00000AA4	EIMS2	Extended Interrupt Mask Set/Read Registers	8.2.2.6.15
0x00000AB0	EIMC1	Extended Interrupt Mask Clear Registers 1	8.2.2.6.16
0x00000AB4	EIMC2	Extended Interrupt Mask Clear Registers 2	8.2.2.6.17
0x00000AD0	EIAM1	Extended Interrupt Auto Mask Enable Registers 1	8.2.2.6.18
0x00000AD4	EIAM2	Extended Interrupt Auto Mask Enable Registers 2	8.2.2.6.19
0x00012000 + 0x4*n, n=0...128	RSCINT[n]	RSC Enable Interrupt	8.2.2.6.20

Table 8-10. PF - MSI-X Table Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x000110C0 + 0x4*n, n=0...7	PBACL[n]	MSI-X PBA Clear	8.2.2.7.1
0x000110C8 + 0x4*n, n=0...5	VFPBACL[n]	VF MSI-X PBA Clear	8.2.2.7.2

Table 8-11. PF - Receive Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00005000	RXCSUM	Receive Checksum Control	8.2.2.8.1
0x00005008	RFCTL	Receive Filter Control Register	8.2.2.8.2
0x0000507C	VXLANCTRL	VXLAN Control	8.2.2.8.3
0x00005080	FCTRL	Filter Control Register	8.2.2.8.4
0x00005084	ETAG_ETYPE	E-tag EtherType Register	8.2.2.8.5
0x00005088	VLNCTRL	VLAN Control Register	8.2.2.8.6
0x00005090	MCSTCTRL	Multicast Control Register	8.2.2.8.7
0x00005128 + 0x4*n, n=0...7	ETQF[n]	EType Queue Filter	8.2.2.8.8
0x00005200 + 0x4*n, n=0...127	MTA[n]	Multicast Table Array	8.2.2.8.9
0x00005400 + 0x8*n, n=0...15	RAL_ALIAS[n]	Receive Address Low	8.2.2.8.10
0x00005404 + 0x8*n, n=0...15	RAH_ALIAS[n]	Receive Address High	8.2.2.8.11
0x00005818	MRQC_ALIAS	Multiple Receive Queues Command Register	8.2.2.8.12
0x0000A000 + 0x4*n, n=0...127	VFTA[n]	VLAN Filter Table Array	8.2.2.8.13
0x0000A200 + 0x8*n, n=0...127	RAL[n]	Receive Address Low	8.2.2.8.14
0x0000A204 + 0x8*n, n=0...127	RAH[n]	Receive Address High	8.2.2.8.15
0x0000A600 + 0x4*n, n=0...255	MPSAR[n]	MAC Pool Select Array	8.2.2.8.16
0x0000EA00 + 0x4*n, n=0...63	PSRTYPE[n]	Packet Split Receive Type Register	8.2.2.8.17
0x0000EB00 + 0x4*n, n=0...31	RETA[n]	Redirection Table	8.2.2.8.18
0x0000EB80 + 0x4*n, n=0...9	RSSRK[n]	RSS Random Key Register	8.2.2.8.19
0x0000EC00 + 0x4*n, n=0...7	ETQS[n]	EType Queue Select	8.2.2.8.20
0x0000EC30	SYNQF	SYN Packet Queue Filter	8.2.2.8.21
0x0000EC70	RQTC	RSS Queues per Traffic Class Register	8.2.2.8.22

Table 8-11. PF - Receive Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Number
0x0000EC80	MRQC	Multiple Receive Queues Command Register	8.2.2.8.23
0x0000EE80 + 0x4*n, n=0...95	ERETA[n]	Extended Redirection Table	8.2.2.8.24
0x00018000 + 0x4*n + 0x40*m, n=0...15, m=0...63	VFRSSRK[n,m]	Per-Pool RSS Random Key Register	8.2.2.8.25
0x00019000 + 0x4*n + 0x40*m, n=0...15, m=0...63	VFRETA[n,m]	Per-Pool Redirection Table	8.2.2.8.26

Table 8-12. PF - Receive DMA Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00001000 + 0x40*n, n=0...63 and 0x0000D000 + 0x40*(n-64), n=64...127	RDBAL[n]	Receive Descriptor Base Address Low	8.2.2.9.1
0x00001004 + 0x40*n, n=0...63 and 0x0000D004 + 0x40*(n-64), n=64...127	RDBAH[n]	Receive Descriptor Base Address High	8.2.2.9.2
0x00001008 + 0x40*n, n=0...63 and 0x0000D008 + 0x40*(n-64), n=64...127	RDLEN[n]	Receive Descriptor Length	8.2.2.9.3
0x00001010 + 0x40*n, n=0...63 and 0x0000D010 + 0x40*(n-64), n=64...127	RDH[n]	Receive Descriptor Head	8.2.2.9.4
0x00001014 + 0x40*n, n=0...63 and 0x0000D014 + 0x40*(n-64), n=64...127	SRRCTL[n]	Split Receive Control Registers	8.2.2.9.5
0x00001018 + 0x40*n, n=0...63 and 0x0000D018 + 0x40*(n-64), n=64...127	RDT[n]	Receive Descriptor Tail	8.2.2.9.6
0x00001028 + 0x40*n, n=0...63 and 0x0000D028 + 0x40*(n-64), n=64...127	RXDCTL[n]	Receive Descriptor Control	8.2.2.9.7
0x0000102C + 0x40*n, n=0...63 and 0x0000D02C + 0x40*(n-64), n=64...127	RSCCTL[n]	RSC Control	8.2.2.9.8
0x00002100 + 0x4*n, n=0...15	SRRCTL_ALIAS[n]	Split Receive Control Registers	8.2.2.9.9
0x00002F00	RDRXCTL	Receive DMA Control Register	8.2.2.9.10
0x00003000	RXCTRL	Receive Control Register	8.2.2.9.11
0x00003C00 + 0x4*n, n=0...7	RXPBSIZE[n]	Receive Packet Buffer Size	8.2.2.9.12

Table 8-13. PF - Transmit Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00004950 + 0x4*n, n=0...7	TXPBTHRESH[n]	Tx Packet Buffer Threshold	8.2.2.10.1
0x00004A80	DMATXCTL	DMA Tx Control	8.2.2.10.2
0x00004A88	DTXTCPFLGL	DMA Tx TCP Flags Control Low	8.2.2.10.3
0x00004A8C	DTXTCPFLGH	DMA Tx TCP Flags Control High	8.2.2.10.4
0x00006000 + 0x40*n, n=0...127	TDBAL[n]	Transmit Descriptor Base Address Low	8.2.2.10.5
0x00006004 + 0x40*n, n=0...127	TDBAH[n]	Transmit Descriptor Base Address High	8.2.2.10.6
0x00006008 + 0x40*n, n=0...127	TDLEN[n]	Transmit Descriptor Length	8.2.2.10.7
0x00006010 + 0x40*n, n=0...127	TDH[n]	Transmit Descriptor Head	8.2.2.10.8
0x00006018 + 0x40*n, n=0...127	TDT[n]	Transmit Descriptor Tail	8.2.2.10.9
0x00006028 + 0x40*n, n=0...127	TXDCTL[n]	Transmit Descriptor Control	8.2.2.10.10
0x00006038 + 0x40*n, n=0...127	TDWBAL[n]	Tx Descriptor Completion Write Back Address Low	8.2.2.10.11
0x0000603C + 0x40*n, n=0...127	TDWBAH[n]	Tx Descriptor Completion Write Back Address High	8.2.2.10.12
0x00008100	DTXMXSZRQ	DMA Tx TCP Max Allow Size Requests	8.2.2.10.13
0x00008120	MTQC	Multiple Transmit Queues Command Register	8.2.2.10.14
0x0000CC00 + 0x4*n, n=0...7	TXPBSIZE[n]	Transmit Packet Buffer Size	8.2.2.10.15
0x0000CD10	MNGTXMAP	Manageability Transmit TC Mapping	8.2.2.10.16
0x00017100	TAG_ETYPE	Tags EtherTypes	8.2.2.10.17

Table 8-14. PF - DCB Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00002140 + 0x4*n, n=0...7	RTRPT4C[n]	DCB Receive Packet Plane T4 Config	8.2.2.11.1
0x00002430	RTRPCS	DCB Receive Packet Plane Control and Status	8.2.2.11.2
0x00003020	RTRUP2TC	DCB Receive User Priority to Traffic Class	8.2.2.11.3
0x00004904	RTTDQSEL	DCB Transmit Descriptor Plane Queue Select	8.2.2.11.4
0x00004908	RTTDT1C	DCB Transmit Descriptor Plane T1 Config	8.2.2.11.5
0x00004910 + 0x4*n, n=0...7	RTTDT2C[n]	DCB Transmit Descriptor Plane T2 Config	8.2.2.11.6
0x00004980	RTTQCNRM	DCB Transmit QCN Rate-Scheduler MMW	8.2.2.11.7
0x00004984	RTTQCNRC	DCB Transmit QCN Rate-Scheduler Config	8.2.2.11.8
0x00004988	RTTQCNRS	DCB Transmit QCN Rate-Scheduler Status	8.2.2.11.9
0x0000498C	RTTQCNRR	DCB Transmit QCN Rate Reset	8.2.2.11.10
0x00004A90	RTTQCNTG	DCB Transmit QCN Tagging	8.2.2.11.11
0x000082E0 + 0x4*n, n=0...3	TXLLQ[n]	Strict Low Latency Tx Queues	8.2.2.11.12
0x0000C800	RTTUP2TC	DCB Transmit User Priority to Traffic Class	8.2.2.11.13
0x0000CD00	RTTPCS	DCB Transmit Packet Plane Control and Status	8.2.2.11.14
0x0000CD20 + 0x4*n, n=0...7	RTTPT2C[n]	DCB Transmit Packet Plane T2 Config	8.2.2.11.15

Table 8-15. PF - TPH Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x0000100C + 0x40*n, n=0...63 and 0x0000D00C + 0x40*(n-64), n=64...127	TPH_RXCTRL[n]	Rx TPH Control Register	8.2.2.12.1
0x00002200 + 0x4*n, n=0...15	TPH_RXCTRL_ALIAS[n]	Rx TPH Control Register	8.2.2.12.2
0x0000600C + 0x40*n, n=0...127	TPH_TXCTRL[n]	Tx TPH Control Registers	8.2.2.12.3

Table 8-16. PF - Timers Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x0000004C	TCPTIMER	TCP Timer	8.2.2.13.1

Table 8-17. PF - FCoE Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00002410	FCPTRL	FC Indirect DMA Context: User Descriptor PTR Low	8.2.2.14.1
0x00002414	FCPTRH	FC Indirect DMA Context: User Descriptor PTR High	8.2.2.14.2
0x00002418	FCBUFF	FC Indirect DMA Context: Buffer Control	8.2.2.14.3
0x00002420	FCDMARW	FC Indirect DMA Context: Receive DMA RW	8.2.2.14.4
0x00005100	FCRXCTRL	FC Receive Control	8.2.2.14.5
0x00005108	FCFLT	FC Indirect Filter Context: Control	8.2.2.14.6
0x0000510C	FCSMAC	FC Indirect Filter Context: Source MAC Address	8.2.2.14.7
0x00005110	FCFLTRW	FC Indirect Filter Context: RW Control	8.2.2.14.8
0x00005114	FCD_ID	FC Indirect Filter Context: D_ID Register	8.2.2.14.9
0x000051D8	FCPARAM	FC Indirect Filter Context: Offset Parameter	8.2.2.14.10
0x0000ED00	FCRECTL	FCoE Redirection Control	8.2.2.14.11
0x0000ED10 + 0x4*n, n=0...31	FCRETA[n]	FCoE Redirection Table	8.2.2.14.12
0x00020000 + 0x4*n + 0x10*m, n=0...3, m=0...2047	FCDDC[n,m]	FCoE Direct DMA Context	8.2.2.14.13
0x00028000 + 0x4*n + 0x10*m, n=0...3, m=0...2047	FCDFC[n,m]	FCoE Direct Filter Context	8.2.2.14.14
0x00030000 + 0x4*n, n=0...2047	FCDFCD[n]	FCoE Direct Filter Context D_ID	8.2.2.14.15

Table 8-18. PF - Flow Director Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x0000EE00	FDIRCTRL	Flow Director Filters Control Register	8.2.2.15.1
0x0000EE0C + 0x4*n, n=0...2	FDIRSIPV6[n]	Flow Director Filters Source IPv6	8.2.2.15.2
0x0000EE18	FDIRIPSA	Flow Director Filters IP SA	8.2.2.15.3
0x0000EE1C	FDIRIPDA	Flow Director Filters IP DA	8.2.2.15.4

Table 8-18. PF - Flow Director Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Number
0x0000EE20	FDIRPORT	Flow Director Filters Port	8.2.2.15.5
0x0000EE24	FDIRVLAN	Flow Director Filters VLAN and FLEX Bytes	8.2.2.15.6
0x0000EE28	FDIRHASH	Flow Director Filters Hash Signature	8.2.2.15.7
0x0000EE2C	FDIRCMD	Flow Director Filters Command Register	8.2.2.15.8
0x0000EE38	FDIRFREE	Flow Director Filters Free	8.2.2.15.9
0x0000EE3C	FDIRDIP4M	Flow Director Filters DIPv4 Mask	8.2.2.15.10
0x0000EE40	FDIRSIP4M	Flow Director Filters Source IPv4 Mask	8.2.2.15.11
0x0000EE44	FDIRTCPM	Flow Director Filters TCP Mask	8.2.2.15.12
0x0000EE48	FDIRUDPM	Flow Director Filters UDP Mask	8.2.2.15.13
0x0000EE4C	FDIRLEN	Flow Director Filters Length	8.2.2.15.14
0x0000EE50	FDIRUSTAT	Flow Director Filters Usage Statistics	8.2.2.15.15
0x0000EE54	FDIRFSTAT	Flow Director Filters Failed Usage Statistics	8.2.2.15.16
0x0000EE58	FDIRMATCH	Flow Director Filters Match Statistics	8.2.2.15.17
0x0000EE5C	FDIRMISS	Flow Director Filters Miss Match Statistics	8.2.2.15.18
0x0000EE68	FDIRHKEY	Flow Director Filters Lookup Table HASH Key	8.2.2.15.19
0x0000EE6C	FDIRSKEY	Flow Director Filters Signature HASH Key	8.2.2.15.20
0x0000EE70	FDIRM	Flow Director Filters Other Mask	8.2.2.15.21
0x0000EE74	FDIRIP6M	Flow Director Filters IPv6 Mask	8.2.2.15.22
0x0000EE78	FDIRSCTPM	Flow Director Filters SCTP Mask	8.2.2.15.23

Table 8-19. PF - MAC Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00004240	HLREG0	Highlander Control 0 Register	8.2.2.16.1
0x00004244	HLREG1	Highlander Status 1 Register	8.2.2.16.2
0x00004248	PAP	Pause and Pace Register	8.2.2.16.3
0x0000425C	MSCA	MDI Single Command and Address	8.2.2.16.4
0x00004260	MSRWD	MDI Single Read and Write Data	8.2.2.16.5
0x00004268	MAXFRS	Max Frame Size	8.2.2.16.6
0x000042A4	LINKS	Link Status Register	8.2.2.16.7
0x000042D0	MMNGC	MAC Manageability Control Register	8.2.2.16.8
0x00004330	MACC	MAC Control Register	8.2.2.16.9

Table 8-20. PF - Statistics Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00001030 + 0x40*n, n=0...15	QPRC[n]	Queue Packets Received Count	8.2.2.17.1
0x00001034 + 0x40*n, n=0...15	QBRC_L[n]	Queue Bytes Received Count Low	8.2.2.17.2
0x00001038 + 0x40*n, n=0...15	QBRC_H[n]	Queue Bytes Received Count High	8.2.2.17.3
0x00001430 + 0x40*n, n=0...15	QPRDC[n]	Queue Packets Received Drop Count	8.2.2.17.4
0x00002300 + 0x4*n, n=0...31	RQSMR[n]	Receive Queue Statistic Mapping Registers	8.2.2.17.5
0x0000241C	FCOERPDC	FCoE Rx Packets Dropped Count	8.2.2.17.6
0x00002424	FCLAST	Fiber Channel Last Error Count	8.2.2.17.7
0x00002428	FCOEPRC	FCoE Packets Received Count	8.2.2.17.8
0x0000242C	FCOEDWRC	FCOE DWord Received Count	8.2.2.17.9
0x00002F40	RXDSTATCTRL	Rx DMA Statistic Counter Control	8.2.2.17.10
0x00002F50	RXDGPC	DMA Good Rx Packet Counter	8.2.2.17.11
0x00002F54	RXDGBCL	DMA Good Rx Byte Counter Low	8.2.2.17.12
0x00002F58	RXDGBCH	DMA Good Rx Byte Counter High	8.2.2.17.13
0x00002F5C	RXDDPC	DMA Duplicated Good Rx Packet Counter	8.2.2.17.14
0x00002F60	RXDDBCL	DMA Duplicated Good Rx Byte Counter Low	8.2.2.17.15
0x00002F64	RXDDBCH	DMA Duplicated Good Rx Byte Counter High	8.2.2.17.16
0x00002F68	RXLPBKPC	DMA Good Rx LPBK Packet Counter	8.2.2.17.17
0x00002F6C	RXLPBKBCL	DMA Good Rx LPBK Byte Counter Low	8.2.2.17.18
0x00002F70	RXLPBKBCH	DMA Good Rx LPBK Byte Counter High	8.2.2.17.19
0x00002F74	RXDLPBKPC	DMA Duplicated Good Rx LPBK Packet Counter	8.2.2.17.20
0x00002F78	RXDLPBKBCL	DMA Duplicated Good Rx LPBK Byte Counter Low	8.2.2.17.21
0x00002F7C	RXDLPBKBCH	DMA Duplicated Good Rx LPBK Byte Counter High	8.2.2.17.22
0x00002F90	B2OGPRC	BMC2OS Packets Received by Host	8.2.2.17.23
0x00004000	CRCERRS	CRC Error Count	8.2.2.17.24
0x00004004	ILLERRC	Illegal Byte Error Count	8.2.2.17.25
0x00004008	ERRBC	Error Byte Packet Count	8.2.2.17.26
0x00004010	MSPDC	MAC Short Packet Discard Count	8.2.2.17.27
0x00004018	MBSDC	Bad SFD Count	8.2.2.17.28
0x00004034	MLFC	MAC Local Fault Count	8.2.2.17.29
0x00004038	MRFC	MAC Remote Fault Count	8.2.2.17.30
0x00004040	RLEC	Receive Length Error Count	8.2.2.17.31
0x0000405C	PRC64	Packets Received [64 Bytes] Count	8.2.2.17.32
0x00004060	PRC127	Packets Received [65-127 Bytes] Count	8.2.2.17.33
0x00004064	PRC255	Packets Received [128-255 Bytes] Count	8.2.2.17.34
0x00004068	PRC511	Packets Received [256-511 Bytes] Count	8.2.2.17.35
0x0000406C	PRC1023	Packets Received [512-1023 Bytes] Count	8.2.2.17.36
0x00004070	PRC1522	Packets Received [1024 to Max Bytes] Count	8.2.2.17.37

Table 8-20. PF - Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00004074	GPRC	Good Packets Received Count	8.2.2.17.38
0x00004078	BPRC	Broadcast Packets Received Count	8.2.2.17.39
0x0000407C	MPRC	Multicast Packets Received Count	8.2.2.17.40
0x00004080	GPTC	Good Packets Transmitted Count	8.2.2.17.41
0x00004088	GORCL	Good Octets Received Count Low	8.2.2.17.42
0x0000408C	GORCH	Good Octets Received Count High	8.2.2.17.43
0x00004090	GOTCL	Good Octets Transmitted Count Low	8.2.2.17.44
0x00004094	GOTCH	Good Octets Transmitted Count High	8.2.2.17.45
0x000040A4	RUC	Receive Undersize Count	8.2.2.17.46
0x000040A8	RFC	Receive Fragment Count	8.2.2.17.47
0x000040AC	ROC	Receive Oversize Count	8.2.2.17.48
0x000040B0	RJC	Receive Jabber Count	8.2.2.17.49
0x000040B4	MNGPRC	Management Packets Received Count	8.2.2.17.50
0x000040B8	MNGPDC	Management Packets Dropped Count	8.2.2.17.51
0x000040C0	TORL	Total Octets Received Low	8.2.2.17.52
0x000040C4	TORH	Total Octets Received High	8.2.2.17.53
0x000040D0	TPR	Total Packets Received	8.2.2.17.54
0x000040D4	TPT	Total Packets Transmitted	8.2.2.17.55
0x000040D8	PTC64	Packets Transmitted [64 Bytes] Count	8.2.2.17.56
0x000040DC	PTC127	Packets Transmitted [65-127 Bytes] Count	8.2.2.17.57
0x000040E0	PTC255	Packets Transmitted [128-255 Bytes] Count	8.2.2.17.58
0x000040E4	PTC511	Packets Transmitted [256-511 Bytes] Count	8.2.2.17.59
0x000040E8	PTC1023	Packets Transmitted [512-1023 Bytes] Count	8.2.2.17.60
0x000040EC	PTC1522	Packets Transmitted [Greater than 1024 Bytes] Count	8.2.2.17.61
0x000040F0	MPTC	Multicast Packets Transmitted Count	8.2.2.17.62
0x000040F4	BPTC	Broadcast Packets Transmitted Count	8.2.2.17.63
0x00004120	XEC	XSUM Error Count	8.2.2.17.64
0x00004140 + 0x4*n, n=0...7	PXONRXCNT[n]	Priority XON Received Count	8.2.2.17.65
0x00004160 + 0x4*n, n=0...7	PXOFFRXCNT[n]	Priority XOFF Received Count	8.2.2.17.66
0x00004180	BUPRC	Total Unicast Packets Received (BMC copy)	8.2.2.17.67
0x00004184	BMPRC	BMC Total Multicast Packets Received	8.2.2.17.68
0x00004188	BBPRC	Total Broadcast Packets Received (BMC copy)	8.2.2.17.69
0x0000418C	BUPTC	Total Unicast Packets Transmitted (BMC copy)	8.2.2.17.70
0x00004190	BMPTC	BMC Total Multicast Packets Transmitted	8.2.2.17.71
0x00004194	BBPTC	Total Broadcast Packets Transmitted (BMC copy)	8.2.2.17.72
0x00004198	BCRCERRS	BMC FCS Receive Errors	8.2.2.17.73
0x0000419C	BXONRXC	BMC Pause XON Frames Received	8.2.2.17.74

Table 8-20. PF - Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Number
0x000041A4	LXONRXCNT	Link XON Received Count	8.2.2.17.75
0x000041A8	LXOFFRXCNT	Link XOFF Received Count	8.2.2.17.76
0x000041B0	RXNFGPC	Good Rx Non-Filtered Packet Counter	8.2.2.17.77
0x000041B4	RXNFGBCL	Good Rx Non-Filter Byte Counter Low	8.2.2.17.78
0x000041B8	RXNFGBCH	Good Rx Non-Filter Byte Counter High	8.2.2.17.79
0x000041C0	B2OSPC	BMC2OS Packets Sent by BMC	8.2.2.17.80
0x000041C4	O2BGPTC	OS2BMC Packets Received by BMC	8.2.2.17.81
0x000041E0	BXOFFRXC	BMC Pause XOFF Frames Received	8.2.2.17.82
0x000041E4	BXONTXC	BMC Pause XON Frames Transmitted	8.2.2.17.83
0x000041E8	BXOFFTXC	BMC Pause XOFF Frames Transmitted	8.2.2.17.84
0x000041F0	B2OSDPC	Sideband Receive Dropped Packet Count	8.2.2.17.85
0x00005118	FCCRC	Fiber Channel CRC Error Count	8.2.2.17.86
0x00006030 + 0x40*n, n=0...15	QPTC_ALIAS[n]	Queue Packets Transmitted Count	8.2.2.17.87
0x00008600 + 0x4*n, n=0...31	TQSM[n]	Transmit Queue Statistic Mapping Registers	8.2.2.17.88
0x00008680 + 0x4*n, n=0...15	QPTC[n]	Queue Packets Transmitted Count	8.2.2.17.89
0x00008700 + 0x8*n, n=0...15	QBTC_L[n]	Queue Bytes Transmitted Count Low	8.2.2.17.90
0x00008704 + 0x8*n, n=0...15	QBTC_H[n]	Queue Bytes Transmitted Count High	8.2.2.17.91
0x00008780	SSVPC	Switch Security Violation Packet Count	8.2.2.17.92
0x00008784	FCOEPTC	FCoE Packets Transmitted Count	8.2.2.17.93
0x00008788	FCOEDWTC	FCoE DWord Transmitted Count	8.2.2.17.94
0x000087A0	TXDGPC	DMA Good Tx Packet Counter	8.2.2.17.95
0x000087A4	TXDGBCL	DMA Good Tx Byte Counter Low	8.2.2.17.96
0x000087A8	TXDGBCH	DMA Good Tx Byte Counter High	8.2.2.17.97
0x000087B0	O2BSPC	OS2BMC Packets Transmitted by Host	8.2.2.17.98

Table 8-21. PF - Wake-Up and Proxy Control Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00005800	WUC	Wake-Up Control Register	8.2.2.18.1
0x00005808	WUFC	Wake-Up Filter Control Register	8.2.2.18.2
0x00005810	WUS	Wake-Up Status Register	8.2.2.18.3
0x00005838	IPAV	IP Address Valid	8.2.2.18.4
0x00005840 + 0x8*n, n=0...3	IP4AT[n]	IPv4 Address Table	8.2.2.18.5
0x00005880 + 0x4*n, n=0...3	IP6AT[n]	IPv6 Address Table	8.2.2.18.6
0x00005900	WUPL	Wake-Up Packet Length	8.2.2.18.7
0x00005990 + 0x4*n, n=0...11	IP6AT_EXT[n]	IPv6 Address Table Extended	8.2.2.18.8
0x00005A00 + 0x4*n, n=0...31	WUPM[n]	Wake-Up Packet Memory (128 Bytes)	8.2.2.18.9

Table 8-21. PF - Wake-Up and Proxy Control Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00005F60	PROXYS	Proxying Status Register	8.2.2.18.10
0x00005F64	PROXYFC	Proxying Filter Control Register	8.2.2.18.11
0x00009000 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009600 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_DW_EVEN[n,m]	Filter DW Even	8.2.2.18.12
0x00009004 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009604 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_DW_ODD[n,m]	Filter DW Odd	8.2.2.18.13
0x00009008 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009608 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_MASK[n,m]	Filter Mask	8.2.2.18.14
0x0000900C + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x0000960C + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_LENGTH[n,m]	Filter Length	8.2.2.18.15

Table 8-22. PF - Management Filters Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00005010 + 0x4*n, n=0...7	MAVTV[n]	Management VLAN TAG Value	8.2.2.19.1
0x00005030 + 0x4*n, n=0...7	MFUTP[n]	Management Flex UDP/TCP Ports	8.2.2.19.2
0x00005050 + 0x4*n, n=0...3	BMCIP[n]	BMC IP Address Register	8.2.2.19.3
0x00005060	BMCIPVAL	BMC IP Valid Register	8.2.2.19.4
0x00005160 + 0x4*n, n=0...7	MDEF_EXT[n]	Manageability Decision Filters Ext	8.2.2.19.5
0x00005190 + 0x4*n, n=0...3	METF[n]	Management Ethernet Type Filters	8.2.2.19.6
0x00005820	MANC	Management Control Register	8.2.2.19.7
0x00005864	MNGONLY	Manageability Only Traffic	8.2.2.19.8
0x00005890 + 0x4*n, n=0...7	MDEF[n]	Manageability Decision Filters	8.2.2.19.9
0x000058B0 + 0x4*n + 0x10*m, n=0...3, m=0...3	MIPAF[n,m]	Manageability IP Address Filter	8.2.2.19.10
0x00005910 + 0x8*n, n=0...3	MMAL[n]	Manageability Ethernet MAC Address Low	8.2.2.19.11
0x00005914 + 0x8*n, n=0...3	MMAH[n]	Manageability Ethernet MAC Address High	8.2.2.19.12
0x00009400 + 0x10*n, n=0...15	FTFT_FILTER_EVEN[n]	FTFT Filter DW Even words	8.2.2.19.13
0x00009404 + 0x10*n, n=0...15	FTFT_FILTER_ODD[n]	FTFT Filter DW Odd words	8.2.2.19.14
0x00009408 + 0x10*n, n=0...15	FTFT_FILTER0_MASK[n]	FTFT Filter Mask	8.2.2.19.15
0x0000940C + 0x10*n, n=0...15	FTFT_FILTER0_LENGTH[n]	FTFT Filter Length	8.2.2.19.16

Table 8-23. PF - Manageability (ARC subsystem) HOST Interface Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00010140	SWSM	Software Semaphore Register	8.2.2.20.1
0x00010148	FWSM	Firmware Semaphore Register	8.2.2.20.2
0x00010160	SW_FW_SYNC	Software-Firmware Synchronization	8.2.2.20.3
0x00015800 + 0x4*n, n=0...447	ARCRAM[n]	Host ARC Data RAM	8.2.2.20.4
0x00015F00	HICR	HOST Interface Control Register	8.2.2.20.5
0x00015F40	FWRESETCNT	Firmware Resets Count	8.2.2.20.6

Table 8-24. PF - Time Sync (IEEE 1588) Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x0000003C	TSSDP	Time Sync SDP Configuration Register	8.2.2.21.1
0x00005120	RXMTRL	Rx Message Type Register Low	8.2.2.21.2
0x00005188	TSYNCRXCTL	Rx Time Sync Control Register	8.2.2.21.3
0x000051A4	RXSTMPH	Rx Timestamp High	8.2.2.21.4
0x000051E8	RXSTMPL	Rx Timestamp Low	8.2.2.21.5
0x00008C00	TSYNCTXCTL	Tx Time Sync Control Register	8.2.2.21.6
0x00008C04	TXSTMPL	Tx Timestamp Value Low	8.2.2.21.7
0x00008C08	TXSTMPH	Tx Timestamp Value High	8.2.2.21.8
0x00008C0C	SYSTIMEL	System Time Register Low	8.2.2.21.9
0x00008C10	SYSTIMEH	System Time Register High	8.2.2.21.10
0x00008C14	TIMEINCA	Increment Attributes Register	8.2.2.21.11
0x00008C18	TIMADJ	Time Adjustment Offset Register	8.2.2.21.12
0x00008C20	TSAUXC	TimeSync Auxiliary Control Register	8.2.2.21.13
0x00008C24	TRGTTIMEL0	Target Time Register 0 Low	8.2.2.21.14
0x00008C28	TRGTTIMEH0	Target Time Register 0 High	8.2.2.21.15
0x00008C2C	TRGTTIMEL1	Target Time Register 1 Low	8.2.2.21.16
0x00008C30	TRGTTIMEH1	Target Time Register 1 High	8.2.2.21.17
0x00008C34	FREQOUT0	Frequency Out 0 Control Register	8.2.2.21.18
0x00008C38	FREQOUT1	Frequency Out 1 Control Register	8.2.2.21.19
0x00008C3C	AUXSTMPL0	Auxiliary Timestamp 0 Register Low	8.2.2.21.20
0x00008C40	AUXSTMPH0	Auxiliary Timestamp 0 Register High	8.2.2.21.21
0x00008C44	AUXSTMPL1	Auxiliary Timestamp 1 Register Low	8.2.2.21.22
0x00008C48	AUXSTMPH1	Auxiliary Timestamp 1 Register High	8.2.2.21.23
0x00008C58	SYSTIMR	System Time Register Residue	8.2.2.21.24
0x00008C60	TSICR	Time Sync Interrupt Cause Register	8.2.2.21.25
0x00008C68	TSIM	Time Sync Interrupt Mask Register	8.2.2.21.26

Table 8-25. PF - Virtualization PF Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00000700 + 0x4*n, n=0...1	PFVFLREC[n]	PF VFLR Events Clear	8.2.2.22.1
0x00000710 + 0x4*n, n=0...3	PFMBICR[n]	PF Mailbox Interrupt Causes Register	8.2.2.22.2
0x00000720 + 0x4*n, n=0...1	PFMBIMR[n]	PF Mailbox Interrupt Mask Register	8.2.2.22.3
0x00002F04	PFQDE	PF Queue Drop Enable Register	8.2.2.22.4
0x00002FA4	LMVM_RX	Last Malicious VM - Rx	8.2.2.22.5
0x00002FA8	LVMMC_RX	Last VM Misbehavior Cause - Rx	8.2.2.22.6
0x00002FB0 + 0x4*n, n=0...3	WQBR_RX[n]	Wrong Queue Behavior Register - Rx	8.2.2.22.7
0x00004B00 + 0x4*n, n=0...63	PFMAILBOX[n]	PF Mailbox	8.2.2.22.8
0x000050B0	PFFLPL	Filter Local Packets Low	8.2.2.22.9
0x000050B4	PFFLPH	Filter Local Packets High	8.2.2.22.10
0x00005180 + 0x4*n, n=0...1	PFVMTXSW[n]	PF VM Tx Switch Loopback Enable	8.2.2.22.11
0x000051B0	PFVTCTL	PF Virtual Control Register	8.2.2.22.12
0x000051E0 + 0x4*n, n=0...1	PFVFRE[n]	PF VF Receive Enable	8.2.2.22.13
0x00008000 + 0x4*n, n=0...63	PFVMVIR[n]	PF VM VLAN Insert Register	8.2.2.22.14
0x00008108	LVMMC_TX	Last VM Misbehavior Cause - Tx	8.2.2.22.15
0x00008110 + 0x4*n, n=0...1	PFVFTE[n]	PF VF Transmit Enable	8.2.2.22.16
0x00008124	LMVM_TX	Last Malicious VM - Tx	8.2.2.22.17
0x00008130 + 0x4*n, n=0...3	WQBR_TX[n]	Wrong Queue Behavior Register - Tx	8.2.2.22.18
0x00008200 + 0x4*n, n=0...7	PFVFSPOOF[n]	PFVF Anti-Spoof Control	8.2.2.22.19
0x00008220	PFDTXGSWC	PF DMA Tx General Switch Control	8.2.2.22.20
0x00008790	PFVMECM0	PF VM 0:31 Error Count Mask	8.2.2.22.21
0x00008794	PFVMECM1	PF VM 32:63 Error Count Mask	8.2.2.22.22
0x0000F000 + 0x4*n, n=0...63	PFVML2FLT[n]	PF VM L2Control Register	8.2.2.22.23
0x0000F100 + 0x4*n, n=0...63	PFVLVF[n]	PF VM VLAN Pool Filter	8.2.2.22.24
0x0000F200 + 0x4*n, n=0...127	PFVLVFB[n]	PF VM VLAN Pool Filter Bitmap	8.2.2.22.25
0x0000F400 + 0x4*n, n=0...127	PFUTA[n]	PF Unicast Table Array	8.2.2.22.26
0x0000F600 + 0x4*n, n=0...3	PFMRCTL[n]	PF Mirror Rule Control	8.2.2.22.27
0x0000F610 + 0x4*n, n=0...7	PFMRVLAN[n]	PF Mirror Rule VLAN	8.2.2.22.28
0x0000F630 + 0x4*n, n=0...7	PFMRVM[n]	PF Mirror Rule Pool	8.2.2.22.29
0x00017000 + 0x4*n, n=0...63	PFVMTIR[n]	PF VM Tag Insert Register	8.2.2.22.30

Table 8-26. PF - Power Management Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00002400	DMACR	DMA Coalescing Control Register	8.2.2.23.1
0x00002404	DMCTLX	DMA Coalescing Time to Lx Request	8.2.2.23.2
0x00003300 + 0x4*n, n=0...7	DMCTH[n]	DMA Coalescing Threshold	8.2.2.23.3
0x000041F4	TLPIC	EEE Tx LPI Count	8.2.2.23.4
0x000041F8	RLPIC	EEE Rx LPI Count	8.2.2.23.5
0x00004380	EEE_SU	Energy Efficient Ethernet (EEE) Setup Register	8.2.2.23.6
0x00004398	EEE_STAT	Energy Efficient Ethernet (EEE) STATUS	8.2.2.23.7
0x000043A0	EEER	Energy Efficient Ethernet (EEE) Register	8.2.2.23.8
0x00011708	LTRC	Latency Tolerance Reporting (LTR) Control	8.2.2.23.9
0x00015F20	DMCMNGTH	DMA Coalescing Management Threshold	8.2.2.23.10

Table 8-27. PF - Security Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00008800	SECTXCTRL	Security Tx Control	8.2.2.24.1
0x00008804	SECTXSTAT	Security Tx Status	8.2.2.24.2
0x00008808	SECTXBUFFAF	Security Tx Buffer Almost Full	8.2.2.24.3
0x00008810	SECTXMINIFG	Security Tx Buffer Minimum IFG	8.2.2.24.4
0x00008D00	SECRXCTRL	Security Rx Control	8.2.2.24.5
0x00008D04	SECRXSTAT	Security Rx Status	8.2.2.24.6

Table 8-28. PF - IPsec Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00008900	IPSTXIDX	IPsec Tx Index	8.2.2.25.1
0x00008904	IPSTXSALT	IPsec Tx Salt Register	8.2.2.25.2
0x00008908 + 0x4*n, n=0...3	IPSTXKEY[n]	IPsec Tx Key Registers	8.2.2.25.3
0x00008E00	IPSRXIDX	IPsec Rx Index	8.2.2.25.4
0x00008E04 + 0x4*n, n=0...3	IPSRXIPADDR[n]	IPsec Rx IP Address Register	8.2.2.25.5
0x00008E14	IPSRXSPI	IPsec Rx SPI Register	8.2.2.25.6
0x00008E18	IPSRXIPIDX	IPsec Rx SPI Register IP Index	8.2.2.25.7
0x00008E1C + 0x4*n, n=0...3	IPSRXKEY[n]	IPsec Rx Key Register	8.2.2.25.8
0x00008E2C	IPSRXSALT	IPsec Rx Salt Register	8.2.2.25.9
0x00008E30	IPSRXMOD	IPsec Rx Mode Register	8.2.2.25.10

Table 8-29. PF - VF Registers Mapping in the PF Space Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00000300 + 0x4*n, n=0...63	VFCTRL[n]	VF Control Register	8.2.2.26.1
0x00000B00 + 0x4*n, n=0...63	VFEICR[n]	VF Extended Interrupt Cause	8.2.2.26.2
0x00000C00 + 0x4*n, n=0...63	VFEICS[n]	VF Extended Interrupt Cause Set	8.2.2.26.3
0x00000D00 + 0x4*n, n=0...63	VFEIMS[n]	VF Extended Interrupt Mask Set/Read	8.2.2.26.4
0x00000E00 + 0x4*n, n=0...63	VFEIMC[n]	VF Extended Interrupt Mask Clear	8.2.2.26.5
0x0000101C + 0x40*n, n=0...63	VFGPRC[n]	VF Good Packets Received Count	8.2.2.26.6
0x00001020 + 0x40*n, n=0...63	VFGORC_LSB[n]	VF Good Octets Received Count Low	8.2.2.26.7
0x00003400 + 0x4*n, n=0...63	VFMRQC[n]	VF Multiple Receive Queues Command Register	8.2.2.26.8
0x00004C00 + 0x4*n, n=0...63	VFMAILBOX[n]	VF Mailbox	8.2.2.26.9
0x00004D00 + 0x4*n, n=0...63	VFEIAM[n]	VF Extended Interrupt Auto Mask Enable	8.2.2.26.10
0x00004E00 + 0x4*n, n=0...63	VFIVAR_MISC[n]	VF Interrupt Vector Allocation Registers Misc	8.2.2.26.11
0x00008300 + 0x4*n, n=0...63	VFGPTC[n]	VF Good Packets Transmitted Count	8.2.2.26.12
0x00008400 + 0x8*n, n=0...63	VFGOTC_LSB[n]	VF Good Octets Transmitted Count LSB	8.2.2.26.13
0x00008404 + 0x8*n, n=0...63	VFGOTC_MSB[n]	VF Good Octets Transmitted Count MSB	8.2.2.26.14
0x0000D01C + 0x40*n, n=0...63	VFMPRC[n]	VF Multicast Packets Received Count	8.2.2.26.15
0x0000D020 + 0x40*n, n=0...63	VFGORC_MSB[n]	VF Good Octets Received Count High	8.2.2.26.16
0x00012500 + 0x4*n, n=0...63	VFIVAR[n]	VF Interrupt Vector Allocation Registers	8.2.2.26.17
0x00013000 + 0x4*n + 0x40*m, n=0...15, m=0...63	PFMBMEM[n,m]	PF Mailbox Memory	8.2.2.26.18

8.2.2 Detailed Register Descriptions - PF BAR0

8.2.2.1 PF - General Control Registers

8.2.2.1.1 Device Control Register - CTRL (0x00000000)

CTRL is also mapped to address 0x00004 to maintain compatibility with predecessors.

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	00b	RSV	Reserved.
PCIE_MASTER_DISABLE	2	0b	RW	**PCIe Master Disable** When set, the device blocks new master requests, including manageability requests, by using this function. Once no master requests are pending by using this function, the *PCIE_MASTER_ENABLE_STATUS* bit is cleared. ***Note:*** After any change to this bit, the host must read that the bit has been modified as expected before reading STATUS.*PCIE_MASTER_ENABLE_STATUS* bit (Section 8.2.2.1.2).
LRST	3	0b	RW	**Link Reset** This bit performs a reset of the MAC, PHY, and the entire 10GBase-T Controller (software reset) resulting in a state nearly approximating the state following a power-up reset or internal PCIe reset, except for the system PCI configuration. Normally 0b. Writing 1b initiates the reset. This bit is self-clearing. Also referred to as MAC reset.
RESERVED	25:4	0x0	RSV	Reserved.
RST	26	0b	RW	**Device Reset** This bit performs a complete reset of the device, resulting in a state nearly approximating the state following a power-up reset or internal PCIe reset, except for the system PCI configuration. Normally 0b. Writing 1b initiates the reset. This bit is self-clearing. Also referred to as a software reset or global reset.
RESERVED	31:27	0x0	RSV	Reserved.

Note: *LRST* and *RST* can be used to globally reset the entire 10GBase-T Controller. This register is provided primarily as a software mechanism to recover from an indeterminate or suspected hung hardware state. Most registers (receive, transmit, interrupt, statistics, etc.) and state machines are set to their power-on reset values, approximating the state following a power-on or PCI reset. However, PCIe configuration registers are not reset, thereby leaving the device mapped into system memory space and accessible by a software device driver. To ensure that a global device reset has fully completed and that the device responds to subsequent accesses, programmers must wait approximately 1 ms after setting before attempting to check if the bit has cleared, or to access (read or write) any other device register.

8.2.2.1.2 Device Status Register - STATUS (0x00000008)

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	00b	RSV	Reserved.
LAN_ID	3:2	00b	RO	**LAN ID** Provides software a mechanism to determine the device LAN identifier for this MAC. 00b = LAN 0 01b = LAN 1 All other values are reserved.
RESERVED	6:4	000b	RSV	Reserved.
LINKUP	7	0b	RW	**Linkup Status Indication** This bit is useful for IOV mode. The PF software driver sets it according to the LINKS register and PHY state. It is reflected in the VFSTATUS register indicating link up to the VF drivers.
RESERVED	9:8	00b	RSV	Reserved.
NUM_VFS	17:10	0x0	RO	**Num VFs** This field reflects the value of the Num VFs in the IOV capability structure. ***Note:*** Bit 17 is always 0b.
IOV_ACTIVE	18	0b	RO	**IOV Active** This bit reflects the value of the VF Enable (*VFE*) bit in the IOV Control/Status register.
PCIE_MASTER_ENABLE_STATUS	19	1b	RO	**PCIe Master Enable Status** This is a status bit of the appropriate CTRL.*PCIE_MASTER_DISABLE* bit (Section 8.2.2.1.1). 0b = Associated LAN function does not issue any master request and all previously issued requests are complete. 1b = Associated LAN function can issue master requests.
THERMAL_SENSOR_STATUS	20	0b	RO	**Thermal Sensor Status** This indication is received from the internal PHY. 0b = Thermal sensor normal state. 1b = Thermal sensor current event.
RESERVED	31:21	0x0	RSV	Reserved. Reads as 0b.

8.2.2.1.3 Extended Device Control Register - CTRL_EXT (0x00000018)

Field	Bit(s)	Init.	Type	Description
RESERVED	13:0	0x0	RSV	Reserved.
PFRSTD	14	0b	SC	**PF Reset Done** When set, the *RSTI* bit in all the VFMailbox registers are cleared and the *RSTD* bit in all the VFMailbox registers are set.
RESERVED	16:15	00b	RSV	Reserved.
RO_DIS	17	0b	RW	**Relaxed Ordering Disable** When set to 1b, the device does not request any relaxed ordering transactions. When this bit is cleared and the *Enable Relaxed Ordering* bit in the Device Control register is set, the device requests relaxed ordering transactions per queues as configured in the TPH_RXCTRL[n] and TPH_TXCTRL[n] registers (Section 8.2.2.12.1 and Section 8.2.2.12.3, respectively).
RESERVED	25:18	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
EXTENDED_VLAN	26	0b	RW	**Extended VLAN** When set, all incoming Rx packets are expected to have at least one VLAN with the EtherType as defined in the EXVET register (Section 8.2.2.1.11). This bit should only be reset by a PCIe reset, and should only be changed while Tx and Rx processes are stopped. Should be set to the same value as DMATXCTL.GDV (Section 8.2.2.10.2)
RESERVED	27	0b	RSV	Reserved.
DRV_LOAD	28	0b	RW	**Driver Load** Software device driver loaded and the corresponding network interface is enabled. This bit should be set by the software device driver after it was loaded, and cleared when it unloads or at PCIe reset. The Manageability Controller (MC) loads this bit as an indication that the software device driver successfully loaded to it.
RESERVED	31:29	000b	RSV	Reserved.

8.2.2.1.4 Extended SDP Control - ESDP (0x00000020)

This register is initialized only at LAN_PWR_GOOD, preserving the SDP states across software and PCIe resets. Some specific I/O pins are initialized in other resets in native mode as expected for the specific behavior, and described explicitly as follows:

Field	Bit(s)	Init.	Type	Description
SDP0_DATA	0	0b	RW	**SDP0 Data Value[1,2]** Used to read (write) a value of the software-controlled I/O pin SDP0. If SDP0 is configured as an output (*SDP0_IODIR* = 1b), this bit controls the value driven on the pin. If SDP0 is configured as an input, all reads return the current value of the pin.
SDP1_DATA	1	0b	RW	**SDP1 Data Value[1,3]** Used to read (write) a value of the software-controlled I/O pin SDP1. If SDP1 is configured as an output (*SDP1_IODIR* = 1b), this bit controls the value driven on the pin. If SDP1 is configured as an input, all reads return the current value of the pin.
SDP2_DATA	2	0b	RW	**SDP2 Data Value[1]** Used to read (write) a value of software-controlled I/O pin SDP2. If SDP2 is configured as an output (*SDP2_IODIR* = 1b), this bit controls the value driven on the pin. If SDP2 is configured as an input, all reads return the current value of the pin.
SDP3_DATA	3	0b	RW	**SDP3 Data Value[1]** Used to read (write) a value of the software-controlled I/O pin SDP3. If SDP3 is configured as an output (*SDP3_IODIR* = 1b), this bit controls the value driven on the pin. If SDP3 is configured as an input, all reads return the current value of the pin.
RESERVED	7:4	0x0	RSV	Reserved.
SDP0_IODIR	8	0b	RW	**SDP0 Pin Directionality[1,2]** Controls whether software-controlled pin SDP0 is configured as an input or output. 0b = Input 1b = Output
SDP1_IODIR	9	0b	RW	**SDP1 Pin Directionality[1,3]** Controls whether software-controlled pin SDP1 is configured as an input or output. 0b = Input 1b = Output

Field	Bit(s)	Init.	Type	Description
SDP2_IODIR	10	0b	RW	**SDP2 Pin Directionality**[1] Controls whether software-controlled pin SDP2 is configured as an input or output. 0b = Input 1b = Output
SDP3_IODIR	11	0b	RW	**SDP3 Pin Directionality**[1] Controls whether software-controlled pin SDP3 is configured as an input or output. 0b = Input 1b = Output
RESERVED	15:12	0x0	RSV	Reserved.
SDP0_NATIVE	16	0b	RW	**SDP0 Operating Mode**[2] 0b = Generic software controlled I/O by *SPD0_DATA* and *SDP0_IODIR*. 1b = Native mode operation (connected to hardware function) is IEEE 1588 functionality.
SDP1_NATIVE	17	0b	RW	**SDP1 Operating Mode**[1,3] 0b = Generic software controlled I/O by *SPD1_DATA* and *SDP1_IODIR*. 1b = Native mode operation (connected to hardware function) is IEEE 1588 functionality or thermal sensor mode according to the *SDP1_Function* bit.
SDP2_NATIVE	18	0b	RW	**SDP2 Operating Mode.** 0b = Generic software controlled I/O by *SPD2_DATA* and *SDP2_IODIR*. 1b = Native mode operation (connected to hardware function). 1588 functionality or I^2C functionality according to the *SDP23_FUNCTION* bit.
SDP3_NATIVE	19	0b	RW	**SDP3 Operating Mode.** 0b = Generic software controlled I/O by *SPD3_DATA* and *SDP3_IODIR*. 1b = Native mode operation (connected to hardware function). 1588 functionality or I^2C functionality according to the *SDP23_FUNCTION* bit.
RESERVED	24:20	0x0	RSV	Reserved.
SDP1_FUNCTION	25	0b	RW	**SDP1 Function**[3] SDP1 native mode functionality (*SDP1_NATIVE* = 1b). 0b = 1588 functionality. *SDP1_IODIR* should be configured as an output. 1b = SDP1 Thermal sensor functionality. The SDP1 pin drives the active high thermal sensor level signal that indicates that junction temperature has risen above a preset limit. *SDP1_IODIR* should be configured as an output.
SDP23_FUNCTION	26	0b	RW	**SDP23 Function** Defines the usage of SDP2 & SDP3 when *SDP[23]_NATIVE* is set to 1b. 0b = Use for 1588 functionality. 1b = Use for I^2C functionality. If this bit is set to 1b, *SDP3_NATIVE* should be equal to *SDP2_NATIVE*.
RESERVED	31:27	0x0	RSV	Reserved.

1. Initial values are read from the NVM when direction of SDP is Output (SDPx_IODIR = 1).
2. It is assumed that bit 15 of *NC-SI Configuration 2* word in the NVM is cleared. Otherwise, the SDP0 pin of function 0 is used as input pins that encode the NC-SI Package ID of the controller.
3. It is assumed that bit *SDP_FUNC_OFF_EN* of *NVM Control word 2* is cleared. Otherwise, SDP1 pins are strapped during PE_RST_N to determine that both PCIe functions are disabled.

8.2.2.1.5 PHY GPIO Register - PHY_GPIO (0x00000028)

Field	Bit(s)	Init.	Type	Description
PHY_GPIO	3:0	0x0	RO	**PHY-to-MAC GPIO** Used to read the four internal PHY to MAC general purpose signals.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.1.6 MAC GPIO Register - MAC_GPIO (0x00000030)

Field	Bit(s)	Init.	Type	Description
MAC_GPIO	3:0	0x0	RW	**MAC-to-PHY GPIO** Used to set the four internal MAC-to-PHY general purpose signals.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.1.7 PHY Interrupt Status Register 0 - PHYINT_STATUS0 (0x00000100)

Register contents is valid once the corresponding EEMNGCTL.*CFG_DONE0/1* bit is asserted by firmware. Bits are set by firmware to notify the host of the PHY interrupts that were triggered. This register is reset by hardware only at power-up events. The host is responsible to clear the PHY interrupts once it completes the PHY interrupt handling routine.

Field	Bit(s)	Init.	Type	Description
PMA_RECEIVE_LINK_STATUS	0	0b	RW	Reflects the inverse state of PHY register bit 1.1.2 before it was set by a firmware read.
PMA_TRANSMIT_FAULT	1	0b	RW	Reflects the state of PHY register bit 1.8.B before it was cleared by a firmware read.
PMA_RECEIVE_FAULT	2	0b	RW	Reflects the state of PHY register bit 1.8.A before it was cleared by a firmware read.
PMA_RESERVED	7:3	0x0	RW	Reserved. Read as written.
PCS_RECEIVE_LINK_STATUS	8	0b	RW	Reflects the inverse state of PHY register bit 3.1.2 before it was set by a firmware read.
PCS_TRANSMIT_FAULT	9	0b	RW	Reflects the state of PHY register bit 3.8.B before it was cleared by a firmware read.
PCS_RECEIVE_FAULT	10	0b	RW	Reflects the state of PHY register bit 3.8.A before it was cleared by a firmware read.
PCS_10GBASE_T_BLOCK_LOCK_LATCHED	11	0b	RW	Reflects the inverse state of PHY register bit 3.21.F before it was set by a firmware read.
PCS_10GBASE_T_HIGH_BER_LATCHED	12	0b	RW	Reflects the state of PHY register bit 3.21.E before it was cleared by a firmware read.
PCS_CRC_ERROR	13	0b	RW	Reflects the state of PHY register bit 3.EC00.F before it was cleared by firmware read.
PCS_LDPC_DECODE_FAILURE	14	0b	RW	Reflects the state of PHY register bit 3.EC00.E before it was cleared by a firmware read.
PCS_INVALID_65B_BLOCK	15	0b	RW	Reflects the state of PHY register bit 3.EC00.8 before it was cleared by a firmware read.
PCS_CHANGE_IN_AUXILIARY_BIT	16	0b	RW	Reflects the state of PHY register bit 3.EC00.0 before it was cleared by a firmware read.
PCS_RESERVED	23:17	0x0	RW	Reserved. Read as written.
PHY_XS_RESERVED	31:24	0x0	RW	Reserved. Read as written.

8.2.2.1.8 PHY Interrupt Status Register 1 - PHYINT_STATUS1 (0x00000104)

Register contents is valid once the corresponding EEMNGCTL.*CFG_DONE0/1* bit is asserted by firmware. Bits are set by firmware to notify the host of the PHY interrupts that were triggered. This register is reset by hardware only at power-up events. The host is responsible to clear the PHY interrupts once it completes the PHY interrupt handling routine.

Field	Bit(s)	Init.	Type	Description
AUTO_NEGOTIATION_EXTENDED_NEXT_PAGE_RECEIVED	0	0b	RW	Reflects the state of PHY register bit 7.1.6 before it was cleared by a firmware read.
AUTO_NEGOTIATION_REMOTE_FAULT	1	0b	RW	Reflects the state of PHY register bit 7.1.4 before it was cleared by a firmware read.
AUTO_NEGOTIATION_LINK_STATUS	2	0b	RW	Reflects the inverse state of PHY register bit 7.1.2 before it was cleared by a firmware read.
AUTO_NEGOTIATION_MASTER_SLAVE_CONFIGURATION_FAULT	3	0b	RW	Reflects the state of PHY register bit 7.21.F before it was cleared by a firmware read.
AUTO_NEGOTIATION_COMPLETED_FOR_NON_SUPPORTED_RATE	4	0b	RW	Reflects the state of PHY register bit 7.CC00.3 before it was cleared by a firmware read.
AUTO_NEGOTIATION_COMPLETED_FOR_SUPPORTED_RATE	5	0b	RW	Reflects the state of PHY register bit 7.CC00.2 before it was cleared by a firmware read.
AUTO_NEGOTIATION_AUTOMATIC_DOWNSHIFT	6	0b	RW	Reflects the state of PHY register bit 7.CC00.1 before it was cleared by a firmware read.
AUTO_NEGOTIATION_CONNECTION_STATE_CHANGE	7	0b	RW	Reflects the state of PHY register bit 7.CC00.0 before it was cleared by a firmware read.
AUTO_NEGOTIATION_100BASE_TX_DEVICE_DETECT	8	0b	RW	Reflects the state of PHY register bit 7.EC01.F before it was cleared by a firmware read.
AUTO_NEGOTIATION_ENERGY_ON_LINE_DETECT	9	0b	RW	Reflects the state of PHY register bit 7.EC01.E before it was cleared by a firmware read.
AUTO_NEGOTIATION_NEXT_PAGE_3RD_RECEIVED	10	0b	RW	Reflects the state of PHY register bit 7.EC01.3 before it was cleared by a firmware read.
AUTO_NEGOTIATION_NEXT_PAGE_2ND_RECEIVED	11	0b	RW	Reflects the state of PHY register bit 7.EC01.2 before it was cleared by a firmware read.
AUTO_NEGOTIATION_NEXT_PAGE_1ST_RECEIVED	12	0b	RW	Reflects the state of PHY register bit 7.EC01.1 before it was cleared by a firmware read.
AUTO_NEGOTIATION_BASE_PAGE_RECEIVED	13	0b	RW	Reflects the state of PHY register bit 7.EC01.0 before it was cleared by a firmware read.
AUTO_NEGOTIATION_10BASE_T_DEVICE_DETECT	14	0b	RW	Reflects the inverse state of PHY register bit 7.EC02.2 before it was cleared by a firmware read.
AUTO_NEGOTIATION_PROTOCOL_ERROR	15	0b	RW	Reflects the state of PHY register bit 7.EC01.D before it was cleared by a firmware read.
FLP_IDLE_ERROR	16	0b	RW	Reflects the state of PHY register bit 7.EC01.C before it was cleared by a firmware read.

Field	Bit(s)	Init.	Type	Description
AUTO_NEGOTIATION_GBE_PHY_RESERVED	27:17	0x0	RW	Reserved. Read as written.
PCIE_SERDES_LOSS_OF_SIGNAL_3_0	31:28	0x0	RW	Reflects the state of PHY register bit 4.CC02.F:C before it was cleared by a firmware read.

8.2.2.1.9 PHY Interrupt Status Register 2 - PHYINT_STATUS2 (0x00000108)

Register contents is valid once the corresponding EEMNGCTL.*CFG_DONE0/1* bit is asserted by firmware. Bits are set by firmware to notify the host of the PHY interrupts that were triggered. This register is reset by hardware only at power-up events. The host is responsible to clear the PHY interrupts once it completes the PHY interrupt handling routine.

Field	Bit(s)	Init.	Type	Description
GLOBAL_MAC_RESET	0	0b	RW	Reflects the inverse state of PHY register bit 1E.1200.0 before it was cleared by a firmware read.
GLOBAL_MAC_LOW_POWER_LINK_UP_MODE	1	0b	RW	Reflects the state of PHY register bit 1E.1204.4 before it was cleared by a firmware read.
GLOBAL_MAC_PHY_DISABLE_MODE	2	0b	RW	Reflects the inverse state of PHY register bit 1E.1204.1 before it was cleared by a firmware read.
GLOBAL_MAC_LOW_POWER_MODE	3	0b	RW	Reflects the inverse state of PHY register bit 1E.1204.0 before it was cleared by a firmware read.
GLOBAL_MAC_SPI_GRANT	4	0b	RW	Reflects the state of PHY register bit 1E.1206.2 before it was cleared by a firmware read.
GLOBAL_MAC_SPI_CONTROL	5	0b	RW	Reflects the inverse state of PHY register bit 1E.1206.1 before it was cleared by a firmware read.
GLOBAL_HIGH_TEMPERATURE_FAILURE	6	0b	RW	Reflects the state of PHY register bit 1E.CC00.E before it was cleared by a firmware read.
GLOBAL_LOW_TEMPERATURE_FAILURE	7	0b	RW	Reflects the state of PHY register bit 1E.CC00.D before it was cleared by a firmware read.
GLOBAL_HIGH_TEMPERATURE_WARNING	8	0b	RW	Reflects the state of PHY register bit 1E.CC00.C before it was cleared by a firmware read.
GLOBAL_LOW_TEMPERATURE_WARNING	9	0b	RW	Reflects the state of PHY register bit 1E.CC00.B before it was cleared by a firmware read.
GLOBAL_RESET_COMPLETED	10	0b	RW	Reflects the state of PHY register bit 1E.CC00.6 before it was cleared by a firmware read.
GLOBAL_DEVICE_FAULT	11	0b	RW	Reflects the state of PHY register bit 1E.CC00.4 before it was cleared by a firmware read.
GLOBAL_PAIR_A_CHANGE_OF_STATUS	12	0b	RW	Reflects the state of PHY register bit 1E.CC00.3 before it was cleared by a firmware read.
GLOBAL_PAIR_B_CHANGE_OF_STATUS	13	0b	RW	Reflects the state of PHY register bit 1E.CC00.2 before it was cleared by a firmware read.
GLOBAL_PAIR_C_CHANGE_OF_STATUS	14	0b	RW	Reflects the state of PHY register bit 1E.CC00.1 before it was cleared by a firmware read.
GLOBAL_PAIR_D_CHANGE_OF_STATUS	15	0b	RW	Reflects the state of PHY register bit 1E.CC00.0 before it was cleared by a firmware read.
GLOBAL_MEDIUM_BER	16	0b	RW	Reflects the state of PHY register bit 1E.CC01.E before it was cleared by a firmware read.
GLOBAL_RESERVED	31:17	0x0	RW	Reserved. Read as written.

8.2.2.1.10 LED Control - LEDCTL (0x00000200)

Note: All init bits in this register are read from the NVM. See Section 6.2.5.5 and Section 6.2.5.6.

Field	Bit(s)	Init.	Type	Description
LED0_MODE	3:0	0x0	RW	**LED0 Mode** This field specifies the control source for the LED0 output. An initial value of 0000b selects the LINK_UP indication.
RESERVED	4	0b	RSV	Reserved.
GLOBAL_BLINK_MODE	5	0b	RW	**Global Blink Mode** This bit specifies the blink mode of all LEDs. 0b = Blink at 200 ms on and 200 ms off. 1b = Blink at 83 ms on and 83 ms off.
LED0_IVRT	6	0b	RW	**LED0 Invert** This bit specifies the polarity/inversion of the LED0 source prior to output or blink control. By default the output drives the cathode of the LED0 so when the LED output is 0b the LED0 is on. 0b = LED0 output is active low. 1b = LED0 output is active high.
LED0_BLINK	7	0b	RW	**LED0 Blink** This bit specifies whether to apply blink logic to the (inverted) LED0 control source prior to the LED0 output. 0b = Do not blink LED0 output. 1b = Blink LED0 output.
LED1_MODE	11:8	0x1	RW	**LED1 Mode** This field specifies the control source for the LED1 output. An initial value of 0001b selects the 10 Gb/s link indication.
RESERVED	13:12	00b	RSV	Reserved.
LED1_IVRT	14	0b	RW	**LED1 Invert** This bit specifies the polarity/inversion of the LED1 source prior to output or blink control. By default the output drives the cathode of the LED so when the LED1 output is 0b the LED1 is on. 0b = LED1 output is active low. 1b = LED output is active high.
LED1_BLINK	15	1b	RW	**LED1 Blink** This bit specifies whether to apply blink logic to the (inverted) LED1 control source prior to the LED output. 0b = Do not blink LED output. 1b = Blink LED output.
LED2_MODE	19:16	0x4	RW	**LED2 Mode** This field specifies the control source for the LED2 output. An initial value of 0100b selects LINK/ACTIVITY indication.
RESERVED	21:20	00b	RSV	Reserved.
LED2_IVRT	22	0b	RW	**LED2 Invert** This bit specifies the polarity/inversion of the LED2 source prior to output or blink control. By default the output drives the cathode of the LED2 so when the LED2 output is 0b the LED2 is on. 0b = LED2 output is active low. 1b = LED2 output is active high.
LED2_BLINK	23	0b	RW	**LED2 Blink** This bit specifies whether to apply blink logic to the (inverted) LED2 control source prior to the LED2 output. 0b = Do not blink LED2 output. 1b = Blink LED2 output.
LED3_MODE	27:24	0x5	RW	**LED3 Mode** This field specifies the control source for the LED3 output. An initial value of 0101b selects the 1 Gb/s link indication.

Field	Bit(s)	Init.	Type	Description
RESERVED	29:28	00b	RSV	Reserved.
LED3_IVRT	30	0b	RW	**LED3 Invert** This bit specifies the polarity/inversion of the LED3 source prior to output or blink control. By default the output drives the cathode of the LED3 so when the LED3 output is 0b the LED3 is on. 0b = LED3 output is active low. 1b = LED3 output is active high.
LED3_BLINK	31	0b	RW	**LED3 Blink** This bit specifies whether to apply blink logic to the (inverted) LED3 control source prior to the LED3 output. 0b = Do not blink LED3 output. 1b = Blink LED3 output.

8.2.2.1.11 Extended VLAN Ether Type - Receive - EXVET (0x00005078)

Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0x0	RSV	Reserved.
VET_EXT	31:16	0x8100	RW	**Outer VLAN Ether Type** The VLAN Tag Protocol Identifier (TPID). ***Note:*** This field appears in little endian (MS byte first on the wire).

8.2.2.1.12 Extended VLAN Ether Type - Transmit - EXVET_T (0x00008224)

Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0x0	RSV	Reserved.
VET_EXT	31:16	0x8100	RW	**Outer VLAN Ether Type** The VLAN Tag Protocol Identifier (TPID). ***Note:*** This field appears in little endian (MS byte first on the wire).

8.2.2.1.13 Function Active and Power State to Manageability - FACTPS (0x00010150)

Register for use by the device firmware for configuration.

Note: In regular mode (port swap disabled - *LAN_FUNCTION_SEL* = 0b), the power state indication of port 0 is mapped to the *FUNC0_POWER_STATE* field, and the power state indication of port 1 is mapped the *FUNC1_POWER_STATE* field. Vice-versa when port swap mode is enabled (*LAN_FUNCTION_SEL* = 1b).

Field	Bit(s)	Init.	Type	Description
FUNC0_POWER_STATE	1:0	00b	RO	**Function 0 Power State** Power state indication of function 0. 00b = DR 01b = D0u 10b = D0a 11b = D3

Field	Bit(s)	Init.	Type	Description
LAN0_VALID	2	0b	RO	**LAN 0 Enable** This status of this bit reflects whether the LAN 0 function is disabled through the external pad. 0b = Disabled 1b = Enabled
FUNC0_AUX_EN	3	0b	RO	**Function 0 Auxiliary (AUX) Power PM Enable** Shadow from the configuration space.
RESERVED	5:4	00b	RSV	Reserved.
FUNC1_POWER_STATE	7:6	00b	RO	**Function 1 Power State** Power state indication of function 1. 00b = DR 01b = D0u 10b = D0a 11b = D3
LAN1_VALID	8	0b	RO	**LAN 1 Enable** This status of this bit reflects whether the LAN 1 function is disabled through the external pad. 0b = Disabled 1b = Enabled
FUNC1_AUX_EN	9	0b	RO	**Function 1 Auxiliary (AUX) Power PM Enable** Shadow from the configuration space.
RESERVED	29:10	0x0	RSV	Reserved.
LAN_FUNCTION_SEL	30	0b	RO	**LAN Function Select** When both LAN ports are enabled and *LAN_FUNCTION_SEL* =: 0b = LAN 0 is routed to PCI function 0, and LAN 1 is routed to PCI function 1. 1b = LAN 0 is routed to PCI function 1 and LAN 1 is routed to PCI function 0. This bit is loaded from NVM.
PM_STATE_CHANGED	31	0b	RO	**PM State Changed** An indication that one or more of the functional power states had changed. This bit is also a signal to manageability to create an interrupt. This bit is cleared on read and is not set for at least eight cycles after it was cleared.

8.2.2.1.14 General Receive Control - GRC (0x00010200)

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
APME	1	0b	RW	**Advance Power Management Enable** If set to 1b, manageability wake-up is enabled. The device sets the *PME_Status* bit in the Power Management Control/Status Register (PMCSR), asserts PE_WAKE_N when manageability wake-up is enabled, and when it receives a matching magic packet. It is a single read/write bit in a single register, but has two values depending on the function that accesses the register. The value of this bit is loaded from *NVM Control Word 3* (bit 0 for port 0 and bit 1 for port 1). The bit is loaded after each PCIe reset and at power on.
RESERVED	31:2	0x0	RSV	Reserved.

8.2.2.1.15 Device and Functions Enable Control - DEV_FUNC_EN (0x00010208)

This register is common to the two ports, and reflects the LAN disable and device disable/enable bits in the NVM.

Field	Bit(s)	Init.	Type	Description
LAN_PCI_DISABLE	0	0b	RO	**LAN PCI Disable** When set, one LAN port is disabled. The function that is disabled is determined by the *LAN_DISABLE_SELECT* bit. If the disabled function is function 0, it acts as a dummy function or the other LAN function depending on the Dummy function enable setting. If the disabled port is used for WoL or by MC, only the DMA block of the port is powered down. Otherwise, the port is powered down including the PHY. If the *LAN PCI Disable* bit is set for a port, the port's respective *APM* bit in *NVM Control Word 3* must be cleared as well.
LAN_DISABLE_SELECT	1	0b	RO	**LAN Disable Select** 0b = LAN 0 is disabled. 1b = LAN 1 is disabled.
DEV_OFF_EN	2	0b	RO	**Device Electrical Off Enable** This bit is relevant only when the device is disabled via strapping during PE_RST_N both LANn_DIS_N pins to 0b at once. 0b = Legacy mode (default). Though the device is disabled, the digital I/O pins are not moved to an electrical off state. 1b = Enable device electrical off. When the device is disabled, the digital I/O pins are put at High-Z. For example, electrical off state where pull-ups/downs are at their defined values.
SDP_FUNC_OFF_EN	3	0b	RO	**PCIe Function Off via SDP Pins Enable** 0b = Legacy mode (default), SDPn_1 pins does not control PCIe functions off. 1b = Two SDP pins are used in conjunction by strapping (sampled by PE_RST_N) to disable the two PCIe functions altogether. ***Note:*** If MNG is present, MC-to-LAN paths are not disabled. PCIe Function Off pins can also work independently, where each pin controls the corresponding PCIe function. However independent mode is not supported in this device.
IGNORE_PHY_FW_VALID	4	0b	RO	**Ignore PHY Firmware Valid** Reflects the *Ignore PHY Firmware valid indication* in *NVM Control Word 2*.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.2.1.16 SFP I^2C Command - I2CCMD (0x00015F58)

This register is used by software to read or write to the configuration registers in an SFP module when the *SDP23_FUNCTION* bit and the *SDP[23]_NATIVE* bit are set in ESDP register (Section 8.2.2.1.4).

Field	Bit(s)	Init.	Type	Description
DATA	15:0	X	RW	**Data** **Data in a write command:** Software places the data bits and then the MAC shifts them out to the I^2C bus. **Data in a read command:** The MAC reads these bits serially from the I^2C bus and then software reads them from this location. This field is interpreted as two consecutive bytes unless the I2CPARAMS.*I2C_DATA_ORDER* bit is set, in which case the data is considered as a single 16-bit word. When I2CPARAMS.*ACCESS_WIDTH* = 0b, bits 15:8 are not used.
REGADD	23:16	0x0	RW	**I^2C Register Address** For example, register 0, 1, 2... 255.
RESERVED	26:24	000b	RSV	Reserved

Field	Bit(s)	Init.	Type	Description
OP	27	0b	RW	**Op Code** 0b = I^2C write. 1b = I^2C read.
RESET	28	0b	RW	**Reset Sequence** If set, sends a reset sequence before the actual read or write. This bit is self clearing. A reset sequence is defined as nine consecutive stop conditions.
R	29	1b	RW	**Ready Bit** Set to 1b by the device at the end of the I^2C transaction. Indicates the read or write completed. Reset by a software write of a command.
RESERVED	30	0b	RSV	Reserved
E	31	0b	RW	**Error** This bit set is to 1b by hardware when it fails to complete an I^2C read. Reset by a software write of a command. ***Note:*** This bit is valid only when the Ready (*R*) bit is set.

8.2.2.1.17 SFP I^2C Parameters - I2CPARAMS (0x00015F5C)

This register is used to set the parameters for I^2C access and to allow bit-banging access to the I^2C interface.

Note: This register is reset only on LAN_PWR_GOOD.

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 milliseconds. ***Note:*** A value of zero 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 microseconds. ***Note:*** A value of zero is not valid
I2CBB_EN	8	0b	RW	**I^2C Bit-Bang Enable** If set, the I2C_CLK and I2C_DATA lines are controlled via the *CLK*, *DATA*, and *DATA_OE_N* bits of this register. Otherwise, they are controlled by hardware once activated via the I2CCMD register (Section 8.2.2.1.16).
CLK_OUT	9	0b	RW	**I^2C Clock** While in bit-bang mode, controls the value driven on the I2C_CLK pad of this port.
DATA_OUT	10	0b	RW	**I^2C_DATA** While in bit-bang mode and when the *DATA_OE_N* field is zero, controls the value driven on the I2C_DATA pad of this port.
DATA_OE_N	11	0b	RW	**I^2C Data Output Enable** While in bit-bang mode, controls the direction of the I2C_DATA pad of this port. 0b = Pad is output. 1b = Pad is input.
DATA_IN	12	0b	RO	**I^2C Data In** Reflects the value of the I2C_DATA pad. While in bit-bang mode when the *DATA_OE_N* field is zero, this field reflects the value set in the *DATA_OUT* field.
CLK_OE_N	13	0b	RW	**I^2C Clock Output Enable** While in bit-bang mode, controls the direction of the I2C_CLK pad of this port. 0b = Pad is output. 1b = Pad is input.

Field	Bit(s)	Init.	Type	Description
CLK_IN	14	0b	RO	**I²C Clock In Value** Reflects the value of the I2C_CLK pad. While in bit-bang mode when the *CLK_OE_N* field is zero, this field reflects the value set in the *CLK_OUT* field.
CLK_STRETCH_DIS	15	0b	RW	**Clock Stretch Disable** 0b = Enable slave clock stretching support in an I²C access. 1b = Disable clock stretching support in an I²C access.
ACCESS_WIDTH	16	0b	RW	**I²C Access Width** 0b = Byte access. 1b = Word access.
RESERVED	23:17	0x0	RSV	Reserved. Write 0x0, ignore on read.
PHYADD	30:24	0x0	RW	**Device Address Bits 7:1** The actual address used is b{*PHYADD*[6:0], 0}.
I2C_DATA_ORDER	31	0b	RW	**I²C Data Order** 0b = I2CCMD.*DATA* field read in byte order. 1b = I2CCMD.*DATA* field read in word order.

8.2.2.2 PF - NVM Registers

8.2.2.2.1 EEPROM Mode Control Register - EEC (0x00010010)

Field	Bit(s)	Init.	Type	Description
RESERVED	7:0	0x0	RSV	Reserved. Reads as 0b.
EE_PRES	8	0b	RO	**NVM Present** When this bit is set, indicates that an NVM is present and has the correct signature field. This bit is read-only.
AUTO_RD	9	0b	RO	**NVM Auto-Read Done** When set to 1b, this bit indicates that the auto-read by hardware from the NVM caused by the latest reset for this port is done. This bit is also set when the NVM is not present or when its signature field is not valid. This bit does not reflect the status of the PHY image auto-load process. Use *Reset Completed* (bit 6) in PHY register 1E.CC00 to get this indication.
MNG_READY	10	0b	RO	**Management Auto-Load Done** When set, indicates the sections needed for manageability are valid.
EE_SIZE	14:11	0x7	RO	**NVM Size Via EEPROM Mode** This field defines the size of the NVM that is accessible via EEPROM mode. This is equal to the size of the internal Shadow RAM, fixed to 16 KB (0x7 defines 16 KB - legacy).
PCI__ANA_DONE	15	0b	RO	**PCIe Analog Done** When set to 1b, indicates that the PCIe analog section read from NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid.
PCI_CORE_DONE	16	0b	RO	**PCIe Core Done** When set to 1b, indicates that the core analog section read from NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid. ***Note:*** This bit returns the relevant done indication for the function that reads the register.
PCI_GENERAL_DONE	17	0b	RO	**PCIe General Done** When set to 1b, indicates that the PCIe general section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid.
PCI_FUNC_DONE	18	0b	RO	**PCIe Function Done** When set to 1b, indicates that the PCIe function section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid. ***Note:*** This bit returns the relevant done indication for the function that reads the register.
CORE_DONE	19	0b	RO	**Core Done** When set to 1b, indicates that the core section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid. ***Note:*** This bit returns the relevant done indication for the function that reads the register.
CORE_CSR_DONE	20	0b	RO	**Core CSR Done** When set to 1b, indicates that the core CSR section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid. ***Note:*** This bit returns the relevant done indication for the function that reads the register.

Field	Bit(s)	Init.	Type	Description
MAC_DONE	21	0b	RO	**MAC Done** When set to 1b, indicates that the MAC section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid. ***Note:*** This bit returns the relevant done indication for the function that reads the register.
LCB_DONE	22	0b	RO	**LCB Done** When set, indicates that the LCB section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid.
RESERVED	24:23	00b	RSV	Reserved. Reads as 00b.
SEC1VAL	25	0b	RO	**Sector 1 Valid** Meaningful only when the *EE_PRES* bit is read as 1b. 0b = Indicates that the content of the first section (from byte address 0x0000 to 0x3FFF) is valid. 1b = Indicates that the content of the second Shadow RAM section (from byte address 0x4000 to 0x7FFF) of the Flash device is valid. This bit can be written by firmware
FLUDONE	26	1b	RO	**Flash Update Done** When set to 1b, indicates that the last Flash update process completed. This bit is set by firmware.
RESERVED	27	1b	RSV	Reserved.
RESERVED	31:28	0x0	RSV	Reserved. Reads as 0b.

8.2.2.2.2 Flash Access Register - FLA (0x0001001C)

Note: When NVM is locked, this register is RO to software.

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.
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.
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.
FL_SO	3	0b	ROS	**Data output bit from the Flash** The FL_SO input 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:*** Writes to this bit have no effect.
FL_REQ	4	0b	RW	**Request Flash Access** Software must write 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 0b.
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.
RESERVED	6	0b	RO	Reserved.
JTAG_DIS	7	0b	RO	**JTAG Disable** Reflects the value of the *JTAG_DIS* strap.

Field	Bit(s)	Init.	Type	Description
RESERVED	11:8	0x0	RSV	Reserved. Reads as 0b.
FL_SIZE	14:12	101b	RO	**Flash Size** Indicates Flash size of 64 KB * 2 ** "*FL_SIZE*". The Flash size impacts the requested memory space for the Flash and expansion ROM BARs in PCIe configuration space (according to the *CSRSIZE* and *FL_BAR_SIZE* fields in the PCI_LBARCTRL register - Section 8.2.2.5.6). Supported Flash sizes: 101b = 2 MB 110b = 4 MB This register limits the accesses to the actual flash size.
RESERVED	15	0b	RSV	Reserved. Reads as 0b.
FL_SADDR	28:16	0x0	RW	**Flash Sector Erase Address** Determines which 4 KB sector is erased when the *FL_SER* command is used. This address is expressed in sector index units, starting from sector index 0.
FL_SER	29	0b	RW	**Flash Sector Erase Command** This bit is auto-cleared and reads as 0b. The 4 KB sector index to be erased is determined by the *FL_SADDR* field. ***Note:*** The address should fit in the size defined in the FLA.*FL_SIZE* field.
FL_BUSY	30	0b	RO	**Flash Busy** This bit is set to 1b by hardware while Flash access has been granted to a client. ***Note:*** This bit is read-only from a software perspective. ***Note:*** This bit only reflect flash busy state if FLA mechanisms, like *FL_SER* or *FL_DER*, are used. If the flash is busy due to some action done directly on the SPI pins, it is not reflected in this bit.
FL_DER	31	0b	RW	**Flash Device Erase Command** This bit is auto-cleared and reads as 0b. The entire Flash device is erased.

8.2.2.2.3 Manageability EEPROM-Mode Control Register - EEMNGCTL (0x00010110)

This register is read/write to manageability firmware, and is read-only to host software. The transactions performed through this register are directed to/from the internal Shadow RAM.

Field	Bit(s)	Init.	Type	Description
ADDR	12:0	0x0	RW	**Address** This field is written by manageability along with the *START* and *WRITE* bits to indicate which NVM word address to read or write. Only the first valid 16 KB is accessible through this EEPROM mode interface dedicated to manageability.
RESERVED	14:13	00b	RSV	Reserved. Reads as 0b.
START	15	0b	RW	**Start** Writing a 1b to this bit causes the device to start the read or write operation of the Shadow RAM according to the write bit. This bit is self cleared by hardware. If one of the *CFG_DONE0/1* bit's value is set in this write, writing of this bit is ignored.
WRITE	16	0b	RW	**Write** This bit signals the device if the current operation is read or write. 0b = Read 1b = Write
EEBUSY	17	0b	RW	**EEPROM Mode Busy** This bit indicates that the internal Shadow RAM is busy and should not be accessed.

Field	Bit(s)	Init.	Type	Description
CFG_DONE0	18	0b	RW	**Configuration Done Port 0** Manageability configuration cycle of port 0 completed. This bit indicates that the manageability configuration cycle (configuration of PCIe, core or PHY) completed. This bit is set to 1b by manageability firmware to indicate that the configuration completed and is cleared by software. Writing a 0b by firmware does not affect the state of this bit. Writing a 1b by software does not affect the state of this bit. ***Note:*** Software should not try to access the PHY for configuration before this bit is set.
CFG_DONE1	19	0b	RW	**Configuration Done Port 1** Manageability configuration cycle of port 1 completed. This bit indicates that the manageability configuration cycle (configuration of PCIe, core or PHY) completed. This bit is set to 1b by manageability firmware to indicate that the configuration completed and is cleared by software. Writing a 0b by firmware does not affect the state of this bit. Writing a 1b by software does not affect the state of this bit. ***Note:*** Software should not try to access the PHY for configuration before this bit is set.
TRANS_ABORTED	20	0b	RO	**Transaction Aborted** When read as 1b, indicates that the current EEMNGCTL access was aborted due to a firmware reset. The bit is cleared by hardware on the next write access to the EEMNGCTL register, regardless of the written value.
RESERVED	30:21	0x0	RSV	Reserved.
DONE	31	1b	RW	**Transaction Done** This bit is cleared when the *START* and *WRITE* bits are set by manageability, and is set back again when the Shadow RAM write or read transaction completes.

8.2.2.2.4 Manageability EEPROM-Mode Read/Write Data - EEMNGDATA (0x00010114)

This register is read/write to manageability firmware, and is read-only to host software.

Field	Bit(s)	Init.	Type	Description
WRDATA	15:0	0x0	RW	**Write Data** Data to be written to the Shadow RAM.
RDDATA	31:16	X	RW	**Read Data** Data returned from the read command. ***Note:*** This field is read only.

8.2.2.2.5 Manageability Flash Control Register - FLMNGCTL (0x00010118)

This register is read/write to manageability firmware, and is read-only to host software.

Field	Bit(s)	Init.	Type	Description
ADDR	23:0	0x0	RW	**Address** This field is written by manageability along with *CMD* and *CMDV* to indicate which Flash address to read or write. For the Sector Erase command (*CMD* = 10b), it must point to any address in the 4 KB sector to be erased.

Field	Bit(s)	Init.	Type	Description
CMD	25:24	00b	RW	**Command** Indicates which command should be executed. Valid only when the *CMDV* bit is set. 00b = Read 01b = Write (single byte) 10b = Sector Erase — The ADDR field determines the 4 KB sector index to be erased by this command. 11b = Device Erase
CMDV	26	0b	RW	**Command Valid** When set, indicates that the manageability firmware issues a new command and is cleared by hardware at the end of the command.
FLBUSY	27	0b	RW	**Flash Busy** This bit indicates that the Flash is busy processing a Flash transaction and should not be accessed.
RESERVED	29:28	00b	RSV	Reserved.
DATA_VALID	30	0b	RW	**Read Done** Read done and data in FLMNGDATA is valid. This bit is cleared by firmware when it sets the *CMDV* bit. It is set by hardware for each DWord read that completes. This bit is read/clear by hardware enabling the multiple DWord read flow.
GLDONE	31	1b	RW	**Global Done** This bit clears after the *CMDV* bit is set by manageability, and is set back again when all Flash transactions complete. For example, the Flash device finished reading all the requested read or other accesses (write and erase).

8.2.2.2.6 Manageability Flash Read/Write Data - FLMNGDATA (0x0001011C)

This register is read/write to manageability firmware, and is read-only to host software.

Field	Bit(s)	Init.	Type	Description
DATA	31:0	0x0	RW	**Data** Read/Write data on a read transaction. This register contains the data returned from the Flash read. On write transactions, bits 7:0 are written to the Flash.

8.2.2.2.7 JEDEC ID 1 - JEDEC_ID_1 (0x00015F2C)

Exposes the JEDEC ID of the connected flash. Reflects an internal auxiliary register.

Note: This register is valid only if a valid firmware is present.

Field	Bit(s)	Init.	Type	Description
BANK	6:0	0x0	ROS	**Bank** Defines the bank number of the manufacturer ID (number of 0x7F read).
VALID	7	0b	ROS	**Valid** When this bit is set, the firmware stored a valid content to this register.
MANUFACTURER_ID	15:8	0x0	ROS	**Manufacturer ID** Returns the manufacturer ID read by RDID command.
DEVICE_ID_1	23:16	0x0	ROS	**Device ID 1** First byte of the device ID read by RDID command (*Family* and *Density* fields).
DEVICE_ID_2	31:24	0x0	ROS	**Device ID 2** Second Byte of the device ID read by RDID command (Sub version and Revision).

8.2.2.3 PF - Flow Control Registers

8.2.2.3.1 Flow Control Transmit Timer Value n - FCTTVN[n] (0x00003200 + 0x4*n, n=0...3)

Each 32-bit register (n=0...3) refers to two timer values (register 0 refers to timer 0 and 1; register 1 refers to timer 2 and 3, etc.).

Note: The 16-bit value in the *TTV* field is inserted into a transmitted frame (either XOFF frames or any pause frame value in any software transmitted packets). It counts in units of slot time (usually 64 bytes). The device uses a fixed slot time value of 64-byte times.

Field	Bit(s)	Init.	Type	Description
TTV_2N	15:0	0x0	RW	**Transmit Timer Value 2n** Timer value included in XOFF frames as Timer (2n). The same value must be set to User Priorities (UPs) attached to the same TC, as defined in the RTTUP2TC register. For legacy 802.3X flow control packets, TTV0 is the only timer that is used.
TTV_2N_1	31:16	0x0	RW	**Transmit Timer Value 2n+1** Timer value included in XOFF frames as Timer 2n+1. The same value must be set to UPs attached to the same TC, as defined in the RTTUP2TC register.

8.2.2.3.2 Flow Control Receive Threshold Low - FCRTL[n] (0x00003220 + 0x4*n, n=0...7)

This register contains the receive threshold used to determine when to send an XON packet, and counts in units of bytes. The lower four bits must be programmed to 0x0 (16-byte granularity). Software must set *XONE* to enable the transmission of XON frames. Each time incoming packets cross the receive high threshold (become more full), and then crosses the receive low threshold, with *XONE* enabled (1b), hardware transmits an XON frame.

Note: Each 32-bit register (n=0...7) refers to a different receive packet buffer.

Field	Bit(s)	Init.	Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
RTL	18:5	0x0	RW	**Receive Threshold Low n** Receive packet buffer n FIFO low water mark for flow control transmission (32-byte granularity).
RESERVED	30:19	0x0	RSV	Reserved.
XONE	31	0b	RW	**XON Enable n** Per the receive packet buffer XON enable. 0b = Disabled 1b = Enabled

8.2.2.3.3 Flow Control Receive Threshold High - FCRTH[n] (0x00003260 + 0x4*n, n=0...7)

This register contains the receive threshold used to determine when to send an XOFF packet and counts in units of bytes. This value must be at least eight bytes less than the maximum number of bytes allocated to the receive packet buffer and the lower five bits must be programmed to 0x0 (32-byte granularity). Each time the receive FIFO reaches the fullness indicated by *RTH*, hardware transmits a pause frame if the transmission of flow control frames is enabled.

Note: Each 32-bit register (n=0... 7) refers to a different receive packet buffer.

Field	Bit(s)	Init.	Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
RTH	18:5	0x0	RW	**Receive Threshold High n** Receive packet buffer n FIFO high water mark for flow control transmission (32-byte granularity).
RESERVED	30:19	0x0	RSV	Reserved.
FCEN	31	0b	RW	**Flow Control Enable** Transmit Flow control enable for packet buffer n. Should be set only for traffic classes to which UPs are mapped (in RTRUP2TC register).

8.2.2.3.4 Flow Control Refresh Threshold Value - FCRTV (0x000032A0)

Field	Bit(s)	Init.	Type	Description
FC_REFRESH_TH	15:0	0x0	RW	**Flow Control Refresh Threshold** This value is used to calculate the actual refresh period for sending the next pause frame if conditions for a pause state are still valid (buffer fullness above low threshold value). The formula for the refresh period for priority group N is: `FCTTV[N/2].TTV[Nmod2] - FCRTV.FC_REFRESH_TH` ***Note:*** The *FC_REFRESH_TH* must be smaller than TTV of the TC and larger than the maximum packet size in the TC + FC packet size + link latency and Tx latency and Rx latency in 64-byte units.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.3.5 Flow Control Configuration - FCCFG (0x00003D00)

Field	Bit(s)	Init.	Type	Description
RESERVED	2:0	000b	RSV	Reserved.
TFCE	4:3	00b	RW	**Transmit Flow Control Enable** These bits indicate that the device transmits flow control packets (XON/XOFF frames) based on receive fullness. If auto-negotiation is enabled, this bit should be set by software to the negotiated flow control value. 00b = Transmit flow control disabled. 01b = Link flow control enabled. 10b = Priority flow control enabled. 11b = Reserved. ***Note:*** Priority flow control should be enabled in DCB mode only.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.2.3.6 MAC Flow Control Register - MFLCN (0x00004294)

Field	Bit(s)	Init.	Type	Description
PMCF	0	0b	RW	**Pass MAC Control Frames** Filter out unrecognized pause (flow control opcode 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 only when RFCE or RPFCM are set to 1b. If both *RFCE* and *RPFCM* are set to 0b, this bit has no effect on incoming PAUSE frames.
RPFCM	2	0b	RW	**Receive Priority Flow Control Mode** Indicates that the X550 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 must be set as a logical "OR" over the *RPFCE*[7:0] bitmap. It is useful to control forwarding of PFC frames to host if required. ***Notes:*** 1. Priority Flow control should be enabled in DCB mode only. 2. 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. 3. This bit should not be set if bit 3 (*RFCE*) is set.
RFCE	3	0b	RW	**Receive Link Flow Control Enable** Indicates that the X550 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 indication received for UP#n is ignored and transmit from UP#n is not paused. 1b = The device responds to the reception of Priority Flow Control packets for UP#n.
RESERVED	31:12	0x0	RSV	Reserved.

8.2.2.3.7 Priority Flow Control Type Opcode - PFCTOP (0x0000431C)

This register contains the *Type* and *Opcode* fields that are matched against a recognized priority flow control packet.

Field	Bit(s)	Init.	Type	Description
FCT	15:0	0x8808	RW	**Priority Flow Control EtherType** This field appears in little endian (MS byte first on the wire).
FCOP	31:16	0x0101	RW	**Priority Flow Control Opcode** This field appears in big endian (LS byte first on the wire).

8.2.2.3.8 Transmit Flow Control Status - TFCS (0x0000CE00)

Field	Bit(s)	Init.	Type	Description
TC_XON	7:0	0xFF	RO	**TC XON** TC is in FC XON state.
RESERVED	31:8	0x0	RSV	Reserved.

8.2.2.4 PF - PCIe Registers

This section contains the registers used to control the PCIe core behavior.

8.2.2.4.1 PCIe Function Status 1 - PCI_STATUS1 (0x00011028)

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.

8.2.2.4.2 PCIe Firmware Control - PCI_FWCTRL (0x00011040)

Field	Bit(s)	Init.	Type	Description
TCO_ISOLATE	0	0b	RW	**TCO Isolate** Set to 1b on a TCO Reset SMBus or NC-SI OEM command, when the *TCO_ISOLATE* bit is set. Host write cycles are completed successfully on the PCIe, but silently ignored by internal logic. ***Note:*** When firmware initiates the TCO Isolate command, it also writes a value of 0x03 to the FWSM.*EXT_ERR_IND* field. This bit is RO and mirrors the value of the *Isolate* bit in the internal management registers. ***Note:*** 1. Bit is reset by LAN_PWR_GOOD and firmware reset only. 2. Bit reflects internal management register *Aux* bit.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.4.3 PCIe CSR Access Timeout - PCI_CSRTO (0x00011044)

Field	Bit(s)	Init.	Type	Description
CSR_TO	15:0	0x000F	RW	**CSR Timeout** Defines the timeout value in micro second for CSR accesses
MSIX_TO	31:16	0x0028	RW	**MSI-X Timeout** Defines the timeout value in micro second for MSI-X table accesses

8.2.2.4.4 PCIe Control Extended Register - GCR_EXT (0x00011050)

Field	Bit(s)	Init.	Type	Description
VT_MODE	1:0	00b	RW	**VT Mode of operation** Defines the allocation of physical registers to the VFs. Software must set this field the same as GPIE. VT_Mode: 00b = noVT — Reserved for the case that STATUS.*IOV_ENA* is not set. 01b = VT16 — Resources are allocated to 16 VFs. 10b = VT32 — Resources are allocated to 32 VFs 11b = VT64 — Resources are allocated to 64 VFs.
RESERVED	3:2	00b	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
APBACD	4	0b	RW	**Auto PBA Clear Disable** 0b = Any active PBA entry is cleared on the falling edge of the appropriate interrupt request to the PCIe block. 1b = Software can clear PBA only by direct write to clear access to the PBA bit. The appropriate interrupt request is cleared when software sets the associated interrupt mask bit in the EIMS (re-enabling the interrupt) or by direct write to clear the PBA.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.2.4.5 PCI Flash Access Timeout - PCI_FLASHTO (0x00011054)

Field	Bit(s)	Init.	Type	Description
PCI_FLASHTO	31:0	0x00000FA0	RW	**PCI Flash Timeout** Defines the timeout value in microseconds for flash accesses.

8.2.2.4.6 Function Requester ID Information Register - FUNC_RID (0x00011070)

Field	Bit(s)	Init.	Type	Description
FUNCTION_NUMBER	2:0	000b	RO	**Function Number** Function number assigned to the function based on BIOS/OS Enumeration.
DEVICE_NUMBER	7:3	0x0	RO	**Device Number** Device number assigned to the function based on BIOS/OS Enumeration.
BUS_NUMBER	15:8	0x0	RO	**Bus Number** Bus Number assigned to the function based on BIOS/OS Enumeration.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.4.7 PCIe Revision ID - PCI_REVID (0x00011098)

Field	Bit(s)	Init.	Type	Description
NVM_REVID	7:0	0x0	RW	**NVM Revision ID** Value of Rev ID loaded from NVM. This value is XORed with the hardware default to create the value reflected in the config space.
RESERVED	31:8	0x0	RSV	Reserved.

8.2.2.4.8 PCIe Errors Reported - PCI_PCIERR (0x00011140)

This register indicates which PCIe errors are reported to device software.

Field	Bit(s)	Init.	Type	Description
PCIE_ERR_REP	31:0	0x0	RO	**PCIe Errors Reported** Each bit corresponds to a particular error event as described in Section 3.1.5.8, "Proprietary Error Reporting".

8.2.2.4.9 PCIe Interrupt Cause - PCI_ICAUSE (0x00011520)

Field	Bit(s)	Init.	Type	Description
PCIE_ERR_CAUSE	31:0	0x0	RW1C	Each bit corresponds to a particular error event as described in Section 3.1.5.8, "Proprietary Error Reporting".

8.2.2.4.10 PCIe Interrupts Enable - PCI_IENA (0x00011528)

Field	Bit(s)	Init.	Type	Description
PCIE_ERR_EN	31:0	0x0	RW	Each bit corresponds to a particular error event as described in Section 3.1.5.8, "Proprietary Error Reporting".

8.2.2.4.11 PCIe VM Pending Index - PCI_VMINDEX (0x00011530)

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 PCI_VMPEND register (Section 8.2.2.4.12). The VM index is an absolute index in the range of 0 through 63. It can be set by the software only to VMs that the PF owns and only to VMs which are not assigned to a VF.
RESERVED	31:9	0x0	RSV	Reserved.

8.2.2.4.12 PCIe VM Pending Status - PCI_VMPEND (0x00011538)

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 PCI_VMINDEX register (Section 8.2.2.4.11). This flag is set to 1b as long as there is at least one PCIe transaction, pending for its completion.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.4.13 PCIe Default Revision ID - PCI_DREVID (0x00011540)

Field	Bit(s)	Init.	Type	Description
DEFAULT_REVID	7:0	0x1	RO	**Default Revision ID** Mirroring of Default Rev ID prior to NVM load. 0x00 = A0 0x01 = B0
RESERVED	31:8	0x0	RSV	Reserved.

8.2.2.4.14 PCIe Byte Counter High - PCI_BYTCTH (0x00011544)

A byte counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_BW_BCT	31:0	0x0	RO	**PCIe Byte Counter High** This register contains the high double-word of a 64-bit counter that counts PCIe payload bytes. This register gets stuck at its maximum value of 0xFF...F.

8.2.2.4.15 PCIe Byte Counter Low - PCI_BYTCTL (0x00011548)

A byte counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_BW_BCT	31:0	0x0	RO	**PCIe Byte Counter Low** This register contains the low double-word of a 64-bit counter that counts PCIe payload bytes. This register gets stuck at its maximum value of 0xFF...F.

8.2.2.4.16 PCIe LCB Data Port - PCI_LCBDATA (0x00011734)

Field	Bit(s)	Init.	Type	Description
LCB_DATA	31:0	0x0	RW	**LCB Data** A 32-bit data register (part of a port) used to load NVM configuration into the PCIe LCB unit. Operates together with the PCI_LCBADD register (Section 8.2.2.4.17).

8.2.2.4.17 PCIe LCB Address Port - PCI_LCBADD (0x00011788)

Field	Bit(s)	Init.	Type	Description
ADDRESS	17:0	0x0	RW	**Address** An 18-bit address register (part of a port) used to load NVM configuration into the PCIe LCB unit. Operates together with the PCI_LCBDATA register (Section 8.2.2.4.16).
RESERVED	19:18	00b	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
BLOCK_ID	27:20	0x0	RW	**Block ID** An ID of a sub-unit in the PCIe unit. Supported values are: 0 = NPQ: RLAN 1 = NPQ: TLAN 2 = NPQ: RX_PE 3 = NPQ: TX_PE 4 = NPQ: PMAT 5 = NPQ: MNG 6 = NPQ: TDPU 7 = PQ: RLAN 8 = PQ: TLAN 9 = PQ: RX_PE 10 = PQ: TX_PE 11 = PQ: PMAT 12 = PQ: MNG 13 = PQ: RDPU 14 = VDM 15 = Don't care 16 = HIU: CFG/Slave/MSG/MSI-X 126 = LCB's internal config space registers 127 = LCB's internal memory space registers
RESERVED	31:28	0x0	RSV	Reserved.

8.2.2.4.18 PCIe Statistic Control Register #1 - PCI_GSCL_1 (0x00011800)

This register controls the operation of the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
GIO_COUNT_EN_0	0	0b	RW	**GIO Counter 0 Enable** Enables PCIe statistic counter number 0.
GIO_COUNT_EN_1	1	0b	RW	**GIO Counter 1 Enable** Enables PCIe statistic counter number 1.
GIO_COUNT_EN_2	2	0b	RW	**GIO Counter 2 Enable** Enables PCIe statistic counter number 2.
GIO_COUNT_EN_3	3	0b	RW	**GIO Counter 3 Enable** Enables PCIe statistic counter number 3.
LBC_ENABLE_0	4	0b	RW	**Leaky Bucket Counter 0 Enable** 0b = Leaky Bucket mode is disabled and the counter is incremented by one for each event. 1b = Statistics counter 0 operates in Leaky Bucket mode.
LBC_ENABLE_1	5	0b	RW	**Leaky Bucket Counter 1 Enable** 0b = Leaky Bucket mode is disabled and the counter is incremented by one for each event. 1b = Statistics counter 1 operates in Leaky Bucket mode.
LBC_ENABLE_2	6	0b	RW	**Leaky Bucket Counter 2 Enable** 0b = Leaky Bucket mode is disabled and the counter is incremented by one for each event. 1b = Statistics counter 2 operates in Leaky Bucket mode.
LBC_ENABLE_3	7	0b	RW	**Leaky Bucket Counter 3 Enable** 0b = Leaky Bucket mode is disabled and the counter is incremented by one for each event. 1b = Statistics counter 3 operates in Leaky Bucket mode.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_LAT_EN	8	0b	RW	**PCIe Count Latency Enable** Enables the latency counter. 0b = Disable the latency counter. 1b = Enable the latency counter.
PCI_COUNT_LAT_EV	13:9	0x0	RW	**PCIe Count Latency Event** Selects the event to be measured.
PCI_COUNT_BW_EN	14	0b	RW	**PCIe Count Bandwidth Enable** Enables the bandwidth counter. 0b = Disable the bandwidth counter. 1b = Enable the bandwidth counter.
PCI_COUNT_BW_EV	19:15	0x0	RW	**PCIe Count Bandwidth Event** Selects the event to be measured.
RESERVED	27:20	0x0	RSV	Reserved.
GIO_64_BIT_EN	28	0b	RW	**GIO 64-bit Enable** Enables two 64-bit counters instead of four 32-bit counters.
GIO_COUNT_RESET	29	0b	RWS	**GIO Counter Reset** Reset indication of PCIe statistic counters. Write of zero has no effect.
GIO_COUNT_STOP	30	0b	RWS	**GIO Counter Stop** Stop indication of PCIe statistic counters. Write of zero has no effect.
GIO_COUNT_START	31	0b	RWS	**GIO Counter Start** Start indication of PCIe statistic counters. Write of zero has no effect.

8.2.2.4.19 PCIe Statistic Control Registers #2 - PCI_GSCL_2 (0x00011804)

This register defines the events counted by the performance counters.

Field	Bit(s)	Init.	Type	Description
GIO_EVENT_NUM_0	7:0	0x0	RW	**GIO Event Number 0** Event number that counter 0 counts (GSCN_0).
GIO_EVENT_NUM_1	15:8	0x0	RW	**GIO Event Number 1** Event number that counter 1 counts (GSCN_1).
GIO_EVENT_NUM_2	23:16	0x0	RW	**GIO Event Number 2** Event number that counter 2 counts (GSCN_2).
GIO_EVENT_NUM_3	31:24	0x0	RW	**GIO Event Number 3** Event number that counter 3 counts (GSCN_3).

8.2.2.4.20 PCIe Statistic Control Register #5...#8 - PCI_GSCL_5_8[n] (0x00011810 + 0x4*n, n=0...3)

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	**Leaky Bucket Counter Threshold n** Threshold for the Leaky Bucket Counter n.
LBC_TIMER_N	31:16	0x0	RW	**Leaky Bucket Counter Timer n** Time period between decrementing the value in Leaky Bucket Counter n. The time period is defined in µs units.

8.2.2.4.21 PCIe Statistic Counter Registers #0...#3 - PCI_GSCN_0_3[n] (0x00011820 + 0x4*n, n=0...3)

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	**Event Counter** Event counter as defined in PCI_GSCL_2.*GIO_EVENT_NUM_[0...3]* fields. These registers are stuck at their maximum value of 0xFF...F.

8.2.2.5 PF - PCIe Configuration Space Setting Registers

This section contains registers used to set the default setting of the PCIe configuration space. These registers are reset and loaded from the NVM at PCIe reset.

8.2.2.5.1 PCIe PF Configuration - PCI_CNF (0x00011000)

Contains the per-PF configuration loaded from the NVM.

Field	Bit(s)	Init.	Type	Description
RESERVED	2:0	000b	RW	Reserved.
EXROM_DIS	3	1b	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.
INT_PIN	6:5	00b	RW	**Interrupt Pin** Controls the value advertised in the *Interrupt Pin* field of the PCI configuration header for this function. 00b = INTA# 01b = INTB# 10b = INTC# 11b = INTD# The value advertised in the PCI configuration header is the value loaded from NVM + 1. The default value for port 1 is 0x1.
RESERVED	31:7	0x0	RSV	Reserved.

8.2.2.5.2 PCIe PF Device ID - PCI_PFDEVID (0x00011008)

Contains the per-PF Device ID.

Field	Bit(s)	Init.	Type	Description
PF_DEV_ID_LAN	15:0	0x1562	RW	**PF Device ID LAN** Contains the Device ID for this PF when the function has an Ethernet device Class Code
PF_DEV_ID_SAN	31:16	0x1562	RW	**PF Device ID SAN** Contains the Device ID for this PF when the function has a SCSI device Class Code.

8.2.2.5.3 PCIe VF Device ID - PCI_VFDEVID (0x00011010)

Contains the per-PF Device IDs for its VFs.

Field	Bit(s)	Init.	Type	Description
VF_DEV_ID_LAN	15:0	0x1565	RW	**VF Device ID LAN** Contains the device ID for this PF's VFs when the function has an Ethernet device class code.
VF_DEV_ID_SAN	31:16	0x1565	RW	**VF Device ID SAN** Contains the device ID for this PF's VFs when the function has a SCSI device class code.

8.2.2.5.4 PCIe Storage Class - PCI_CLASS (0x00011038)

Contains the per-PF configuration loaded from the NVM.

Field	Bit(s)	Init.	Type	Description
STORAGE_CLASS	0	0b	RW	**Storage Class** 0b = The class code of this port is set to 0x020000 (LAN). 1b = The class code of this port is set to 0x010000 (SCSI).
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.5.5 PCIe Vendor ID - PCI_VENDORID (0x0001103C)

The Vendor ID exposed in config space. A value of 0xFFFF is not loaded to configuration space. Shared for all PFs.

Field	Bit(s)	Init.	Type	Description
VENDOR_ID	15:0	0x8086	RW	**Vendor ID** Contains the Vendor ID exposed in offset 0x0 in the configuration space of all functions. A value of 0xFFFF is ignored.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.5.6 PCI BAR Control - PCI_LBARCTRL (0x00011048)

Field	Bit(s)	Init.	Type	Description
PREFBAR	0	1b	RW	**Prefetch BAR** Prefetchable bit indication in the memory BARs (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.
CSRSIZE	2	0b	RW	**CSR Size** The *CSRSIZE* and *FL_BAR_SIZE* fields define the usable FLASH size and CSR mapping window size as described in Section 9.2.2.16.
RESERVED	5:3	001b	RSV	Reserved.
FL_BAR_SIZE	8:6	101b	RW	This field indicates the size of the external Flash as: `64KB x (2 ** FL_BAR_SIZE)` as used to define exposure in the BAR. The following values are supported: 101b = 2 MB 110b = 4 MB 111b = 8 MB All other values are reserved.
RESERVED	10:9	00b	RW	Reserved.

Field	Bit(s)	Init.	Type	Description
EXROM_BAR_SIZE	13:11	011b	RW	**EXROM BAR Size** This field indicates the size of the expansion ROM BAR as: `64KB x (2 ** EXROM_BAR_SIZE)` Values are: 000b = Corresponds to a 64 KB size. 001b = Corresponds to a 128 KB size. 010b = Corresponds to a 256 KB size. 011b = Corresponds to a 512 KB size. 100b = Corresponds to a 1 MB size. 101b = Corresponds to a 2 MB size. 110b = Corresponds to a 4 MB size. 111b = Corresponds to a 8 MB size. Default value is 512 KB (011b).
RESERVED	31:14	0x0	RSV	Reserved.

8.2.2.5.7 PCIe Subsystem ID - PCI_SUBSYSID (0x00011058)

Field	Bit(s)	Init.	Type	Description
SUB_VEN_ID	15:0	0x8086	RW	**Subsystem Vendor ID** Loaded to the PCI configuration Subsystem Vendor ID Register.
SUB_ID	31:16	0x0	RW	**Subsystem ID** Loaded to the PCI configuration Subsystem ID Register.

8.2.2.5.8 PCIe Power Data Register - PCI_PWRDATA (0x00011060)

Field	Bit(s)	Init.	Type	Description
D0_POWER	7:0	0x0	RW	**D0 Power** The value in this field is reflected in the PCI Power Management Data register of the LAN functions for D0 power consumption and dissipation (*Data_Select* = 0 or 4).
COMM_POWER	15:8	0x0	RW	**Comm Power** The value in this field is reflected in the PCI Power Management Data register of function 0 when the *Data_Select* field is set to 8 (common function).
D3_POWER	23:16	0x0	RW	**D3 Power** The value in this field is reflected in the PCI Power Management Data register of the LAN functions for D3 power consumption and dissipation (*Data_Select* = 3 or 7).
RESERVED	31:24	0x0	RSV	Reserved.

8.2.2.5.9 PCIe Serial Number MAC Address High - PCI_SERH (0x00011078)

Field	Bit(s)	Init.	Type	Description
SER_NUM_H	15:0	0x0	RW	**Serial Number High** The high Word of the Ethernet MAC Address used to generate the PCIe serial number.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.5.10 PCIe Capabilities Control - PCI_CAPCTRL (0x00011080)

Determines PCIe capabilities supported by the device and that software is allowed to enable or disable.

Field	Bit(s)	Init.	Type	Description
VPD_EN	0	0b	RW	**VPD Enable** 0b = The PCIe VPD Capability is not present and is not exposed. 1b = The PCIe VPD Capability is present and exposed.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.5.11 PCIe Capabilities Support - PCI_CAPSUP (0x00011088)

Determines PCIe capabilities supported by the device.

Field	Bit(s)	Init.	Type	Description
PCIE_VER	0	1b	RW	**PCIe Version** Determines the PCIe capability version. 0b = Capability version: 0x1 1b = Capability version: 0x2
RESERVED	1	0b	RSV	Reserved.
LTR_EN	2	1b	RW	**LTR Enable** A value of 1b indicates support for the PCIe Latency Tolerance Reporting (LTR) Capability.
TPH_EN	3	1b	RW	**TPH Enable** A value of 1b indicates support for the PCIe TPH Requester Capability.
ARI_EN	4	1b	RW	**ARI Enable** A value of 1b indicates support for the PCIe ARI Capability.
IOV_EN	5	1b	RW	**IOV Enable** A value of 1b indicates support for the PCIe SR-IOV Capability.
ACS_EN	6	1b	RW	**ACS Enable** A value of 1b indicates support for the PCIe ACS Capability.
SEC_EN	7	0b	RW	**SEC Enable** A value of 1b indicates support for the Secondary PCI Express Extended Capability.
RESERVED	14:8	0x0	RSV	Reserved.
ECRC_MCTP_GEN	15	0b	RW	**ECRC Generation for MCTP** 0b = Do not add ECRC to MCTP packets even if ECRC is enabled. 1b = Add ECRC to MCTP packets if ECRC is enabled via the *ECRC Generation Enable* field in PCIe Advanced Error Capabilities and Control Register.
ECRC_GEN_EN	16	1b	RW	**ECRC Generation Enable** Loaded into the *ECRC Generation Capable* bit of the PCIe Configuration registers.
ECRC_CHK_EN	17	1b	RW	**ECRC Check Enable** Loaded into the *ECRC Check Capable* bit of the PCIe Configuration registers.
IDO_EN	18	1b	RW	**IDO Enable** Enables ID-based ordering (IDO).
MSI_MASK	19	1b	RW	**MSI Mask** MSI per-vector masking setting. This bit is loaded to the masking bit (bit 8) in the Message Control of the MSI Configuration Capability structure.
CSR_CONF_EN	20	1b	RW	**CSP Configuration Enable** Enables Access to CSRs via the PCI Configuration Space. See Section 8.1.3.
RESERVED	29:21	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
LOAD_SUBSYS_ID	30	0b	RW	**Load Subsystem IDs** When set to 1b, indicates that the device loads its PCIe subsystem ID and sub-system vendor ID from NVM.
LOAD_DEV_ID	31	0b	RW	**Load Device ID** When set to 1b, indicates that the device loads its PCI device IDs from NVM.

8.2.2.5.12 PCIe Link Capabilities - PCI_LINKCAP (0x00011090)

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. 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	010b	RW	**Max Payload Size Supported** Loaded to the PCIe Device Capabilities Register. Supported values are: 000b = 128 B 001b = 256 B 010b = 512 B 011b = 1 KB 100b = 2 KB Otherwise, keep hardware default.
MAX_LINK_WIDTH	12:9	0x07	RW	**Max Link Width** Loaded to the PCIe Link Capabilities Register 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.

8.2.2.5.13 PCIe PM Support - PCI_PMSUP (0x000110A0)

This register contains parameters that define PCIe power management support.

Field	Bit(s)	Init.	Type	Description
RESERVED	7:0	0x97	RSV	Reserved.
L0S_ACC_LAT	10:8	011b	RW	**L0s Acceptable Latency** Loaded to the *Endpoint L0s Acceptable Latency* field in the PCIe Device Capabilities register.
L1_ACC_LAT	13:11	110b	RW	**L1 Acceptable Latency** Loaded to the *Endpoint L1 Acceptable Latency* field in the PCIe Device Capabilities register.
RESERVED	14	1b	RSV	Reserved.
RESERVED	31:15	0x0	RSV	Reserved.

8.2.2.5.14 PCIe VF Capabilities Support - PCI_VFSUP (0x000110A8)

Field	Bit(s)	Init.	Type	Description
VF_PREFETCH	0	1b	RW	**VF Prefetchable** 0b = IOV memory BARs are declared as non-prefetchable. 1b = IOV memory BARs are declared as prefetchable.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.5.15 PCIe Global Config - PCI_GLBL_CNF (0x000110B8)

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	00b	RSV	Reserved.
WAKE_PIN_EN	2	0b	RW	**Wake Pin Enable** When set to 1b, enables the use of PE_WAKE_N pin for PME event in all power states. Otherwise, it is enabled only in Dr state.
PCIE_CLKGATE_DIS	3	1b	RW	**PCIe Clock Gate Disable** When set to 0b, enables dynamic clock gating of the PCIe clocks (LCB, HIU & CSR).
PCIE_CLKGATE_L1_ONLY	4	1b	RW	**PCIe Clock Gate L1 Only** When set to 1b, and if *PCIE_CLKGATE_DIS* is 0, the clock gating of the PCIe clocks is only when the PCIe is in L1 state.
PCIE_PCLK_GATE_EN	5	0b	RW	**PCIe PCLK Gate Enable** When set to 1b, and if *PCIE_CLKGATE_DIS* is 0b, the PCIe PCLK is also dynamically gated.
RESERVED	7:6	00b	RSV	Reserved.
PCIE_CLKGATE_TIMER	15:8	0x10	RW	**PCIe Clock Gate Timer** Clock gating idle timer. This field defines the number of clocks (CSR clock) of idle-detect before gating the PCIe clocks.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.5.16 PCIe Upper Address - PCI_UPADD (0x000110E8)

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. Bits that should not be set in the addresses on the PCIe should be set in this field. For example in a 48-bit architecture, the value of this field should be 0xFFFF0000.

8.2.2.5.17 PCIe Serial Number MAC Address Low - PCI_SERL (0x000110F0)

Field	Bit(s)	Init.	Type	Description
SER_NUM_L	31:0	0x0	RW	**Serial Number Low** The low DW of the Ethernet MAC Address used to generate the PCIe serial number.

8.2.2.5.18 PCIe Global Config 2 - PCI_CNF2 (0x000110F8)

This register contains global status fields of the PCIe configuration.

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
CACHELINE_SIZE	1	0b	RW	**Cache Line Size** Determines the system cache line size 0b = 64 B 1b = 128 B This field is loaded from NVM.
MSI_X_PF_N	12:2	0x3F	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	0x02	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).
NUM_VFS	31:24	0x40	RW	**Number of VFs** The number of VFs requested by the function. Valid values are 0 - 64. If value is zero, SR-IOV capability is hidden.

8.2.2.6 PF - Interrupt Registers

8.2.2.6.1 Extended Interrupt Cause Register - EICR (0x00000800)

The EICR register is RW1C and can be optionally cleared on a read depending on the ODC flag setting in the GPIE register (Section 8.2.2.6.9).

Field	Bit(s)	Init.	Type	Description
RTXQ	15:0	0x0	RW1C	**Receive/Transmit Queue Interrupts** One bit per queue or a bundle of queues, activated on receive/transmit events.The mapping of queue to the RTxQ bits is done by the IVAR registers.
FLOW_DIRECTOR	16	0b	RW1C	**Flow Director** Flow director exception is activated by one of the following events: • Filter removal failed (there was no matched filter to be removed). • The number of remaining free filters in the flexible filter table exceeds (goes below) the FDIRCTRL.*FULL_THRESH* (Section 8.2.2.15.1). • Filter programming failed due to no space in the flow director table. Note that this case should not happen if the driver handles the FDIRCTRL.*FULL_THRESH* event.
RX_MISS	17	0b	RW1C	**Rx Miss** Missed packet interrupt is activated for each received packet that overflows the Rx packet buffer (overrun).
PCI_EXCEPTION	18	0b	RW1C	**PCI Exception** The PCI exception is activated by one of the events described in Section 3.1.5.8, "Proprietary Error Reporting" The specific PCI event is error reported in the PCI_ICAUSE register (Section 8.2.2.4.9).
MAILBOX	19	0b	RW1C	**Mailbox** VF to PF mailbox interrupt. Caused by a VF write access to the PF mailbox or by a malicious event detection
LSC	20	0b	RW1C	**Link Status Change** This bit is set each time the link status changes (either from up-to-down, or from down-to-up).
LINKSEC	21	0b	RW1C	**LinkSec** Indicates that the Tx LinkSec packet counter reached the threshold requiring key exchange.
MNG	22	0b	RW1C	**Manageability Event Detected** Indicates that a manageability event occurred. When the device is in power-down mode, the BMC might generate a PME for the same events that would cause an interrupt when the device is at the D0 state. This interrupt bit is also used to alert the host when the BMC IP Address was changed or when EEMNGCTL.*CFG_DONE0/1* bit is set by firmware (see Section 3.7.3.4.5). It is also used to indicate ECC events in the embedded management controller memories.
TS	23	0b	RW1C	**Thermal Sensor event** Indicates that a thermal sensor trip point was crossed.
TIMESYNC	24	0b	RW1C	**TimeSync Interrupt** Indicates that a Time Sync event has occurred. Check TSIM register for precise cause.
GPI_SDP0	25	0b	RW1C	**General Purpose Interrupt on SDP0** If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when a transition to high is sampled on SDP0.

Field	Bit(s)	Init.	Type	Description
GPI_SDP1	26	0b	RW1C	**General Purpose Interrupt on SDP1** If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when a transition to high is sampled on SDP1.
GPI_SDP2	27	0b	RW1C	**General Purpose Interrupt on SDP2** If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when the SDP2 is sampled high.
ECC	28	0b	RW1C	**Unrecoverable ECC error** This bit is set when one of the following occurs: • An unrecoverable error is detected in one of the device memories (except embedded management controller memories). • A CRC error occurred on the second attempt to load the PHY image from NVM. • PHY micro-controller watchdog failure. Software should issue a software reset following this error.
PHY_GLOBAL_INTERRUPT	29	0b	RW1C	**PHY Global Interrupt** PHY Interrupt (non-fatal).
TCP_TIMER	30	0b	RW1C	**TCP Timer Expired** This bit is set when the timer expires.
OTHER_CAUSE	31	0b	ROS	**Other Cause Interrupt** Activated when any Bit[29:16] in this register is set and its relevant mask bit in the EIMS register (Section 8.2.2.6.5) is enabled.

8.2.2.6.2 Extended Interrupt Cause Set Register - EICS (0x00000808)

Field	Bit(s)	Init.	Type	Description
INTERRUPT_CAUSE_SET	30:0	0x0	WO	**Interrupt Cause Set** Setting any bit in this field, sets its corresponding bit in the EICR register (Section 8.2.2.6.1)and generates an interrupt if enabled by the EIMS register (Section 8.2.2.6.5).
RESERVED	31	0b	RSV	Reserved.

8.2.2.6.3 Extended Interrupt Auto Clear Register - EIAC (0x00000810)

Note: Bits[29:16] should never be set to auto clear since they share the same MSI-X vector.

Field	Bit(s)	Init.	Type	Description
RTXQ_AUTO_CLEAR	15:0	0x0	RW	**RTxQ Auto-Clear** 0b = The corresponding bits in the EICR register (Section 8.2.2.6.1) are not auto-cleared. 1b = Each bit enables auto-clear of the corresponding RTxQ bits in the EICR register following interrupt assertion.
RESERVED	29:16	0x0	RSV	Reserved.
TCP_TIMER_AUTO_CLEAR	30	0b	RW	**TCP Timer Auto-Clear** 0b = Auto-clear is not enabled. 1b = This bit enables auto-clear of the TCP timer interrupt cause in the EICR register following interrupt assertion.
RESERVED	31	0b	RSV	Reserved.

8.2.2.6.4 Extended Interrupt Throttle Registers - EITR[n] (0x00000820 + 0x4*n, n=0...23 and 0x00012300 + 0x4*(n-24), n=24...128)

Mapping of the EITR registers to the MSI-X vectors is described in Section 7.3.4.3.3, "MSI-X Vectors Mapping to EITR".

Note: Additional address(es): 0x012300 + 4*(n-24), n=24...128

Field	Bit(s)	Init.	Type	Description
RESERVED	2:0	000b	RSV	Reserved.
ITR_INTERVAL	11:3	0x0	RW	**Interrupt Interval** Minimum inter-interrupt interval specified in 2.048 µs units at 1 GbE and 10 GbE link. At 100 Mb/s, link the interval is specified in 20.48 µs units. At 0x0, interrupt throttling is disabled while any event causes an immediate interrupt.
RESERVED	13:12	00b	RSV	Reserved.
HIGH_PRIORITY	14	0b	RW	**High Priority vector** Setting this bit causes assertion of this vector to break DMA coalescing.
RESERVED	15	0b	RSV	Reserved.
LLI_CREDIT	20:16	0x0	RW	**LLI Credit** Reflects the current credits for associated interrupt. When *CNT_WDIS* is not set on write cycle, this field must be set to zero.
ITR_COUNTER	27:21	0x0	RW	**Interrupt Counter** This field represents the 7 MS bits (out of 9 bits) of the ITR counter. It is a down counter that is loaded with *ITR_INTERVAL* value each time the associated interrupt is asserted. When the ITR counter reaches zero, it stops counting and triggers an interrupt. On a write cycle, software must set this field to 0 if *CNT_WDIS* in this register is cleared (write enable to the ITR counter).
RESERVED	30:28	000b	RSV	Reserved.
CNT_WDIS	31	0b	RW	**Counter Write Disable** Write disable to the LLI credit and ITR counter. 0b = Software must set the LLI credit and ITR counter to zero, which enables an immediate interrupt on packet reception. 1b = The LLI credit and ITR counter are not overwritten by the write access. This bit is write only and always read as zero.

8.2.2.6.5 Extended Interrupt Mask Set/Read Register - EIMS (0x00000880)

Field	Bit(s)	Init.	Type	Description
INTERRUPT_ENABLE	30:0	0x0	RWS	**Interrupt Enable** Each bit that is set to 1b enables its corresponding interrupt in the EICR register (Section 8.2.2.6.1). Writing a 1b to any bit sets it. Writing 0b has no impact. Reading this register provides a map of those interrupts that are enabled.
RESERVED	31	0b	RSV	Reserved.

8.2.2.6.6 Extended Interrupt Mask Clear Register - EIMC (0x00000888)

Field	Bit(s)	Init.	Type	Description
INTERRUPT_MASK	30:0	0x0	WO	**Interrupt Mask** Writing a 1b to any bit clears its corresponding bit in the EIMS register (Section 8.2.2.6.5), disabling the corresponding interrupt in the EICR register (Section 8.2.2.6.1). Writing 0b has no impact. Reading this register provides no meaningful data.
RESERVED	31	0b	RSV	Reserved.

8.2.2.6.7 Extended Interrupt Auto Mask Enable Register - EIAM (0x00000890)

Field	Bit(s)	Init.	Type	Description
AUTO_MASK	30:0	0x0	RW	**Auto-Mask Enable** At 1b, each bit enables auto-set and auto-clear of its corresponding bits in the EIMS register (Section 8.2.2.6.5). ***Note:*** If any of the *AUTO_MASK* enable bits are set, the GPIE.*EIAME* bit (Section 8.2.2.6.9) must be set as well.
RESERVED	31	0b	RSV	Reserved.

8.2.2.6.8 MSIX to EITR Select - EITRSEL (0x00000894)

Field	Bit(s)	Init.	Type	Description
VFSELECT	31:0	0x0	RW	**Virtual Function Select** Each bit 'n' in this register selects the VF index (32+'n') or PF interrupt source for the EITR registers (VF 0-31 are not multiplexed as described in Section 7.3.4.3.3, "MSI-X Vectors Mapping to EITR"). 0b = Selects the PF. 1b = Selects the VF.

8.2.2.6.9 General Purpose Interrupt Enable - GPIE (0x00000898)

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
SDP0_GPIEN	1	0b	RW	**General Purpose Interrupt Detection Enable for SDP0** If software-controllable I/O pin SDP0 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection, resulting in setting EICR[25] bit.
SDP1_GPIEN	2	0b	RW	**General Purpose Interrupt Detection Enable for SDP1** If software-controllable I/O pin SDP1 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection., resulting in setting EICR[26] bit
SDP2_GPIEN	3	0b	RW	**General Purpose Interrupt Detection Enable for SDP2** If software-controllable I/O pin SDP2 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection, resulting in setting EICR[27] bit

Field	Bit(s)	Init.	Type	Description
MULTIPLE_MSIX	4	0b	RW	**MSI-X Mode** Selects between MSI-X interrupts and other interrupt modes. 0b = Legacy and MSI mode (non-MSI-X mode). 1b = MSI-X mode.
OCD	5	0b	RW	**Other Clear Disable** 0b = The whole EICR register is cleared on read. 1b = Only Bits[29:16] of the EICR are cleared on read.
EIMEN	6	0b	RW	**EICS Immediate Interrupt Enable** 0b = The EICS interrupt waits for EITR expiration 1b = Setting bit in the EICS causes a Low Latency Interrupt.
RESERVED	10:7	0x0	RSV	Reserved.
RSC_DELAY	13:11	000b	RW	**RCS Delay** Delay from RSC completion triggered by ITR and interrupt assertion. The delay in 1 GbE or 10 GbE mode equals: (*RSC_DELAY* + 1) x 4 µs = 4, 8, 12...32 µs The delay in 100 Mb/s mode equals: (*RSC_DELAY* + 1) x 40 µs = 40, 80, 120...320 µs
VT_MODE	15:14	00b	RW	**VT Mode** Specifies the number of active Virtual functions. Software must set this field the same as GCR_EXT.*VT_MODE* (Section 8.2.2.4.4). 00b = Non IOV mode. 01b = 16 VF mode. 10b = 32 VF mode. 11b = 64 VF mode.
RESERVED	29:16	0x0	RSV	Reserved.
EIAME	30	0b	RW	**Extended Interrupt Auto Mask Enable** When set, the EIMS register can be auto-cleared (depending on EIAM setting) upon interrupt assertion. In any case, the EIMS register can be auto-cleared (depending on EIAM setting) following a write-to-clear (or read) to the EICR register. Software might set the *EIAME* bit only in MSI-X mode.
PBA_SUPPORT	31	0b	RW	**PBA Support** When set, setting one of the extended interrupts masks via EIMS causes the PBA bit of the associated MSI-X vector to be cleared. Otherwise, the device behaves in a way supporting legacy INT-x interrupts. ***Note:*** Should be cleared when working in INT-x or MSI mode and set in MSI-X mode.

8.2.2.6.10 Interrupt Vector Allocation Registers - IVAR[n] (0x00000900 + 0x4*n, n=0...63)

These registers map interrupt causes into EICR entries (legacy/MSI modes) or into MSI-X vectors (MSI-X modes). See Section 7.3.4 for mapping and use of these registers. Transmit and receive queues mapping to IVAR registers is shown here:

Field	Bit(s)	Init.	Type	Description
INT_ALLOC_0	5:0	X	RW	**Interrupt Allocation 0** The interrupt allocation for Rx queue ('2xN' for IVAR register 'N').
RESERVED	6	0b	RSV	Reserved.
INT_ALLOC_VAL_0	7	0b	RW	**Interrupt Allocation Valid 0** Interrupt allocation valid indication for *INT_ALLOC*[0].
INT_ALLOC_1	13:8	X	RW	**Interrupt Allocation 1** The interrupt allocation for Tx queue ('2xN' for IVAR register 'N').

Field	Bit(s)	Init.	Type	Description
RESERVED	14	0b	RSV	Reserved.
INT_ALLOC_VAL_1	15	0b	RW	**Interrupt Allocation Valid 1** Interrupt allocation valid indication for *INT_ALLOC*[1].
INT_ALLOC_2	21:16	X	RW	**Interrupt Allocation 2** The interrupt allocation for Rx queue ('2xN'+1 for IVAR register 'N').
RESERVED	22	0b	RSV	Reserved.
INT_ALLOC_VAL_2	23	0b	RW	**Interrupt Allocation Valid 2** Interrupt allocation valid indication for *INT_ALLOC*[2].
INT_ALLOC_3	29:24	X	RW	**Interrupt Allocation 3** The interrupt allocation for Tx queue ('2xN'+1 for IVAR register 'N').
RESERVED	30	0b	RSV	Reserved.
INT_ALLOC_VAL_3	31	0b	RW	**Interrupt Allocation Valid 3** Interrupt allocation valid indication for *INT_ALLOC*[3].

8.2.2.6.11 Miscellaneous Interrupt Vector Allocation - IVAR_MISC (0x00000A00)

These register maps interrupt causes into MSI-X vectors (MSI-X modes). See Section 7.3.4 for mapping and use of these registers.

Note: The *INT_ALLOC_VAL_1* bit default value is 1b to enable legacy driver functionality.

Field	Bit(s)	Init.	Type	Description
INT_ALLOC_0	6:0	X	RW	**Interrupt Allocation 0** Defines the MSI-X vector assigned to the TCP timer interrupt cause. The value must be in the 0-63 range.
INT_ALLOC_VAL_0	7	0b	RW	**Interrupt Allocation Valid 0** Valid bit for *INT_ALLOC*[0].
INT_ALLOC_1	14:8	X	RW	**Interrupt Allocation 1** Defines the MSI-X vector assigned to the "Other" interrupt cause. The value must be in the 0-63 range.
INT_ALLOC_VAL_1	15	1b	RW	**Interrupt Allocation Valid 1** Valid bit for *INT_ALLOC*[1].
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.6.12 Extended Interrupt Cause Set Registers 1 - EICS1 (0x00000A90)

Field	Bit(s)	Init.	Type	Description
INTERRUPT_CAUSE_SET	31:0	0x0	WO	**Interrupt Cause Set** Setting any bit in this register sets its corresponding bit in the EICR[n] register, and generates an interrupt if enabled by the EIMS[n] register. Reading this register provides no meaningful data.

8.2.2.6.13 Extended Interrupt Cause Set Registers 2 - EICS2 (0x00000A94)

Field	Bit(s)	Init.	Type	Description
INTERRUPT_CAUSE_SET	31:0	0x0	WO	**Interrupt Cause Set** Setting any bit in this register sets its corresponding bit in the EICR[n] register, and generates an interrupt if enabled by the EIMS[n] register. Reading this register provides no meaningful data.

8.2.2.6.14 Extended Interrupt Mask Set/Read Registers - EIMS1 (0x00000AA0)

Field	Bit(s)	Init.	Type	Description
INTERRUPT_ENABLE	31:0	0x0	RWS	**Interrupt Enable** Each bit at 1b enables its corresponding interrupt in the EICR[n] register. Writing 1b to any bit sets it. Writing 0b has no impact. Reading this register provides a map of those interrupts that are enabled. Bits[15:0] of EIMS1 are mirrored in EIMS Bits[15:0].

8.2.2.6.15 Extended Interrupt Mask Set/Read Registers - EIMS2 (0x00000AA4)

Field	Bit(s)	Init.	Type	Description
INTERRUPT_ENABLE	31:0	0x0	RWS	**Interrupt Enable** Each bit at 1b enables its corresponding interrupt in the EICR[n] register. Writing 1b to any bit sets it. Writing 0b has no impact. Reading this register provides a map of those interrupts that are enabled. Bits[15:0] of EIMS1 are mirrored in EIMS Bits[15:0].

8.2.2.6.16 Extended Interrupt Mask Clear Registers 1 - EIMC1 (0x00000AB0)

Field	Bit(s)	Init.	Type	Description
INTERRUPT_MASK	31:0	0x0	WO	**Interrupt Mask** Writing 1b to any bit clears its corresponding bit in the EIMS[n] register, disabling the corresponding interrupt in the EICR[n] register. Writing 0b has no impact. Reading this register provides no meaningful data.

8.2.2.6.17 Extended Interrupt Mask Clear Registers 2 - EIMC2 (0x00000AB4)

Field	Bit(s)	Init.	Type	Description
INTERRUPT_MASK	31:0	0x0	WO	**Interrupt Mask** Writing 1b to any bit clears its corresponding bit in the EIMS[n] register, disabling the corresponding interrupt in the EICR[n] register. Writing 0b has no impact. Reading this register provides no meaningful data.

8.2.2.6.18 Extended Interrupt Auto Mask Enable Registers 1 - EIAM1 (0x00000AD0)

Field	Bit(s)	Init.	Type	Description
AUTO_MASK	31:0	0x0	RW	**Auto-Mask** At 1b, each bit enables auto-set and auto-clear of its corresponding bits in the EIMS[n] register. Bits[15:0] of EIAM1 are mirrored in EIAM Bits[15:0]. ***Note:*** If any of the *AUTO_MASK* enable bits is set, the GPIE.*EIAME* bit must be set as well.

8.2.2.6.19 Extended Interrupt Auto Mask Enable Registers 2 - EIAM2 (0x00000AD4)

Field	Bit(s)	Init.	Type	Description
AUTO_MASK	31:0	0x0	RW	**Auto-Mask** At 1b, each bit enables auto-set and auto-clear of its corresponding bits in the EIMS[n] register. Bits[15:0] of EIAM1 are mirrored in EIAM Bits[15:0]. ***Note:*** If any of the *AUTO_MASK* enable bits is set, the GPIE.*EIAME* bit must be set as well.

8.2.2.6.20 RSC Enable Interrupt - RSCINT[n] (0x00012000 + 0x4*n, n=0...128)

Field	Bit(s)	Init.	Type	Description
RSCEN	0	1b	RW	**RSC Enable** This bit enables RSC on the receive queues associated with interrupt vector n.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.7 PF - MSI-X Table Registers

The MSI-X capability is described in Section 9.2.3.3. The MSI-X table is described in Section 9.2.3.3.4, and the Pending Bit Array (PBA) is described in Section 9.2.3.3.5. These registers are located in the MSI-X BAR.

8.2.2.7.1 MSI-X PBA Clear - PBACL[n] (0x000110C0 + 0x4*n, n=0...7)

Note: PBACL[0] is also mapped to address 0x11068 to maintain compatibility with previous products.

Field	Bit(s)	Init.	Type	Description
PENBITCLR	31:0	0x0	RW	**MSI-X Pending Bits Clear** Writing 1b to any bit clears it's content. Writing 0b has no effect. Reading this register returns the MSIPBA.*PENBIT* value.

8.2.2.7.2 VF MSI-X PBA Clear - VFPBACL[n] (0x000110C8 + 0x4*n, n=0...5)

These registers reflect the VFPBACL bits of the VFs. This PBA is a vector of 192 bits. The vector starts at Bit[31] of register p=5 and ends at Bit[0] of register p=0 (“reverse” ordering). Each VF has 3 bits in this vector while, PENBIT[2:0] of VF=vi are mapped to bits vi * 3... vi * 3 + 2. Explicitly, PENBIT[2] of VF0 is at Bit[31] of register p=5, and so on.

Field	Bit(s)	Init.	Type	Description
PENBITCLR	31:0	0x0	RW	**MSI-X Pending Bits Clear** Writing 1b to any bit clears it's content. Writing 0b has no effect. Reading this register returns the MSIPBA.*PENBIT* value.

8.2.2.8 PF - Receive Registers

8.2.2.8.1 Receive Checksum Control - RXCSUM (0x00005000)

This register controls the receive checksum off-loading features of the X550.

Note: This register should only be initialized (written) when the receiver is not enabled. For example, only write this register when RXCTRL.*RXEN* = 0 (Section 8.2.2.9.11).

Field	Bit(s)	Init.	Type	Description
RESERVED	9:0	0x0	RSV	Reserved.
ICMPV6XSUM	10	0b	RW	**ICMPv6 Checksum Enable** 0b = Disable ICMPv6 checksum calculation. 1b = Enable ICMPv6 checksum calculation. ***Note:*** ICMPv6 checksum offload is supported only for packets sent to firmware for Proxying.
RESERVED	11	0b	RSV	Reserved.
IPPCSE	12	0b	RW	**IP Payload Checksum Enable** If set, a partial checksum is calculated for fragmented UDP packets. Relevant only if *PCSD* bit = 0b
PCSD	13	0b	RW	**RSS/Fragment Checksum Status Selection** The *Fragment Checksum* and *IP Identification* fields are mutually exclusive with the RSS hash. Only one of the two options is reported in the Rx descriptor. 0b = The extended descriptor write-back contains the fragment checksum. 1b = The extended descriptor write-back has the *RSS* field.
RESERVED	31:14	0x0	RSV	Reserved.

8.2.2.8.2 Receive Filter Control Register - RFCTL (0x00005008)

Field	Bit(s)	Init.	Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
RESERVED	5:0	0x0	RSV	Reserved.
NFSW_DIS	6	0b	RW	**NFS Write Disable** Disable filtering of NFS write request headers.
NFSR_DIS	7	0b	RW	**NFS Read Disable** Disable filtering of NFS read reply headers.
NFS_VER	9:8	00b	RW	**NFS Version** NFS version recognized by hardware. 00b = NFS version 2. 01b = NFS version 3. 10b = NFS version 4. 11b = Reserved for future use.
IPV6_DIS	10	0b	RW	**IPv6 Disable** Disable IPv6 packet filtering. Must always be set to 0b.
IP6XSUM_DIS	11	0b	RW	**IPv6 Xsum Disable** Disable XSUM on IPv6 packets. Must always be set to 0b.
RESERVED	13:12	00b	RSV	Reserved.
IPFRSP_DIS	14	0b	RW	**IP Fragment Split Disable** When this bit is set, the header of IP fragmented packets are not set. Must always be set to 0b.

Field	Bit(s)	Init.	Type	Description
RESERVED	15	0b	RSV	Reserved.
IPV6_EXDIS	16	0b	RW	**IPv6 Extension Header Disable** Chicken bit to disable the IPv6 extension headers parsing for XSUM offload, Header split and Filtering: 0b = Parse and recognize allowed IPV6 extension headers (Hop-by-Hop, Destination Options, and Routing). 1b = Reserved. Must always be set to 0b.
RESERVED	31:17	0x0	RSV	Reserved. Should be written with 0 to ensure future compatibility.

8.2.2.8.3 VXLAN Control - VXLANCTRL (0x0000507C)

Field	Bit(s)	Init.	Type	Description
UDPPORT	15:0	0x0	RW	**UDP Port** Defines the UDP port used to identify VXLAN traffic.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.8.4 Filter Control Register - FCTRL (0x00005080)

Note: Before receive filters are updated/modified, the RXCTRL.*RXEN* bit should be set to 0b (refer to Section 8.2.2.9.11). Once the proper filters are set, the RXCTRL.*RXEN* bit can be set to 1b to re-enable the receiver. In FCoE mode, DDP contexts should be invalidated before clearing RXCTRL.*RXEN* bit.

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
SBP	1	0b	RW	**Store Bad Packets** 0b = Do not store 1b = Store. ***Notes:*** 1. CRC errors before the SFD are ignored. Any packet must have a valid SFD (RX_DV with no RX_ER in the XGMII/GMII i/f) to be recognized by the device (even bad packets). 2. Packets with errors are not routed to manageability even if this bit is set. 3. Erroneous packets might be routed to the default queue rather than the originally-intended queue. 4. In packets shorter than 64 bytes, the checksum errors might be hidden while MAC errors are reported. 5. Packet with Valid Error (caused by Byte Error or Illegal Error) might have data contamination in the last 8 bytes when stored in the Host memory if the *SBP* bit is set.
RESERVED	6:2	0x0	RSV	Reserved.
TPE	7	0b	RW	**Tag Promiscuous Enable** When set, any packet with active tag is accepted. The active tag to accept is defined by the PFVTCTL.*POOLING_MODE* field (Section 8.2.2.22.12). If the PFVTCTL.*POOLING_MODE* field equals: 01b = Accept all packets with E-tag. 10b = Reserved. Other = Ignore this bit.

Field	Bit(s)	Init.	Type	Description
MPE	8	0b	RW	**Multicast Promiscuous Enable** 0b = Disabled 1b = Enabled When set, all received multicast and broadcast packets pass L2 filtering and can be directed to the MNG or Host by a one of the decision filters.
UPE	9	0b	RW	**Unicast Promiscuous Enable** 0b = Disabled 1b = Enabled
BAM	10	0b	RW	**Broadcast Accept Mode** 0b = Ignore broadcast packets to Host. 1b = Accept broadcast packets to Host.
RESERVED	31:11	0x0	RSV	Reserved.

8.2.2.8.5 E-tag EtherType Register - ETAG_ETYPE (0x00005084)

Field	Bit(s)	Init.	Type	Description
ETAG_ETYPE	15:0	0x893F	RW	**E-tag EtherType Value**
RESERVED	30:16	0x0	RSV	Reserved.
VALID	31	0b	RW	**Valid Bit** 0b = Entry is disabled. 1b = Entry is enabled.

8.2.2.8.6 VLAN Control Register - VLNCTRL (0x00005088)

Field	Bit(s)	Init.	Type	Description
VET	15:0	0x8100	RW	**VLAN Ether Type (the VLAN Tag Protocol Identifier - TPID)** This register contains the type field hardware that's matched against to recognize an 802.1Q (VLAN) Ethernet packet. For proper operation, software must not change the default setting of this field (802.3ac standard defines it as 0x8100). This field must be set to the same value as the *VT* field in the DMATXCTL register (Section 8.2.2.10.2). ***Note:*** This field appears in little endian order (the upper byte is first on the wire (VLNCTRL.*VET*[15:8]).
RESERVED	26:16	0x0	RSV	Reserved.
UP_FIRST_TAG_EN	27	1b	RW	**UP First Tag Enable** If set, the UP used for traffic class determination is taken from the first tag with UP,. Otherwise, it is taken from the inner VLAN.
DEI	28	0b	RW	**Drop Eligible Indicator Bit Value** If *DEIEN* is set to 1b, .1q packets with DEI equal to this field are accepted. Otherwise, the .1q packet is discarded.
DEIEN	29	0b	RW	**Drop Eligible Indicator Enable** 0b = Disabled (*DEI* bit not compared to decide packet acceptance). 1b = Enabled (DEI from packet must match the *DEI* field to accept a .1q packet).
VFE	30	0b	RW	**VLAN Filter Enable** 0b = Disabled (VFTA filter table does not decide packet acceptance). 1b = Enabled (VFTA filter table decides packet acceptance for .1q packets).
RESERVED	31	0b	RSV	Reserved.

8.2.2.8.7 Multicast Control Register - MCSTCTRL (0x00005090)

Field	Bit(s)	Init.	Type	Description
MO	00b	0x0	RW	**Multicast Offset** This determines which bits of the incoming multicast address are used in looking up the bit vector. 00b = [47:36] 01b = [46:35] 10b = [45:34] 11b = [43:32]
MFE	2	0b	RW	**Multicast Filter Enable** 0b = The multicast table array filter (MTA[n]) is disabled. 1b = The multicast table array (MTA[n]) is enabled.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.2.8.8 EType Queue Filter - ETQF[n] (0x00005128 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
ETYPE	15:0	0x0	RW	**EtherType** Identifies the protocol running on top of IEEE 802. Used to route Rx packets containing this EtherType to a specific Rx queue. ***Note:*** This field is defined in Little Endian (MS byte is first on the wire).
RESERVED	19:16	0x0	RSV	Reserved.
POOL	25:20	0x0	RW	**Pool** In virtualization modes, determines the target pool for the packet.
POOL_ENABLE	26	0b	RW	**Pool Enable** In virtualization modes, enables the *Pool* field.
FCOE	27	0b	RW	**FCoE** When set, packets that match this filter are identified as FCoE packets.
CNM	28	0b	RW	**CNM** When set, a packet with this EType is identified as a CNM packet.
TX_ANTISPOOF	29	0b	RW	**Transmit Anti-spoof** If set, this EtherType is candidate for anti-spoof action on transmitted packets. The actual action applied is set according to the PFVFSPOOF.ETHERTYPEAS or PFVFSPOOF.ETHERTYPELB (refer to Section 8.2.2.22.19).
IEEE_1588_TIME_STAMP	30	0b	RW	**IEEE 1588 Time Stamp** When set, packets with this EType are time-stamped according to the IEEE 1588 specification.
FILTER_ENABLE	31	0b	RW	**Filter Enable** 0b = The filter is disabled for any functionality. 1b = The filter is enabled. Exact actions are determined by separate bits.

8.2.2.8.9 Multicast Table Array - MTA[n] (0x00005200 + 0x4*n, n=0...127)

This table should be initialized by software before transmit and receive are enabled.

Field	Bit(s)	Init.	Type	Description
BIT_VECTOR	31:0	X	RW	**Bit Vector** Word wide bit vector specifying 32 bits in the multicast address filter table. This device provides multicast filtering for 4096 multicast addresses by providing single bit entry per multicast address. Those 4096 address locations organized in a Multicast Table Array (128 registers of 32 bits each). Only 12 bits out of the 48-bit destination address are considered as multicast address. Those 12 bits can be selected by *MO* field of MCSTCTRL register (Section 8.2.2.8.7). The 7 MS bits of the Ethernet MAC Address (out of the 12 bits) selects the register index, while the 5 LS bits (out of the 12 bits) selects the bit within a register.

8.2.2.8.10 Receive Address Low - RAL_ALIAS[n] (0x00005400 + 0x8*n, n=0...15; RW)

These registers are aliases to the first 16 RAL addresses.

Field definitions are the same as those defined in Section 8.2.2.8.14

8.2.2.8.11 Receive Address High - RAH_ALIAS[n] (0x00005404 + 0x8*n, n=0...15; RW)

These registers are aliases to the first 16 RAH addresses.

Field definitions are the same as those defined in Section 8.2.2.8.15

8.2.2.8.12 Multiple Receive Queues Command Register - MRQC_ALIAS (0x00005818; RW)

This is an alias to the MRQC register.

Field definitions are the same as those defined in Section 8.2.2.8.23

8.2.2.8.13 VLAN Filter Table Array - VFTA[n] (0x0000A000 + 0x4*n, n=0...127)

This table should be initialized by software before transmit and receive are enabled.

Field	Bit(s)	Init.	Type	Description
VLAN_FLT	31:0	X	RW	**VLAN Filter** Each bit 'i' in register 'n' affects packets with VLAN VID equal to 32*n+i. 128 VLAN Filter registers compose a table of 4096 bits that cover all possible VIDs. Each bit when set, allows packets with the associated VID to pass. Each bit when cleared, blocks packets with this VID.

8.2.2.8.14 Receive Address Low - RAL[n] (0x0000A200 + 0x8*n, n=0...127)

These registers contain the lower 32 bits of the 48-bit Ethernet MAC Address. All 32 bits are valid. If the NVM is present, the first register (RAL0) is loaded from the NVM.

Note: The first 16 MAC Addresses are also mapped to CSR addresses 0x5400-0x5478 for backward compatibility.

Field	Bit(s)	Init.	Type	Description
RAL	31:0	X	RW	**Receive Address Low** The lower 32 bits of the 48-bit Ethernet MAC Address. ***Note:*** This field is defined in Big Endian (LS byte of RAL is first on the wire).

8.2.2.8.15 Receive Address High - RAH[n] (0x0000A204 + 0x8*n, n=0...127)

These registers contain the upper 16 bits of the 48-bit Ethernet MAC Address. The complete address is {RAH, RAL}. AV determines whether this address is compared against the incoming packet. The *AV* field is cleared by a master reset.

Note: The first receive address register (RAR0) is also used for exact match pause frame checking (DA matches the first register). RAR0 should always be used to store the individual Ethernet MAC Address of the adapter. After reset, if the NVM is present, the first register (Receive Address Register 0) is loaded from the *IA* field in the NVM, its *Address Select* field is 00b, and its *Address Valid* field is 1b. If no NVM is present, the *Address Valid* field is 0b. The *Address Valid* field for all of the other registers are 0b. The first 16 MAC Addresses are also mapped to CSR addresses 0x5404 - 0x547C for backward compatibility.

Field	Bit(s)	Init.	Type	Description
RAH	15:0	X	RW	**Receive Address High** The upper 16 bits of the 48-bit Ethernet MAC Address. ***Note:*** This field is defined in Big Endian (MS byte of RAH is Last on the wire).
RESERVED	29:16	0x0	RSV	Reserved.
ADTYPE	30	0b	RW	**Address Type** 0b = MAC 1b = E-tag Fixed to zero in RAH[0].
AV	31	0b	RW	**Address Valid** All Receive Addresses are not initialized by hardware and software should initialize them before receive is enabled. If the NVM is present, the Receive Address[0] is loaded from the NVM and its *Address Valid* field is set to 1b after a software or PCI reset or NVM read.

8.2.2.8.16 MAC Pool Select Array - MPSAR[n] (0x0000A600 + 0x4*n, n=0...255)

Software should initialize these registers before transmit and receive are enabled.

Field	Bit(s)	Init.	Type	Description
POOL_ENA	31:0	X	RW	**Pool Enable bit array** Each couple of registers ‘2*n’ and ‘2*n+1’ are associated with Ethernet MAC Address Filter ‘n’ as defined by RAL[n] and RAH[n]. Each bit when set, enables packet reception with the associated Pools as described here: Bit ‘i’ in register ‘2*n’ is associated with POOL ‘i’. Bit ‘i’ in register ‘2*n+1’ is associated with POOL ‘32+i’.

8.2.2.8.17 Packet Split Receive Type Register - PSRTYPE[n] (0x0000EA00 + 0x4*n, n=0...63)

Notes:

- This register must be initialized by software.
- Packets are split according to the lowest-indexed entry that applies to the packet and that is enabled. For example, if bits 4 and 8 are set, an IPv4 packet that is not TCP is split after the IPv4 header. The exception to this rule is for tunnel packets. If PSR_TYPE15 (split on tunnel) or PSR_TYPE16 (Split on outer L2 header) is set, they take precedence on the other filters. At most one of those should be set.
- This bit mask table enables or disables each type of header to be split. A value of '1'b enables an entry.
- In virtualization mode, a separate PSRTYPE register is provided per pool up to the number of pools enabled. In non-virtualization mode, only PSRTYPE[0] is used.
- Additional address(es): 0x05480 + 4*n, n=0... 15

Field	Bit(s)	Init.	Type	Description
RESERVED	3:0	0x0	RSV	Reserved.
PSR_TYPE4	4	0b	RW	**PSR Type 4** Split received TCP packets after TCP header.
PSR_TYPE5	5	0b	RW	**PSR Type 5** Split received UDP packets after UDP header.
RESERVED	7:6	00b	RSV	Reserved.
PSR_TYPE8	8	0b	RW	**PSR Type 8** Split received IPv4 packets after IPv4 header.
PSR_TYPE9	9	0b	RW	**PSR Type 9** Split received IPv6 packets after IPv6 header.
RESERVED	11:10	00b	RSV	Reserved.
PSR_TYPE12	12	0b	RW	**PSR Type 12** Split received L2 packets after L2 header.
RESERVED	14:13	00b	RSV	Reserved.
PSR_TYPE15	15	0b	RW	**PSR Type 15** Split on tunnel header
PSR_TYPE16	16	0b	RW	**PSR Type 16** Split on outer L2 header
RESERVED	18:17	00b	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
RESERVED	28:19	X	RSV	Reserved.
RQPL	31:29	X	RW	**Receive Queue Pool** Defines the number of bits to use for RSS redirection of packets dedicated to this pool. Valid values are: 000b = All the traffic of the pool is sent to queue 0 of the pool. 001b = Up to 2 queues can be enabled for this pool. 010b = Up to 4 queues can be enabled for this pool. The default value should be 010b. This field is used only if MRQC.*MRQE* equals 1001b, 1010b, 1011b, 1110b or 1111b (refer to Section 8.2.2.8.23).

8.2.2.8.18 Redirection Table - RETA[n] (0x0000EB00 + 0x4*n, n=0...31)

The redirection table has 128 entries in 32 registers.

Note: The contents of the redirection table are not defined following reset of the Memory Configuration Registers. System software must initialize the table prior to enabling multiple receive queues. It might also update the redirection table during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

Field	Bit(s)	Init.	Type	Description
ENTRY0	5:0	X	RW	**Entry 0** Defines the RSS output index for hash value '4*n+0', where 'n' is the register index equal to 0...31.
RESERVED	7:6	00b	RSV	Reserved.
ENTRY1	13:8	X	RW	**Entry 1** Defines the RSS output index for hash value '4*n+1', where 'n' is the register index equal to 0...31.
RESERVED	15:14	00b	RSV	Reserved.
ENTRY2	21:16	X	RW	**Entry 2** Defines the RSS output index for hash value '4*n+2', where 'n' is the register index equal to 0...31.
RESERVED	23:22	00b	RSV	Reserved.
ENTRY3	29:24	X	RW	**Entry 3** Defines the RSS output index for hash value '4*n+3', where 'n' is the register index equal to 0...31.
RESERVED	31:30	00b	RSV	Reserved.

8.2.2.8.19 RSS Random Key Register - RSSRK[n] (0x0000EB80 + 0x4*n, n=0...9)

The RSS Random Key is 40 bytes wide. See Section 7.1.3.7.3, "RSS Hash Function".

Note: Additional address(es): 0x05C80 + 4*n, n=0...9

Field	Bit(s)	Init.	Type	Description
K0	7:0	0x0	RW	**Key 0** RSS Key Byte '4*n+0' of the RSS random key, for each register 'n'.
K1	15:8	0x0	RW	**Key 1** RSS Key Byte '4*n+1' of the RSS random key, for each register 'n'.
K2	23:16	0x0	RW	**Key 2** RSS Key Byte '4*n+2' of the RSS random key, for each register 'n'.
K3	31:24	0x0	RW	**Key 3** RSS Key Byte '4*n+3' of the RSS random key, for each register 'n'.

8.2.2.8.20 EType Queue Select - ETQS[n] (0x0000EC00 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0x0	RSV	Reserved.
RX_QUEUE	22:16	0x0	RW	**Receive Queue** Identifies the Rx queue associated with this EType.
RESERVED	30:23	0x0	RSV	Reserved.
QUEUE_ENABLE	31	0b	RW	**Queue Enable** When set, enables queueing of Rx packets by the EType defined in the matching ETQF register to the queue indicated in this register.

8.2.2.8.21 SYN Packet Queue Filter - SYNQF (0x0000EC30)

Field	Bit(s)	Init.	Type	Description
QUEUE_ENABLE	0	0b	RW	**Queue Enable** When set, enables routing of Rx packets to the queue indicated in this register.
RX_QUEUE	7:1	0x0	RW	**Receive Queue** Identifies an Rx queue associated with SYN packets.
RESERVED	31:8	0x0	RSV	Reserved.

8.2.2.8.22 RSS Queues per Traffic Class Register - RQTC (0x0000EC70)

Field	Bit(s)	Init.	Type	Description
RQTC0	2:0	100b	RW	**Receive Queue Traffic Class 0** Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 0. A value of zero means that all the traffic of TC0 is sent to queue 0 of the traffic class. This field is used only if MRQC.*MRQE* = 0100b or 0101b (see Section 8.2.2.8.23). A value of 7 (111b) in this field is not valid and is considered to be zero (000b).
RESERVED	3	0b	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
RQTC1	6:4	100b	RW	**Receive Queue Traffic Class 1** Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 1. A value of zero means that all the traffic of TC1 is sent to queue 0 of the traffic class. This field is used only if MRQC.*MRQE* = 0100b or 0101b (see Section 8.2.2.8.23). A value of 7 in this field is not valid and is considered as if zero is written.
RESERVED	7	0b	RSV	Reserved.
RQTC2	10:8	100b	RW	**Receive Queue Traffic Class 2** Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 2. A value of zero means that all the traffic of TC2 is sent to queue 0 of the traffic class. This field is used only if MRQC.*MRQE* = 0100b or 0101b (see Section 8.2.2.8.23). A value of 7 (111b) in this field is not valid and is considered to be zero (000b).
RESERVED	11	0b	RSV	Reserved.
RQTC3	14:12	100b	RW	**Receive Queue Traffic Class 3** Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 3. A value of zero means that all the traffic of TC3 is sent to queue 0 of the traffic class. This field is used only if MRQC.*MRQE* = 0100b or 0101b (see Section 8.2.2.8.23). A value of 7 (111b) in this field is not valid and is considered to be zero (000b).
RESERVED	15	0b	RSV	Reserved.
RQTC4	18:16	100b	RW	**Receive Queue Traffic Class 4** Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 4. A value of zero means that all the traffic of TC4 is sent to queue 0 of the traffic class. This field is used only if MRQC.*MRQE* = 0100b (see Section 8.2.2.8.23). A value of 7 (111b) in this field is not valid and is considered to be zero (000b).
RESERVED	19	0b	RSV	Reserved.
RQTC5	22:20	100b	RW	**Receive Queue Traffic Class 5** Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 5. A value of zero means that all the traffic of TC5 is sent to queue 0 of the traffic class. This field is used only if MRQC.*MRQE* = 0100b (see Section 8.2.2.8.23). A value of 7 (111b) in this field is not valid and is considered to be zero (000b).
RESERVED	23	0b	RSV	Reserved.
RQTC6	26:24	100b	RW	**Receive Queue Traffic Class 6** Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 6. A value of zero means that all the traffic of TC6 is sent to queue 0 of the traffic class. This field is used only if MRQC.*MRQE* = 0100b (see Section 8.2.2.8.23). A value of 7 (111b) in this field is not valid and is considered to be zero (000b).
RESERVED	27	0b	RSV	Reserved.
RQTC7	30:28	100b	RW	**Receive Queue Traffic Class 7** Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 7. A value of zero means that all the traffic of TC7 is sent to queue 0 of the traffic class. This field is used only if MRQC.*MRQE* = 0100b (see Section 8.2.2.8.23). A value of 7 (111b) in this field is not valid and is considered to be zero (000b).
RESERVED	31	0b	RSV	Reserved.

8.2.2.8.23 Multiple Receive Queues Command Register - MRQC (0x0000EC80)

Field	Bit(s)	Init.	Type	Description
MRQE	3:0	0x0	RW	**Multiple Receive Queues Enable** Defines allocation of the Rx queues per RSS, virtualization, and DCB. 0000b = RSS disabled. 0001b = RSS only — Single set of RSS 64 queues. 0010b = DCB and RSS disabled — 8 TCs, each allocated 1 queue. 0011b = DCB and RSS disabled — 4 TCs, each allocated 1 queue. 0100b = DCB and RSS — 8 TCs, each allocated 16 RSS queues. 0101b = DCB and RSS — 4 TCs, each allocated 32 RSS queues. 0110b = Reserved. 0111b = Reserved. 1000b = Virtualization only — 64 pools, no RSS, each pool allocated 2 queues. 1001b = Virtualization and RSS — 62 pools, each allocated 2 RSS queues — 1 pool allocated 4 queues. 1010b = Virtualization and RSS — 32 pools, each allocated 4 RSS queues. 1011b = Virtualization and RSS — 64 pools, each allocated 2 RSS queues. 1100b = Virtualization and DCB — 16 pools, each allocated 8 TCs. 1101b = Virtualization and DCB — 32 pools, each allocated 4 TCs. 1110b = Virtualization, DCB and RSS — 16 pools, each allocated 4 TCs, each allocated 2 RSS queues. 1111b = Virtualization and RSS — 60 pools, each allocated 2 RSS queues — 2 pools, each allocated 4 queues.
RESERVED	12:4	0x0	RSV	Reserved.
MULTIPLE_RSS	13	0b	RW	**Multiple RSS Enable** 0b = The device uses a single RSS Key (Legacy) 1b = The device uses up to 64 RSS Keys (one per pool) each RSS has a redirection table with 64 entries of 2 bits each. This bit is relevant only if *MRQE* = 1001b, 1010b, 1011b, 1110b, or 1111b
RSC_DIS_LP	14	0b	RW	**RSC Disable LLI & Push Coalescing** If set, RSC does not coalesce packets with LLI or PSH.
L3L4TXSWEN	15	0b	RW	**Enable L3/L4 filtering for Tx Switched packets** 0b = L3/L4 filtering (including RSS, 5-Tuples, Flow Director) is disabled. Packets are directed to the default queue (queue 0) per pool. 1b = L3/L4 filtering is enabled.
RSS_FIELD_ENABLE	31:16	0x0	RW	**RSS Field Enable** Each bit, when set, enables a specific field selection to be used by the hash function. Several bits can be set at the same time. Bit[16] = Enable TcpIPv4 hash function. Bit[17] = Enable IPv4 hash function. Bit[18] = Reserved. Bit[19] = Reserved. Bit[20] = Enable IPv6 hash function. Bit[21] = Enable TcpIPv6 hash function. Bit[22] = Enable UdpIPV4. Bit[23] = Enable UdpIPV6. Bit[24] = Reserved. Bits[31:25] = Reserved. Zero. ***Note:*** On Tunnel packets IPv4-IPv6, only the IPv4 header might be used for the RSS filtering.

8.2.2.8.24 Extended Redirection Table - ERETA[n] (0x0000EE80 + 0x4*n, n=0...95)

The extended redirection table adds 384 entries to extend RETA in 96 registers.

Note: The contents of the extended redirection table are not defined following reset of the Memory Configuration Registers. System software must initialize the Table prior to enabling multiple receive queues. It might also update the redirection table during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

Field	Bit(s)	Init.	Type	Description
ENTRY0	5:0	X	RW	**Entry 0** Defines the RSS output index for hash value ‘4*[n+32]’, where ‘n’ is the register index equal to 0...95.
RESERVED	7:6	00b	RSV	Reserved.
ENTRY1	13:8	X	RW	**Entry 1** Defines the RSS output index for hash value ‘4*[n+32]+1’, where ‘n’ is the register index equal to 0...95.
RESERVED	15:14	00b	RSV	Reserved.
ENTRY2	21:16	X	RW	**Entry 2** Defines the RSS output index for hash value ‘4*[n+32]+2’, where ‘n’ is the register index equal to 0...95.
RESERVED	23:22	00b	RSV	Reserved.
ENTRY3	29:24	X	RW	**Entry 3** Defines the RSS output index for hash value ‘4*[n+32]+3’, where ‘n’ is the register index equal to 0...95.
RESERVED	31:30	00b	RSV	Reserved.

8.2.2.8.25 Per-Pool RSS Random Key Register - VFRSSRK[n,m] (0x00018000 + 0x4*n + 0x40*m, n=0...15, m=0...63)

The RSS Random Key is 40 bytes wide.

Note: Only the 10 first registers in each VF array are implemented (0x00 - 0x24).

Field	Bit(s)	Init.	Type	Description
K0	7:0	0x0	RW	**Key 0** RSS Key Byte ‘4*n+0’ of the RSS random key, for each register ‘n’.
K1	15:8	0x0	RW	**Key 1** RSS Key Byte ‘4*n+1’ of the RSS random key, for each register ‘n’.
K2	23:16	0x0	RW	**Key 2** RSS Key Byte ‘4*n+2’ of the RSS random key, for each register ‘n’.
K3	31:24	0x0	RW	**Key 3** RSS Key Byte ‘4*n+3’ of the RSS random key, for each register ‘n’.

8.2.2.8.26 Per-Pool Redirection Table - VFRETA[n,m] (0x00019000 + 0x4*n + 0x40*m, n=0...15, m=0...63)

The redirection table has 64 entries in 16 registers.

Note: The content of the redirection tables are not defined following reset of the Memory Configuration registers. System software must initialize the table prior to enabling multiple receive queues. It might also update the redirection table during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

Field	Bit(s)	Init.	Type	Description
ENTRY0	1:0	X	RW	**Entry 0** Defines the RSS output index for hash value '4*n+0', where 'n' is the register index equal to 0...31.
RESERVED	7:2	0x0	RSV	Reserved.
ENTRY1	9:8	X	RW	**Entry 1** Defines the RSS output index for hash value 4*n+1', where 'n' is the register index equal to 0...31.
RESERVED	15:10	0x0	RSV	Reserved.
ENTRY2	17:16	X	RW	**Entry 2** Defines the RSS output index for hash value '4*n+2', where 'n' is the register index equal to 0...31.
RESERVED	23:18	0x0	RSV	Reserved.
ENTRY3	25:24	X	RW	**Entry 3** Defines the RSS output index for hash value '4*n+3', where 'n' is the register index equal to 0...31.
RESERVED	31:26	0x0	RSV	Reserved.

8.2.2.9 PF - Receive DMA Registers

8.2.2.9.1 Receive Descriptor Base Address Low - RDBAL[n] (0x00001000 + 0x40*n, n=0...63 and 0x0000D000 + 0x40*(n-64), n=64...127)

This register contains the lower 32 bits of the 64-bit descriptor base address. The lower 7 bits are always ignored. The Receive Descriptor Base Address must point to a 128-byte aligned block of data.

Field	Bit(s)	Init.	Type	Description
ZERO	6:0	0x0	RW	**Zero** Ignored on writes. Returns 0 on reads. Bits[6:0]
RDBAL	31:7	X	RW	**Receive Descriptor Base Address Low** Bits[31:7]

8.2.2.9.2 Receive Descriptor Base Address High - RDBAH[n] (0x00001004 + 0x40*n, n=0...63 and 0x0000D004 + 0x40*(n-64), n=64...127)

This register contains the upper 32 bits of the 64-bit descriptor base address.

Field	Bit(s)	Init.	Type	Description
RDBAH	31:0	X	RW	**Receive Descriptor Base Address High** Bits[63:32]

8.2.2.9.3 Receive Descriptor Length - RDLEN[n] (0x00001008 + 0x40*n, n=0...63 and 0x0000D008 + 0x40*(n-64), n=64...127)

Note: Additional address(es): 0x0D008 + 0x40*(n-64), n=64...127

Field	Bit(s)	Init.	Type	Description
LEN	19:0	0x0	RW	**Descriptor Ring Length** This register sets the number of bytes allocated for descriptors in the circular descriptor buffer. It must be 128-byte aligned (7 LS bits must be set to zero). ***Note:*** Validated lengths up to 512 K (32 K descriptors).
RESERVED	31:20	0x0	RSV	Reserved. Reads as 0. Should be written to 0 for future compatibility.

8.2.2.9.4 Receive Descriptor Head - RDH[n] (0x00001010 + 0x40*n, n=0...63 and 0x0000D010 + 0x40*(n-64), n=64...127)

Note: Additional address(es): 0x0D010 + 0x40*(n-64), n=64...127

Field	Bit(s)	Init.	Type	Description
RDH	15:0	0x0	RO	**Receive Descriptor Head** This register holds the head pointer for the receive descriptor buffer in descriptor units (16-byte datum). The *RDH* is controlled by hardware.
RESERVED	31:16	0x0	RSV	Reserved. Should be written with 0.

8.2.2.9.5 Split Receive Control Registers - SRRCTL[n] (0x00001014 + 0x40*n, n=0...63 and 0x0000D014 + 0x40*(n-64), n=64...127)

Note: BSIZEHEADER must be bigger than 0 if DESCTYPE is equal to 010, 011 100 or 101.
Additional address(es): 0x0D014 + 0x40*(n-64), n=64...127 / 0x02100 + 4*n, [n=0...15]

Field	Bit(s)	Init.	Type	Description
BSIZEPACKET	4:0	0x2	RW	**Receive Buffer Size for Packet Buffer** The value is in 1-KB resolution. Value can be from 1 KB to 16 KB. Default buffer size is 2 KB. This field should not be set to 0. This field should be greater or equal to 2 in queues which RSC is enabled.
RESERVED	7:5	000b	RSV	Reserved. Should be written with 0 to ensure future compatibility.
BSIZEHEADER	13:8	0x4	RW	**Receive Buffer Size for Header Buffer** The value is in 64-byte resolution. Value can be from 64 bytes to 1024 bytes. The max supported header size is limited to 1023. Default buffer size is 256 bytes. This field must be greater than 0 if the value of *DESCTYPE* is greater than or equal to 2. Values above 1024 bytes are reserved for internal use only.
RESERVED	21:14	0x0	RSV	Reserved.
RDMTS	24:22	000b	RW	**Receive Descriptor Minimum Threshold Size** An interrupt associated with this queue is asserted whenever the number of free descriptors is decreased to *RDMTS* * 64 (this event is considered as Rx ring buffer "almost empty").
DESCTYPE	27:25	000b	RW	**Descriptor Type** Defines the descriptor type in Rx 000b = Legacy 001b = Advanced descriptor one buffer 010b = Advanced descriptor header splitting 011b = Reserved. 100b = Reserved.) 101b = Advanced descriptor header splitting always use header buffer. 110b = Reserved. 111b = Reserved.
DROP_EN	28	0b	RW	**Drop Enabled** If set to 1b, packets received to the queue when no descriptors are available to store them are dropped.
RESERVED	31:29	000b	RSV	Reserved. Should be written with 0 to ensure future compatibility.

8.2.2.9.6 Receive Descriptor Tail - RDT[n] (0x00001018 + 0x40*n, n=0...63 and 0x0000D018 + 0x40*(n-64), n=64...127)

Note: Software should write an even number to the tail register, if the Packet Split feature is used. The tail pointer should be set to one descriptor beyond the last empty descriptor in Host Descriptor Ring. Additional address(es): 0x0D018 + 0x40*(n-64), n=64...127.

Field	Bit(s)	Init.	Type	Description
RDT	15:0	0x0	RW	**Receive Descriptor Tail** This register contains the tail pointer for the receive descriptor buffer. The register points to a 16-byte datum. Software writes the tail register to add receive descriptors to the hardware free list for the ring.
RESERVED	31:16	0x0	RSV	Reserved. Reads as 0. Should be written to 0 for future compatibility.

8.2.2.9.7 Receive Descriptor Control - RXDCTL[n] (0x00001028 + 0x40*n, n=0...63 and 0x0000D028 + 0x40*(n-64), n=64...127)

Note: Additional address(es): 0x0D028 + 0x40*(n-64), n=64...127

Field	Bit(s)	Init.	Type	Description
RLPML	13:0	0x0	RW	**Long Packet Size Filter** Defined in bytes. Packets larger than the *RLPML* are discarded. The filter is enabled by the *RLMPL_EN* bit in this register. This field might not be supported per Rx queue in future products. ***Notes:*** 1. The device closes active RSC flows when an LLI packet was dropped due to *RLPML* exceeded. 2. The packet size compared to *RLMPL* includes any optional timestamp added into the packet.
RESERVED	14	0b	RSV	Reserved. Software might Read and Write maintaining backward compatibility.
RLPML_EN	15	0b	RW	**Enable Long Packet Size Filter** 0b = Disable 1b = Enable
RESERVED	22:16	0x0	RSV	Reserved. Software might Read and Write maintaining backward compatibility.
RESERVED	24:23	00b	RSV	Reserved.
ENABLE	25	0b	RW	**Receive Queue Enable** When set, the *ENABLE* bit enables the operation of the specific receive queue. On read, gets the actual status of the queue (internal indication that the queue is actually enabled/disabled).
RESERVED	29:26	0x0	RSV	Reserved.
VME	30	0b	RW	**VLAN Mode Enable** 0b = Do not strip VLAN tag. 1b = Strip VLAN tag from received 802.1Q packets destined to this queue.
RESERVED	31	0b	RSV	Reserved.

8.2.2.9.8 RSC Control - RSCCTL[n] (0x0000102C + 0x40*n, n=0...63 and 0x0000D02C + 0x40*(n-64), n=64...127)

Note: Additional address(es): 0x0D02C + 0x40*(n-64), n=64...127

Field	Bit(s)	Init.	Type	Description
RSCEN	0	0b	RW	**RSC Enable** When the *RSCEN* bit is set, RSC is enabled on this queue.
RESERVED	1	0b	RSV	Reserved.
MAXDESC	3:2	00b	RW	**Maximum Descriptors** Maximum descriptors per large receive as follows: 00b = Maximum of 1 descriptor per Large Receive. 01b = Maximum of 4 descriptors per Large Receive. 10b = Maximum of 8 descriptors per Large Receive. 11b = Maximum of 16 descriptors per Large Receive. ***Note:*** *MAXDESC* * SRRCTL.*BSIZEPACKET* must not exceed 64 KB minus 1 (which is the maximum total length in the IP header), and must be larger than expected received MSS.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.9.9 Split Receive Control Registers - SRRCTL_ALIAS[n] (0x00002100 + 0x4*n, n=0...15; RW)

Field definitions are the same as those defined in Section 8.2.2.9.5.

8.2.2.9.10 Receive DMA Control Register - RDRXCTL (0x00002F00)

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	00b	RSV	Reserved.
PSP	2	0b	RW	**Pad Small Receive Packets** If this field is set, in virtualized operating mode, strip CRC (HLREG0.*RXCRCSTRP*) should be set.
DMAIDONE	3	0b	RO	**DMA Initialization Done** When read as 1 indicates that the DMA initialization cycle is done (RO).
RESERVED	4	0b	RSV	Reserved.
RSC_PUSH_DIS	5	0b	RW	**RSC Push Disable** Disable coalescing of packets with PUSH flag asserted.
RESERVED	7:6	00b	RSV	Reserved.
TPH_AUTOLEARN	8	0b	RSV	TPH hints auto learn If set, the device learns the CPUID value in this register from the completer ID in read completion of receive descriptors fetch. ***Note:*** If TPH is disabled, this bit should not be set, as the learn capability is independent of the TPH capability.
RESERVED	27:9	0x30044	RSV	Reserved. Must be set to 0x30044.
MBINTEN	28	0b	RW	**Malicious Behavior Interrupt Enable for DMA Rx errors.** 0b = Disabled 1b = Enabled
MDP_EN	29	0b	RW	**Malicious Driver Protection Enable for DMA Rx** 0b = Mechanism is disabled. 1b = Mechanism is enabled.
RESERVED	31:30	00b	RSV	Reserved.

8.2.2.9.11 Receive Control Register - RXCTRL (0x00003000)

Field	Bit(s)	Init.	Type	Description
RXEN	0	0b	RW	**Receive Enable** When set to 0, packets are not received.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.9.12 Receive Packet Buffer Size - RXPBSIZE[n] (0x00003C00 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
RESERVED	9:0	0x0	RSV	Reserved.
SIZE	19:10	0x180	RW	**Size** Receive packet buffer size for traffic class 'n', where 'n' is the register index. The size is defined in KB units, and the default size of PB[0] is 384 KB. The default size of PB[1-7] is also 384 KB, but it is meaningless in non-DCB mode. When DCB mode is enabled, the size of PB[1-7] must be set to meaningful values. The total meaningful allocated PB sizes plus the size allocated to the flow director filters should be equal to 384 KB. Possible PB allocation in DCB mode for 8 x TCs is 0x30 (48 KB) for all PBs. Other possible setting of 4 x TCs is 0x60 (96 KB) for all PB[0-3], and 0x0 for PB[4-7]. Section 3.7.4.3.5, "Packet Buffer Size" for other optional settings with/without the flow director filters. ***Note:*** In any setting the *RXPBSIZE*[0] must always be enabled (greater than zero). Default value is 0x180 (384 KB).
RESERVED	31:20	0x0	RSV	Reserved.

8.2.2.10 PF - Transmit Registers

8.2.2.10.1 Tx Packet Buffer Threshold - TXPBTHRESH[n] (0x00004950 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
THRESH	9:0	0x96	RW	**Threshold** Used for checking room in the Tx packet buffer of TCn. *Threshold* is in KB units. When the packet buffer is filled up with payload over that threshold, no more data read requests are sent. Default values: 0x0 = (0 KB) for TXPBSIZE1-7. 0x96 = (150 KB) for TXPBSIZE0. It should be set to: `Packet Buffer Size - MSS` For example, if the packet buffer size is set to 20 KB in the corresponding TXPBSIZE.*SIZE* field, and if an MSS of 9.5 KB Jumbo frames is supported for TCn, it is set to 0xA (10 KB).
RESERVED	31:10	0x0	RSV	Reserved.

8.2.2.10.2 DMA Tx Control - DMATXCTL (0x00004A80)

Field	Bit(s)	Init.	Type	Description
TE	0	0b	RW	**Transmit Enable** When set, this bit enables the transmit operation of the DMA-Tx.
RESERVED	2:1	10b	RSV	Reserved.
GDV	3	0b	RW	**Global Double VLAN mode** When set, this bit enables the Double VLAN mode. Should be set to the same value as CTRL_EXT.*EXTENDED_VLAN* (Section 8.2.2.1.3).
RESERVED	4	1b	RSV	Reserved.
MDP_EN	5	0b	RW	**Malicious Driver Protection Enable for DMA Tx** 0b = Mechanism is disabled. 1b = Mechanism is enabled.
MBINTEN	6	0b	RW	**Malicious Behavior Interrupt Enable for DMA Tx errors** 0b = Disabled 1b = Enabled
TPH_AUTOLEARN	7	0b	RW	**TPH Hints Auto-Learn** If set, the device learns the CPUID value in this register from the completer ID in read completion of transmit descriptors fetch. ***Note:*** If TPH is disabled, this bit should not be set, as the learn capability is independent of the TPH capability.
RESERVED	15:8	0x0	RSV	Reserved.
VT	31:16	0x8100	RW	**VLAN Type** VLAN Ether-Type (the VLAN Tag Protocol Identifier - TPID). For proper operation, software must not change the default setting of this field (802.3ac standard defines it as 0x8100). This field must be set to the same value as the *VET* field in the VLNCTRL register (Section 8.2.2.8.6).

8.2.2.10.3 DMA Tx TCP Flags Control Low - DTXTCPFLGL (0x00004A88)

This register holds the "mask" bits for the TCP flags in Tx segmentation (described in Section 7.2.4.7.1 and Section 7.2.4.7.2).

Field	Bit(s)	Init.	Type	Description
TCP_FLG_FIRST_SEG	11:0	0xFF6	RW	**TCP Flags first Segment** Bits that make AND operation with the TCP flags at TCP header in the first segment.
RESERVED	15:12	0x0	RSV	Reserved.
TCP_FLG_MID_SEG	27:16	0xFF6	RW	**TCP Flags Middle Segments** The low bits that make AND operation with the TCP flags at TCP header in the middle segments.
RESERVED	31:28	0x0	RSV	Reserved.

8.2.2.10.4 DMA Tx TCP Flags Control High - DTXTCPFLGH (0x00004A8C)

This register holds the "mask" bits for the TCP flags in Tx segmentation (described in Section 7.2.4.7.3).

Field	Bit(s)	Init.	Type	Description
TCP_FLG_LST_SEG	11:0	0xF7F	RW	**TCP Flags Last Segment** Bits that make AND operation with the TCP flags at TCP header in the last segment.
RESERVED	31:12	0x0	RSV	Reserved.

8.2.2.10.5 Transmit Descriptor Base Address Low - TDBAL[n] (0x00006000 + 0x40*n, n=0...127)

This register contains the lower 32 bits of the 64-bit descriptor base address. The lower 7 bits are ignored. The Transmit Descriptor Base Address must point to a 128-byte aligned block of data.

Field	Bit(s)	Init.	Type	Description
ZERO	6:0	0x0	RW	**Zero** Ignored on writes. Returns 0 on reads. Bits[6:0]
TDBAL	31:7	X	RW	**Transmit Descriptor Base Address Low** Bits[31:7]

8.2.2.10.6 Transmit Descriptor Base Address High - TDBAH[n] (0x00006004 + 0x40*n, n=0...127)

This register contains the upper 32 bits of the 64-bit descriptor base address.

Field	Bit(s)	Init.	Type	Description
TDBAH	31:0	X	RW	**Transmit Descriptor Base Address High** Bits[63:32]

8.2.2.10.7 Transmit Descriptor Length - TDLEN[n] (0x00006008 + 0x40*n, n=0...127)

Field	Bit(s)	Init.	Type	Description
LEN	19:0	0x0	RW	**Descriptor Ring Length** This register sets the number of bytes allocated for descriptors in the circular descriptor buffer. It must be 128-byte aligned (7 LS bit must be set to zero). ***Note:*** Validated Lengths up to 512 K (32 K descriptors).
RESERVED	31:20	0x0	RSV	Reserved. Reads as 0. Should be written to 0.

8.2.2.10.8 Transmit Descriptor Head - TDH[n] (0x00006010 + 0x40*n, n=0...127)

Note: The values in these registers might point to descriptors that are still not in the host memory. As a result, the host cannot rely on these values to determine which descriptor to release.

Note: The only time that software should write to this register is after a reset (hardware reset or CTRL.*RST*) and before enabling the transmit function (TXDCTL.*ENABLE*). If software were to write to this register while the transmit function was enabled, the on-chip descriptor buffers might be invalidated and hardware could become confused.

Field	Bit(s)	Init.	Type	Description
TDH	15:0	0x0	RO	**Transmit Descriptor Head** This register contains the head pointer for the transmit descriptor ring. It points to a 16-byte datum. Hardware controls this pointer.
RESERVED	31:16	0x0	RSV	Reserved. Should be written with 0.

8.2.2.10.9 Transmit Descriptor Tail - TDT[n] (0x00006018 + 0x40*n, n=0...127)

Note: Hardware attempts to transmit all packets referenced by descriptors between head and tail.

Field	Bit(s)	Init.	Type	Description
TDT	15:0	0x0	RW	**Transmit Descriptor Tail** This register contains the tail pointer for the transmit descriptor ring. It points to a 16B datum. Software writes the tail pointer to add more descriptors to the transmit ready queue.
RESERVED	31:16	0x0	RSV	Reserved. Reads as 0. Should be written to 0 for future compatibility.

8.2.2.10.10 Transmit Descriptor Control - TXDCTL[n] (0x00006028 + 0x40*n, n=0...127)

This register controls the fetching and write back of transmit descriptors. The three threshold values are used to determine when descriptors are read from and written to host memory. For *PTHRESH* and *HTHRESH* recommended settings, see Section 7.2.3.4, "Transmit Descriptor Fetching".

Note: When *WTHRESH* = 0b, only descriptors with the *RS* bit set are written back.

Field	Bit(s)	Init.	Type	Description
PTHRESH	6:0	0x0	RW	**Pre-Fetch Threshold** Controls when a pre-fetch of descriptors is considered. This threshold refers to the number of valid, unprocessed transmit descriptors are in the on-chip buffer. If this number drops below *PTHRESH*, the algorithm considers pre-fetching descriptors from host memory. However, this fetch does not happen unless there are at least *HTHRESH* valid descriptors in host memory to fetch. ***Note:*** *HTHRESH* should be given a non-zero value each time *PTHRESH* is used.
RESERVED	7	0b	RSV	Reserved.
HTHRESH	14:8	0x0	RW	**Host Threshold**
RESERVED	15	0b	RSV	Reserved.
WTHRESH	22:16	0x0	RW	**Write-Back Threshold** Controls the write back of processed transmit descriptors. This threshold refers to the number of transmit descriptors in the on-chip buffer that are ready to be written back to host memory. In the absence of external events (explicit flushes), the write back occurs only after at least *WTHRESH* descriptors are available for write back. ***Notes:*** 1. Since the default value for a write-back threshold is 0b, descriptors are normally written back as soon as they are processed. *WTHRESH* must be written to a non-zero value to take advantage of the write back bursting capabilities. 2. When *WTHRESH* is set to a non-zero value, the software device driver should not set the *RS* bit in the Tx descriptors. When *WTHRESH* is set to zero, the software device driver should set the *RS* bit in the last Tx descriptors of every packet (if TSO is the last descriptor of the entire large send). 3. When head write back is enabled (TDWBAL[n].*HEAD_WB_EN* = 1b), the *WTHRESH* must be set to zero (refer to Section 8.2.2.10.11).
RESERVED	24:23	00b	RSV	Reserved.
ENABLE	25	0b	RW	**Transmit Queue Enable** When set, this bit enables the operation of a specific transmit queue. The default value for all queues is 0b. Setting this bit initializes all the internal registers of a specific queue. Until then, the state of the queue is kept and can be used for debug purposes. When disabling a queue, this bit is cleared only after all activity at the queue stopped. ***Note:*** 1. When setting the global Tx enable DMATXCTL.*TE*, the *Enable* bit of Tx queue zero is enabled as well (refer to Section 8.2.2.10.2). 2. Following a write cycle by software, this bit is set by hardware only when the queue is actually enabled. When read, gets the actual status of the queue (internal indication that the queue is actually enabled/disabled).
SWFLSH	26	0b	RW	**Transmit Software Flush** This bit enables software to trigger descriptor write-back flushing, independent of other conditions. This bit is self-cleared by hardware.
RESERVED	31:27	0x0	RSV	Reserved.

8.2.2.10.11 Tx Descriptor Completion Write Back Address Low - TDWBAL[n] (0x00006038 + 0x40*n, n=0...127)

Field	Bit(s)	Init.	Type	Description
HEAD_WB_EN	0	0b	RW	**Head Write-Back Enable** 0b = Head write-back is disabled. 1b = Head write-back is enabled. When *HEAD_WB_EN* is set, Tx descriptors are not written back.
RESERVED	1	0b	RSV	Reserved.
HEADWB_LOW	31:2	0x0	RW	**Head Write-Back Low** Lowest 32 bits of the head write-back memory location (DWORD aligned). The last two bits of this field are ignored and are always read as 00b, meaning that the actual address is QWORD aligned.

8.2.2.10.12 Tx Descriptor Completion Write Back Address High - TDWBAH[n] (0x0000603C + 0x40*n, n=0...127)

Field	Bit(s)	Init.	Type	Description
HEADWB_HIGH	31:0	0x0	RW	**Head Write-Back High** Highest 32 bits of Head write-back memory location (for 64-bit addressing).

8.2.2.10.13 DMA Tx TCP Max Allow Size Requests - DTXMXSZRQ (0x00008100)

This register limits the total number of data bytes that might be in outstanding PCIe requests from the host memory. This allows requests to send low latency packets to be serviced in a timely manner, as this request is serviced right after the current outstanding requests are completed.

Field	Bit(s)	Init.	Type	Description
MAX_BYTES_NUM_REQ	11:0	0x10	RW	**Max Allowed Number of Bytes Requests** The maximum allowed amount of 256 bytes outstanding requests. If the total size request is higher than the amount in the field, no arbitration is done and no new packet is requested.
RESERVED	31:12	0x0	RSV	Reserved.

8.2.2.10.14 Multiple Transmit Queues Command Register - MTQC (0x00008120)

Note: For permitted value and functionality of *DCB_ENA*, *VT_ENA*, and *NUM_TC_OR_Q*. For Tx queue assignment in DCB and VT modes, see Table 7-28 on page 420.

Note: This register can be modified only as part of the initialization phase.

Field	Bit(s)	Init.	Type	Description
DCB_ENA	0	0b	RW	**DCB Enabled Mode**
VT_ENA	1	0b	RW	**Virtualization Enabled Mode** When set, the device supports either 16, 32, or 64 Pools. This bit should be set the same as PFVTCTL.*VT_ENA* (Section 8.2.2.22.12).
NUM_TC_OR_Q	3:2	00b	RW	**Number of TCs or Number of Tx Queues per Pools** See functionality in Section 7.2.1.2.1, "Tx Queues Assignment".

Field	Bit(s)	Init.	Type	Description
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.10.15 Transmit Packet Buffer Size - TXPBSIZE[n] (0x0000CC00 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
RESERVED	9:0	0x0	RSV	Reserved.
SIZE	19:10	0xA0	RW	**Size** Transmit packet buffer size of TCn. At default (no DCB), only packet buffer 0 is enabled and TXPBSIZE values for TC 1-7 are meaningless. Other than the default configuration, the X550 supports partitioned configurations when DCB is enabled. Symmetrical 8 TCs partitioning: 0x14 (20KB) for TXPBSIZE[0...7]. Symmetrical 4 TCs partitioning: 0x28 (40KB) for TXPBSIZE[0...3] and 0x0 (0KB) for TXPBSIZE[4...7]. Non-symmetrical partitioning are supported as well. To enable wire speed transmission, it is recommended to set the transmit packet buffers to: • At least 2 times MSS plus PCIe latency (approximate 1 KB) when IPsec AH is not enabled (security block is not enabled or operates in path through mode). • At least 3 times MSS plus PCIe latency when IPsec AH is enabled (security block operates in store and forward mode). The default value is 0xA0 (160 KB)
RESERVED	31:20	0x0	RSV	Reserved.

8.2.2.10.16 Manageability Transmit TC Mapping - MNGTXMAP (0x0000CD10)

Field	Bit(s)	Init.	Type	Description
MAP	2:0	000b	RW	**MAP** Indicates the TC that the transmit Manageability traffic is routed to.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.2.10.17 Tags EtherTypes - TAG_ETYPE (0x00017100)

Field	Bit(s)	Init.	Type	Description
ETAG_ETHERTYPE	15:0	0x893F	RW	**E-tag EtherType** The EtherType to use to identify E-tags.
RESERVED	31:16	0x8926	RSV	Reserved.

8.2.2.11 PF - DCB Registers

DCB registers are owned by the PF in an IOV mode.

8.2.2.11.1 DCB Receive Packet Plane T4 Config - RTRPT4C[n] (0x00002140 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
CRQ	8:0	0x0	RW	**Credit Refill Quantum** Amount of credits to refill in 64-byte granularity. Possible values are 0x000 - 0x1FF (0 - 32,704 bytes)
BWG	11:9	000b	RW	**Bandwidth Group Index** Assignment of this PB to a bandwidth group.
MCL	23:12	0x0	RW	**Max Credit Limit** Max amount of credits for a configured PB in 64-byte granularity. Possible values are 0x000 - 0xFFF (0 - 262,080 bytes)
RESERVED	29:24	0x0	RSV	Reserved.
GSP	30	0b	RW	**Group Strict Priority** When set to 1b, enables strict priority to the appropriate PB over any traffic of other PBs within the Group.
LSP	31	0b	RW	**Link Strict Priority** If set to 1b, enables strict priority to the appropriate PB over any traffic of other PBs.

8.2.2.11.2 DCB Receive Packet Plane Control and Status - RTRPCS (0x00002430)

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
RRM	1	0b	RW	**Receive Recycle Mode** Defines the recycle mode within a BWG. 0b = No recycle. 1b = Recycle within the BWG. It is the only supported mode when DCB is enabled.
RAC	2	0b	RW	**Receive Arbitration Control** 0b = RR — Round Robin. 1b = WSP — Weighted Strict Priority.
RESERVED	5:3	000b	RSV	Reserved.
ARBDIS	6	0b	RW	**DCB Arbiter Disable** When set to 1b, this bit pauses the Rx packet plane arbitration state-machine. Therefore, during nominal operation this bit should be set to 0b.
RESERVED	26:7	0x0	RSV	Reserved.
BWGC	27	0b	RW	**BWG Credits** This bit affects the Rx WSP arbiter described in Figure 7-35 on page 491. 0b = The step "T5[BWG].Credits > 0? or no GSP in the BWG?" is taking affect (default operation). 1b = Do not bypass BWG credits consideration before transmit, meaning the second part of the condition case "or no GSP in the BWG?" is not taking place. ***Note:*** Bit RTTPCS(27) has similar affect on the Tx Packet plane Control as the BWGC on.
RESERVED	31:28	0x6	RSV	Reserved.

8.2.2.11.3 DCB Receive User Priority to Traffic Class - RTRUP2TC (0x00003020)

Field	Bit(s)	Init.	Type	Description
UP0MAP	2:0	000b	RW	**Receive User Priority 0 to Traffic Class Mapping** When set to n, User Priority 0 is bound to Traffic Class n. Used for two purposes: • Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 0 is routed. • Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the *Timer 0* field and *Class Enable Vector* bit 0 set is sent.
UP1MAP	5:3	000b	RW	**Receive User Priority 1 to Traffic Class Mapping** When set to n, User Priority 1 is bound to Traffic Class n. Used for two purposes: • Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 1 is routed. • Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the *Timer 1* field and *Class Enable Vector* bit 1 set is sent.
UP2MAP	8:6	000b	RW	**Receive User Priority 2 to Traffic Class Mapping** When set to n, User Priority 2 is bound to Traffic Class n. Used for two purposes: • Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 2 is routed. • Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the *Timer 2* field and *Class Enable Vector* bit 2 set is sent.
UP3MAP	11:9	000b	RW	**Receive User Priority 3 to Traffic Class Mapping** When set to n, User Priority 3 is bound to Traffic Class n. Used for two purposes: • Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 3 is routed. • Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the *Timer 3* field and *Class Enable Vector* bit 3 set is sent.
UP4MAP	14:12	000b	RW	**Receive User Priority 4 to Traffic Class Mapping** When set to n, User Priority 4 is bound to Traffic Class n. Used for two purposes: • Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 4 is routed. • Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the *Timer 4* field and *Class Enable Vector* bit 4 set is sent.
UP5MAP	17:15	000b	RW	**Receive User Priority 5 to Traffic Class Mapping** When set to n, User Priority 5 is bound to Traffic Class n. Used for two purposes: • Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 5 is routed. • Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the *Timer 5* field and *Class Enable Vector* bit 5 set is sent.
UP6MAP	20:18	000b	RW	**Receive User Priority 6 to Traffic Class Mapping** When set to n, User Priority 6 is bound to Traffic Class n. Used for two purposes: • Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 6 is routed. • Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the *Timer 6* field and *Class Enable Vector* bit 6 set is sent.

Field	Bit(s)	Init.	Type	Description
UP7MAP	23:21	000b	RW	**Receive User Priority 7 to Traffic Class Mapping.** When set to n, User Priority 7 is bound to Traffic Class n. Used for two purposes: • Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 7 is routed. • Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the *Timer 7* field and *Class Enable Vector* bit 7 set is sent.
RESERVED	31:24	0x0	RSV	Reserved.

8.2.2.11.4 DCB Transmit Descriptor Plane Queue Select - RTTDQSEL (0x00004904)

Field	Bit(s)	Init.	Type	Description
TXDQ_IDX	6:0	0x0	RW	**Tx Descriptor Queue Index or Tx Pool of Queues Index** This register is used to set VM and BQCN transmit scheduler parameters that are configured per Tx queue or per Tx pool of queues via indirect access. It means that prior to read or write access such registers, software has to make sure this field contains the index of the Tx queue or Tx pool of queue to be accessed. When DCB is disabled, VM parameters include a pool of Tx queues. As a result, this field points to the index of the pool (and not a queue index). When DCB is enabled, and/or when programming rate limiters, this field points to a Tx queue index. The registers concerned by this index are: • RTTDT1C (Section 8.2.2.11.5) • RTTQCNRC (Section 8.2.2.11.8) • RTTQCNRS (Section 8.2.2.11.9)
RESERVED	31:7	0x0	RSV	Reserved.

8.2.2.11.5 DCB Transmit Descriptor Plane T1 Config - RTTDT1C (0x00004908)

128 internal registers indirectly addressed via RTTDQSEL.*TXDQ_IDX* (refer to Section 8.2.2.11.4). When DCB is disabled, configure the pool index with the credits allocated to the entire pool.

Field	Bit(s)	Init.	Type	Description
CRQ	13:0	X	RW	**Credit Refill Quantum** Amount of credits to refill the VM in 64-byte granularity. Possible values are 0x000 - 0x3FFF (0 - 1,048,512 bytes)
RESERVED	31:14	0x0	RSV	Reserved.

8.2.2.11.6 DCB Transmit Descriptor Plane T2 Config - RTTDT2C[n] (0x00004910 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
CRQ	8:0	0x0	RW	**Credit Refill Quantum** Amount of credits to refill the TC in 64-byte granularity. Possible values are 0x000 - 0x1FF (0 - 32,704 bytes)
BWG	11:9	000b	RW	**Bandwidth Group Index** Assignment of this TC to a bandwidth group.

Field	Bit(s)	Init.	Type	Description
MCL	23:12	0x0	RW	**Max Credit Limit** Max amount of credits for a configured TC in 64-byte granularity. Possible values are 0x000 - 0xFFF (0 - 262,080 bytes)
RESERVED	29:24	0x0	RSV	Reserved.
GSP	30	0b	RW	**Group Strict Priority** When set to 1b, enables strict priority to the appropriate TC over any traffic of other TCs within the group.
LSP	31	0b	RW	**Link Strict Priority** When set to 1b, enables strict priority to the appropriate TC over any traffic of other TCs.

8.2.2.11.7 DCB Transmit QCN Rate-Scheduler MMW - RTTQCNRM (0x00004980)

Field	Bit(s)	Init.	Type	Description
MMW_SIZE	10:0	0x0	RW	**Max Memory Window Size** max memory window size for the QCN rate-scheduler, for all Tx queues. It is the maximum amount of 1 KB units of payload compensation time that can be accumulated for Tx queues attached to TCn. This number must be multiplied by the QCN Rate-Factor of the Tx queue before performing the MMW saturation check for that queue.
RESERVED	31:11	0x0	RSV	Reserved.

8.2.2.11.8 DCB Transmit QCN Rate-Scheduler Config - RTTQCNRC (0x00004984)

128 internal registers indirectly addressed via RTTDQSEL.*TXDQ_IDX* (refer to Section 8.2.2.11.4).

Field	Bit(s)	Init.	Type	Description
RF_DEC	13:0	X	RW	**Rate Factor Decimal** Tx QCN rate scheduler rate factor hexadecimal part for the Tx queue indexed by *TXDQ_IDX* field in RTTDQSEL register. Rate factor bits that come after the hexadecimal point. Meaningful only if *RS_ENA* bit is set.
RF_INT	23:14	X	RW	**Rate Factor Integral** Tx QCN rate scheduler rate factor integral part for the Tx queue indexed by *TXDQ_IDX* field in RTTDQSEL register. Rate factor bits that come before the hexadecimal point. Rate factor is defined as the ratio between the nominal link rate (i.e. 10 Gb/s) and the maximum rate allowed to that queue via QCN. Minimum allowed bandwidth share for a queue is 0.1% of the link rate, i.e. 10 Mb/s for devices operated at 10 Gb/s, leading to a maximum allowed rate factor of 1000. Meaningful only if *RS_ENA* bit is set.
RESERVED	30:24	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
RS_ENA	31	0b	SC	**Rate Scheduler Enable** Tx QCN rate scheduler enable for the Tx queue indexed by *TXDQ_IDX* field in RTTDQSEL register. 0b = The QCN rate factor programmed in this register is meaningless, and the switch for that queue is always forced to "on". The queue is not rate-controlled for QCN. 1b = The QCN rate programmed in this register is enforced (i.e., the queue is rate controlled for QCN). At the time it is set, the current timer value is loaded into the Timestamp stored for that entry. Bandwidth Group Assignment of this TC to a bandwidth group. Each TC must be assigned to a different BWG number, unless the TC is a member of a BWG. No more than two TCs can share the same BWG.

8.2.2.11.9 DCB Transmit QCN Rate-Scheduler Status - RTTQCNRS (0x00004988)

128 internal registers indirectly addressed via RTTDQSEL.*TXDQ_IDX* (refer to Section 8.2.2.11.4).

Field	Bit(s)	Init.	Type	Description
MIFS	31:0	0x0	RW	**Minimum Inter-Frame Spacing** Tx QCN rate scheduler current minimum inter-frame spacing for the Tx queue indexed by *TXDQ_IDX* field in RTTDQSEL register. When read, it is the current algebraic value of the MIFS interval for the queue, expressed in byte units (31 LS-bits taken), relative to the QCN rate scheduler. It is obtained by hardware subtracting the current value of the timer associated to that QCN rate-scheduler from the timestamp stored for that queue. A strict positive value means a switch in "off" state. It is expressed in 2's complement format.

8.2.2.11.10 DCB Transmit QCN Rate Reset - RTTQCNRR (0x0000498C)

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
QCN_CLEAR_ALL	1	0b	RW	**QCN Clear All** Clear all QCN rate-limiters. When set, the 128 RTTQCNRC.*RS_ENA* bits are cleared, releasing any active QCN rate-limiter. This bit must be set by software whenever the link speed has changed. This bit is self-cleared by hardware.
RESERVED	31:2	0x0	RSV	Reserved.

8.2.2.11.11 DCB Transmit QCN Tagging - RTTQCNTG (0x00004A90)

Field	Bit(s)	Init.	Type	Description
CNTGI	7:0	0x0	RW	**CN-tag Insertion enable per TC bitmap** When bit n is cleared, hardware does not insert any CN-tag in the transmitted frame (rate-controlled or not) sent from the TCn. It is assumed that QCN is disabled for the User Priorities mapped to that TC. When bit n is set, hardware inserts the proper CN-tag in transmitted frames (rate-controlled or not) sent from the TCn. It is assumed that QCN is enabled for the User Priorities mapped to that TC.
RESERVED	31:8	0x0	RSV	Reserved.

8.2.2.11.12 Strict Low Latency Tx Queues - TXLLQ[n] (0x000082E0 + 0x4*n, n=0...3)

Field	Bit(s)	Init.	Type	Description
STRICT_LOW_LATENCY	31:0	0x0	RW	**Strict Low Latency Enable** When set, defines the relevant Tx queue as strict low latency. All queues that belong to the LSP TC must be set as strict low latency queues. Bit 'm' in register 'n' corresponds to Tx queue 32 x 'n' + 'm'.

8.2.2.11.13 DCB Transmit User Priority to Traffic Class - RTTUP2TC (0x0000C800)

Field	Bit(s)	Init.	Type	Description
UP0MAP	2:0	0x0	RW	**Transmit User Priority 0 to Traffic Class Mapping** When set to n, User Priority 0 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the *Timer 0* field and *Class Enable Vector* bit 0 set, to determine which traffic class is paused.
UP1MAP	5:3	0x0	RW	**Transmit User Priority 1 to Traffic Class Mapping** When set to n, User Priority 1 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the *Timer 1* field and *Class Enable Vector* bit 1 set, to determine which traffic class is paused.
UP2MAP	8:6	0x0	RW	**Transmit User Priority 2 to Traffic Class Mapping** When set to n, User Priority 2 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the *Timer 2* field and *Class Enable Vector* bit 2 set, to determine which traffic class is paused.
UP3MAP	11:9	0x0	RW	**Transmit User Priority 3 to Traffic Class Mapping** When set to n, User Priority 3 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the *Timer 3* field and *Class Enable Vector* bit 3 set, to determine which traffic class is paused.
UP4MAP	14:12	0x0	RW	**Transmit User Priority 4 to Traffic Class Mapping** When set to n, User Priority 4 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the *Timer 4* field and *Class Enable Vector* bit 4 set, to determine which traffic class is paused.
UP5MAP	17:15	0x0	RW	**Transmit User Priority 5 to Traffic Class Mapping** When set to n, User Priority 5 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the *Timer 5* field and *Class Enable Vector* bit 5 set, to determine which traffic class is paused.
UP6MAP	20:18	0x0	RW	**Transmit User Priority 6 to Traffic Class Mapping** When set to n, User Priority 6 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the *Timer 6* field and *Class Enable Vector* bit 6 set, to determine which traffic class is paused.
UP7MAP	23:21	0x0	RW	**Transmit User Priority 7 to Traffic Class Mapping** When set to n, User Priority 7 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the *Timer 7* field and *Class Enable Vector* bit 7 set, to determine which traffic class is paused.
RESERVED	31:24	0x0	RSV	Reserved.

8.2.2.11.14 DCB Transmit Packet Plane Control and Status - RTTPCS (0x0000CD00)

Field	Bit(s)	Init.	Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
TPPAC	5	0b	RW	**Transmit Packet Plane Arbitration Control** 0b = RR — Round Robin (with respect to Stop Markers). 1b = SP — Strict Priority (with respect to Stop Markers).
ARBDIS	6	0b	RW	**DCB Arbiter Disable** When set to 1b this bit pauses the Tx Packet plane arbitration state-machine. Therefore, during nominal operation this bit should be set to 0.
RESERVED	7	0b	RSV	Reserved.
TPRM	8	0b	RW	**Transmit Packet Plane Recycle Mode** Defines the recycle mode within a BWG. 0b = No recycle. 1b = Recycle within the BWG.
RESERVED	21:9	0x0	RSV	Reserved.
ARBD	30:22	0x24	RW	**Arbitration Delay** Minimum cycles delay between packets arbitration. When RTTPCS.*TPPAC* is set to 1b, the arbitration delay is according to *ARBD*, otherwise the arbitration delay is 0x0. Should be kept at default in non EEDC mode. In EEDC mode, should be set to 0x004.
BYPASS_DCB_ARB	31	1b	RW	**Bypass DCB Arbiter** 0b = DCB arbiter is used. 1b = DCB arbiter is bypassed. Must be cleared in DCB mode.

8.2.2.11.15 DCB Transmit Packet Plane T2 Config - RTTPT2C[n] (0x0000CD20 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
CRQ	8:0	0x0	RW	**Credit Refill Quantum** Amount of credits to refill the TC in 64-byte granularity. Possible values are 0x000 - 0x1FF (0 - 32,704 bytes).
BWG	11:9	000b	RW	**Bandwidth Group** Assignment of this TC to a bandwidth group.
MCL	23:12	0x0	RW	**Max Credit Limit** Max amount of credits for a configured TC in 64-byte granularity. Possible values are 0x000 - 0xFFF (0 - 262,080 bytes).
RESERVED	29:24	0x0	RSV	Reserved.
GSP	30	0b	RW	**Group Strict Priority** When set to 1. enables strict priority to the appropriate TC over any traffic of other TCs within the Group.
LSP	31	0b	RW	**Link Strict Priority** When set to 1b, enables strict priority to the appropriate TC over any traffic of other TCs.

8.2.2.12 PF - TPH Registers

8.2.2.12.1 Rx TPH Control Register - TPH_RXCTRL[n] (0x0000100C + 0x40*n, n=0...63 and 0x0000D00C + 0x40*(n-64), n=64...127)

Note: Additional address(es): 0x0D00C + 0x40*(n-64), n=64...127

Field	Bit(s)	Init.	Type	Description
RX_DESC_FETCH_TPH_EN	0	0b	RW	**Receive Descriptor Fetch TPH Enable** 0b = Hardware does not enable TPH for descriptor fetches. 1b = Hardware enables TPH for all Rx descriptors fetch from memory. This bit is cleared as a default.
RX_DESC_WB_TPH_EN	1	0b	RW	**Receive Descriptor Writeback TPH Enable** 0b = Hardware does not enable TPH for descriptor write-backs. 1b = Hardware enables TPH for all Rx descriptors written back into memory. This bit is cleared as a default. The hint used is the hint set in the *Socket ID* field.
RX_HEADER_TPH_EN	2	0b	RW	**Receive Header TPH Enable** 0b = Hardware does not enable TPH for Rx headers. 1b = Hardware enables TPH for all received header buffers. This bit is cleared as a default. The hint used is the hint set in the *Socket ID* field.
RX_PAYLOAD_TPH_EN	3	0b	RW	**Receive Payload TPH Enable** 0b = Hardware does not enable TPH for Ethernet payloads. 1b = Hardware enables TPH for all Ethernet payloads written into memory. This bit is cleared as a default. The hint used is the hint set in the *Socket ID* field.
RESERVED	8:4	0x0	RSV	Reserved.
RXDESCREADROEN	9	1b	RW	**Rx Descriptor Read Relax Order Enable**
RESERVED	10	0b	RSV	Reserved.
RXDESCRWBROEN	11	0b	RW	**Rx Descriptor Write Back Relax Order Enable** This bit must be 0b to allow correct functionality of the descriptors write-back.
RESERVED	12	0b	RSV	Reserved.
RXDATAWRITEROEN	13	1b	RW	**Rx Data Write Relax Order Enable**
RESERVED	14	0b	RSV	Reserved.
RXREPHEADERROEN	15	1b	RW	**Rx Split Header Relax Order Enable** This bit impacts only the Relax Order setting of the header part writes. The Relax Order setting of the data part is controlled via the *RXDATAWRITEROEN* bit
RESERVED	23:16	0x0	RSV	Reserved.
CPUID	31:24	0x0	RW	**Physical ID** On TPH-capable platforms, the device driver programs a value based on the relevant Socket ID associated with this receive queue. ***Note:*** For TPH platforms, Bits[31:27] of this field should always be set to zero.

8.2.2.12.2 Rx TPH Control Register - TPH_RXCTRL_ALIAS[n] (0x00002200 + 0x4*n, n=0...15; RW)

Field definitions are the same as those defined in Section 8.2.2.12.1

8.2.2.12.3 Tx TPH Control Registers - TPH_TXCTRL[n] (0x0000600C + 0x40*n, n=0...127)

Note: Additional address(es): 0x0D00C + 0x40*(n-64), n=64...127

Field	Bit(s)	Init.	Type	Description
TX_DESC_FETCH_TPH_EN	0	0b	RW	**Transmit Descriptor Fetch TPH Enable** 0b = Hardware does not enable TPH for descriptor fetches. 1b = Hardware enables TPH for all Tx descriptors fetch from memory. This bit is cleared as a default.
TX_DESC_WB_TPH_EN	1	0b	RW	**Transmit Descriptor Writeback TPH Enable** 0b = Hardware does not enable TPH for descriptor write-backs. 1b = Hardware enables TPH for all Tx descriptors written back into memory. This bit is cleared as a default. The hint used is the hint set in the *Socket ID* field.
RESERVED	2	0b	RSV	Reserved.
TX_PACKET_TPH_EN	3	0b	RW	**Transmit Packet TPH Enable** 0b = Hardware does not enable TPH for Ethernet payloads. 1b = Hardware enables TPH for all Ethernet payloads read from memory. This bit is cleared as a default.
RESERVED	8:4	0x0	RSV	Reserved.
TXDESCREADROEN	9	1b	RW	**Tx Descriptor Read Relax Order Enable**
RESERVED	10	0b	RSV	Reserved.
TXDESCRWBROEN	11	0b	RW	**Tx Descriptor Write Back Relax Order Enable**
RESERVED	12	0b	RSV	Reserved.
TXDATAREADROEN	13	1b	RW	**Tx Data Read Relax Order Enable**
RESERVED	23:14	0x0	RSV	Reserved.
CPUID	31:24	0x0	RW	**Physical ID** On TPH-capable platforms, the device driver programs a value based on the relevant Socket ID associated with this transmit queue. ***Note:*** For TPH platforms, bits 31:27 of this field should always be set to zero.

8.2.2.13 PF - Timers Registers

8.2.2.13.1 TCP Timer - TCPTIMER (0x0000004C)

Field	Bit(s)	Init.	Type	Description
DURATION	7:0	0x0	RW	**Duration** Duration (in ms) of the TCP interrupt interval.
KICKSTART	8	0b	RWS	**Counter Kick-Start** 0b = No effect. 1b = Kick-starts the counter down-count from the initial value defined in *Duration* field.
TCPCOUNTEN	9	0b	RW	**TCP Count Enable** 0b = TCP timer counting is disabled. 1b = TCP timer counting is enabled. Upon enabling, TCP counter counts from its internal state. If the internal state is equal to zero, down-count does not restart until *KICKSTART* is activated. If the internal state is not 0b, down-count continues from the internal state. This enables a pause of the counting for debug purposes.
TCPCOUNTFINISH	10	0b	RWS	**TCP Count Finish** This bit enables software to trigger a TCP timer interrupt, regardless of the internal state. 0b = No effect. 1b = Triggers an interrupt, and resets the internal counter to its initial value. Down-count does not restart until either *KICKSTART* is activated or *Loop* is set.
LOOP	11	0b	RW	**TCP Loop** 0b = TCP counter stops at zero value, and does not re-start until *KICKSTART* is activated. 1b = TCP counter reloads duration each time it reaches zero, and goes on down-counting from this point without kick-starting.
RESERVED	31:12	0x0	RSV	Reserved.

8.2.2.14 PF - FCoE Registers

8.2.2.14.1 FC Indirect DMA Context: User Descriptor PTR Low - FCPTRL (0x00002410)

Field	Bit(s)	Init.	Type	Description
PTR_LOW	31:0	X	RW	**User Descriptor PTR Low** Four least significant bytes of the physical pointer to the User Descriptor list. The pointer must be 16-byte aligned so the 4 LS bits are read-only as zeros.

8.2.2.14.2 FC Indirect DMA Context: User Descriptor PTR High - FCPTRH (0x00002414)

Field	Bit(s)	Init.	Type	Description
PTR_HI	31:0	X	RW	**User Descriptor PTR High** Four most significant bytes of the physical pointer to the User Descriptor list.

8.2.2.14.3 FC Indirect DMA Context: Buffer Control - FCBUFF (0x00002418)

Field	Bit(s)	Init.	Type	Description
VALID	0	0b	RW	**DMA Context Valid** When set to 1b, indicates that the context is valid. If software clears the *VALID* bit, the software should poll it until it is actually cleared by hardware before unlocking the user buffers.
FIRST	1	0b	RW	**DMA First** This bit is a status indication. Software should clear it during FC context programming. The DMA unit sets this bit when it receives a frame that matches the context and marked by the Filter unit as first.
LAST	2	0b	RW	**DMA Last** This bit is a status indication. Software should clear it during FC context programming. Hardware sets this bit when it exhausts the last user buffer.
BUFFSIZE	4:3	00b	RW	**Buffer Size** This field defines the user buffer size used in this context as follows: 00b = 4 KB 01b = 8 KB 10b = 16 KB 11b = 64 KB
WRCONTX	5	0b	RW	**Write DDP Context** 0b = Rad exchange context aimed for initiator (originator) usage. 1b = Write exchange context aimed for target (responder) usage.
BUFFCNT	15:6	0x0	RW	**Buffer Count** Defines the number of the user buffer while 0x00 equals 1024. It is programmed by the software and updated by hardware during reception.
OFFSET	31:16	0x0	RW	**User Buffer Offset** Byte offset within the user buffer to which the FC data of large FC receive should be posted.

8.2.2.14.4 FC Indirect DMA Context: Receive DMA RW - FCDMARW (0x00002420)

Field	Bit(s)	Init.	Type	Description
FCOESEL	10:0	0x0	RW	**FCoE Context Select** This field defines the FCoE Rx context index (equals the RX_ID for that context).
RESERVED	13:11	000b	RSV	Reserved.
WE	14	0b	RW	**Write Enable** When this bit is set, the content of the FCPTRL, FCPTRH, and FCBUFF registers are programmed to the FCoE DMA context of index *FCOESEL*. This bit should never be set together with the *RE* bit in this register
RE	15	0b	RW	**Read Enable** When this bit is set, the internal FCoE DMA context of index *FCOESEL* is fetched to the FCPTRL, FCPTRH, and FCBUFF registers. This bit should never be set together with the *WE* bit in this register.
LASTSIZE	31:16	0x0	RW	**Last User Buffer Size** **Defines the size in bytes of the last user buffer. A value of 0 indicates a 64 KB buffer.**

8.2.2.14.5 FC Receive Control - FCRXCTRL (0x00005100)

Field	Bit(s)	Init.	Type	Description
RESERVED	11:0	0x0	RSV	Reserved.
SMAC_EN	12	1b	RW	**SMAC Enable** 0b = SMAC is ignored. 1b = SMAC field as part of the context matching of received FCoE packets.
DID_EN	13	1b	RW	**D_ID Enable** 0b = *D_ID* is ignored. 1b = Check *D_ID* field as part of the context matching of received FCoE packets.
RESERVED	31:14	0x0	RSV	Reserved.

8.2.2.14.6 FC Indirect Filter Context: Control - FCFLT (0x00005108)

Field	Bit(s)	Init.	Type	Description
VALID	0	0b	RW	**Filter Context Valid** When set to 1b, indicates that the context is valid.
FIRST	1	0b	RW	**Filter First** This bit is a status indication. Software should clear it during FC context programming. The Filter unit sets this bit when it receives a first frame that matches the context.
RESERVED	7:2	X	RSV	Reserved.
SEQ_ID	15:8	X	RW	**Sequence ID** The sequence ID of the last received frame. Initialized to 0 by the driver at context programming.
SEQ_CNT	31:16	X	RW	**Sequence Count** The sequence Count of the expected received frame. Should be set to 0 for read contexts. Should be set to *SEQ_CNT* + 1 of the last packet of the same exchange received from the initiator for write contexts.

8.2.2.14.7 FC Indirect Filter Context: Source MAC Address - FCSMAC (0x0000510C)

Note: This register is stored in network ordering (big endian).

Field	Bit(s)	Init.	Type	Description
FCSMAC	31:0	0x0	RW	**Source MAC Address (low bits)** Used to validate incoming FCoE exchange

8.2.2.14.8 FC Indirect Filter Context: RW Control - FCFLTRW (0x00005110)

Field	Bit(s)	Init.	Type	Description
FCOESEL	10:0	0x0	WO	**FCoE context Select** This field defines the FCoE Rx context index (equals the OX_ID or the RX_ID for that context).
RESERVED	12:11	00b	RSV	Reserved.
RE_VALIDATE	13	0b	WO	**Re-Validate** Fast re-validation of the filter context. Setting this bit together with the *WE* bit in this register validates the selected filter context. Hardware sets the *VALID* bit and clears the *First* bit (described in the FCFLT register in Section 8.2.2.14.6) while keeping all other filter parameters intact.
WE	14	0b	WO	**Write Enable** When this bit is set, the content of the FCFLT and FCSMAC registers, and portions of the FCFLTRW register are programmed to the Filter of index *FCOESEL*. This bit should never be set together with the *RE* bit in this register.
RE	15	0b	WO	**Read Enable** When this bit is set, the internal filter context of index *FCOESEL* is fetched to the FCFLT, FCSMAC, and FCFLTRW registers. This bit should never be set together with the *WE* bit in this register.
SMAC_HIGH	31:16	0x0	RW	**Source MAC Address (high bits)** Used to validate incoming FCoE exchange. This field is stored in network ordering (big endian).

8.2.2.14.9 FC Indirect Filter Context: D_ID Register - FCD_ID (0x00005114)

Field	Bit(s)	Init.	Type	Description
FCD_ID	23:0	0x0	RW	**FCD ID** The *D_ID* to validate incoming FCoE exchange.
RESERVED	31:24	0x0	RSV	Reserved.

8.2.2.14.10 FC Indirect Filter Context: Offset Parameter - FCPARAM (0x000051D8)

Field	Bit(s)	Init.	Type	Description
PARAM	31:0	0x0	RW	**FC Parameter** This field contains the expected FC Parameter in the next received frame. Initialized to 0 by the driver at context programming. ***Note:*** This field is defined in Big Endian (LS byte is first on the wire).

8.2.2.14.11 FCoE Redirection Control - FCRECTL (0x0000ED00)

Field	Bit(s)	Init.	Type	Description
ENA	0	0b	RW	**FC Redirection Enable** 0b = The redirection table is not active. 1b = The FC redirection is enabled. ***Note:*** When FC redirection is enabled, the *POOL_ENABLE* and the *QUEUE_ENABLE* in the ETQF and ETQS registers (respectively) must be cleared for FCoE data packets.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.14.12 FCoE Redirection Table - FCRETA[n] (0x0000ED10 + 0x4*n, n=0...31)

Field	Bit(s)	Init.	Type	Description
TABLE_ENTRY	6:0	0x0	RW	**Table Entry** Defines the redirection output queue number. Register 'n' is the *TABLE_ENTRY* index 'n', which is the matched value to the 6 LS bits of the FC exchange ID (for exchange IDs where 6 LS bits are between 0 and 31).
RESERVED	15:7	0x0	RSV	Reserved.
TABLE_ENTRY_HIGH	22:16	0x0	RW	**Table Entry High** Defines the redirection output queue number. Register 'n' is the *TABLE_ENTRY* high index '32+n', which is the matched value to the 6 LS bits of the FC exchange ID (for exchange IDs where 6 LS bits are between 32 and 63).
RESERVED	31:23	0x0	RSV	Reserved

8.2.2.14.13 FCoE Direct DMA Context - FCDDC[n,m] (0x00020000 + 0x4*n + 0x10*m, n=0...3, m=0...2047)

Field	Bit(s)	Init.	Type	Description
FCDDC	31:0	0x0	RW	**FCoE Direct DMA Context** Direct access to DMA context: DW[0x10*m] = FCPTRL[m] DW[0x10*m+4] = FCPTRH[m] DW[0x10*m+8] = FCBUFF[m] DW[0x10*m+0xC] = FCDMARW[m] (*LASTSIZE* bits only)

8.2.2.14.14 FCoE Direct Filter Context - FCDFC[n,m] (0x00028000 + 0x4*n + 0x10*m, n=0...3, m=0...2047)

Field	Bit(s)	Init.	Type	Description
FCDFCFLT	31:0	0x0	RW	**FCoE Direct Filter Context** Direct access to filter context: DW[0x10*m] = FCFLT[m] DW[0x10*m+0x4] = FCPARAM[m] DW[0x10*m+0x8] = FCSMAC[m] DW[0x10*m+0xC] = FCFLTRW[m] (*SMAC_HIGH* bits only)

8.2.2.14.15 FCoE Direct Filter Context D_ID - FCDFCD[n] (0x00030000 + 0x4*n, n=0...2047)

Field	Bit(s)	Init.	Type	Description
D_ID	23:0	0x0	RW	**D ID** *D_ID* to validate incoming FCoE exchange
RESERVED	31:24	0x0	RSV	Reserved.

8.2.2.15 PF - Flow Director Registers

These registers are used to control the flow director functionality.

8.2.2.15.1 Flow Director Filters Control Register - FDIRCTRL (0x0000EE00)

Note: This register should be configured ONLY as part of the Flow Director initialization flow or Clearing the Flow Director table. Programming of this register with non zero value in the *PBALLOC* field initializes the Flow Director table.

Field	Bit(s)	Init.	Type	Description
PBALLOC	1:0	00b	RW	**PB Allocation** Memory allocation for the Flow Director Filters. 00b = No memory allocation. Flow Director Filters are disabled. 01b = 64 KB (8K minus 1 signature filters or 2K minus 1 perfect match filters) 10b = 128 KB (16K minus 1 signature filters or 4K minus 1 perfect match filters) 11b = 256 KB (32K minus 1 signature filters or 8K minus 1 perfect match filters)
RESERVED	2	0b	RSV	Reserved.
INIT_DONE	3	0b	RO	**Initialization Done** Flow Director initialization completion indication (Read Only status). Indicates that hardware initialized the Flow Director table according to the *PBALLOC* setting. Software must not access any other Flow Director filters registers before the *INIT_DONE* bit is set. When Flow Director filters are enabled (*PBALLOC* > 0), the software must wait for *INIT_DONE* indication before Rx is enabled.
PERFECT_MATCH	4	0b	RW	**Perfect Match** Flow Director Filters Mode of operation. 0b = Hardware supports Signature filters according to the *PBALLOC*. 1b = Hardware supports Perfect Match filters according to the *PBALLOC*.
REPORT_STATUS	5	0b	RW	**Report Status** Report Flow Director filter's status in the *RSS* field of the Rx descriptor for packets that matches a Flow Director filter. Enabling the Flow Director filter's status, the RXCSUM.*PCSD* bit (Section 8.2.2.8.1)should be set as well (disabling the fragment checksum). ***Note:*** The Flow Director filter *Status* and *Error* bits in the *Extended Status* and *Error* fields in the Rx descriptor are always enabled.
RESERVED	6	0b	RSV	Reserved.
REPORT_STATUS_ALWAYS	7	0b	RW	**Report Status Always** Report Flow Director status in the *RSS* field of the Rx descriptor on any packet that can be a candidate for the Flow Director filters. This bit can be set to 1b only when both the RXCSUM.*PCSD* bit and the *REPORT_STATUS* bit in this register are set.
DROP_QUEUE	14:8	0x0	RW	**Drop Queue** Absolute Rx queue index used for the dropped packets. Software might set this queue to an empty one by setting the RDLEN[n] to 0x0.
DROP_NO_MATCH	15	0b	RW	**Drop No Match** If set, send to the *DROP_QUEUE* packets that were candidates for Flow Director and did not match any filters. Candidature of packets is defined in Table 7-5 on page 377.
FLEX_OFFSET	20:16	0x0	RW	**Flexible Offset** Offset within the first 64 bytes of the packet of a flexible 2-byte tuple. The offset is defined in word units counted from the first byte in the destination Ethernet MAC Address.

Field	Bit(s)	Init.	Type	Description
FILTERMODE	23:21	000b	RW	**Filter Mode** Defines the fields on which the Flow director acts: 000b = IPMODE — L3/L4 tuple. 001b = MACVLANMODE — Acts on the MAC and VLAN. 010b = Cloud: NVGRE — Based on TNI, inner MAC, inner VLAN. VXLAN — Based on VNI, inner MAC, inner VLAN. All other values are reserved.
MAX_LENGTH	27:24	0x0	RW	**Max Linked List Length** This field defines the maximum recommended linked list associated to any Hash value. Packets that match filters that exceed *MAX_LENGTH* are reported with an active *Length* bit in the *Extended Error* field. In addition, "Drop" filters that exceed the *MAX_LENGTH* are posted to the Rx queue defined in the filter context rather than the *DROP_QUEUE* defined in this register. *MAX_LENGTH* is defined in units of 2 filters, and exceeding it reflects the addition of two more filters. ***Note:*** Software should set this field to a value that indicates exceptional long buckets. Supporting 32K filters with good hash scheme key, it is expected that a value of 0xA might be a good choice.
FULL_THRESH	31:28	0x0	RW	**Full Threshold** A recommended minimum number of flows that should remain unused (defined in units of 16 filters). When software exceeds this threshold (too low number of unused flows), hardware generates the Flow Director Full interrupt. Software should avoid additional programming following this interrupt.

8.2.2.15.2 Flow Director Filters Source IPv6 - FDIRSIPV6[n] (0x0000EE0C + 0x4*n, n=0...2)

Field	Bit(s)	Init.	Type	Description
IP6SA	31:0	0x0	RW	**IPv6 Source Address** Three LS DWords of the source IPv6 address. The FDIRIPSA register contains the MS DWord of the IP6 address. The LS byte of the FDIRIPSA register is first on the wire and the MS byte of FDIRSIPv6[2] is last on the wire In MACVLANMODE, FDIRSIPv6_0[31:0] and FDIRSIPv6_1[15:0] hold the destination MAC Address value. FDIRSIPv6_1[31:16] and FDIRSIPv6_2[31:0] are reserved and should be set to 0x0.

8.2.2.15.3 Flow Director Filters IP SA - FDIRIPSA (0x0000EE18)

Field	Bit(s)	Init.	Type	Description
IP4SA	31:0	0x0	RW	**IPv4 Source Address** Source IPv4 address or MS DWord of the source IPv6 address, where the field is defined in Big Endian (LS byte is first on the wire). In MACVLANMODE, this field is reserved and should be set to 0x0.

8.2.2.15.4 Flow Director Filters IP DA - FDIRIPDA (0x0000EE1C)

Field	Bit(s)	Init.	Type	Description
IP4DA	31:0	0x0	RW	**IPv4 Destination Address** This field is defined in Big Endian (LS byte is first on the wire). In MACVLANMODE, this field is reserved and should be set to 0x0.

8.2.2.15.5 Flow Director Filters Port - FDIRPORT (0x0000EE20)

Field	Bit(s)	Init.	Type	Description
SOURCE	15:0	0x0	RW	**Source Port Number** This field is defined in Little Endian (MS byte is first on the wire). In MACVLANMODE and cloud modes (NVGRE/VXLAN), this field is reserved and should be set to 0x0.
DESTINATION	31:16	0x0	RW	**Destination Port Number** This field is defined in Little Endian (MS byte is first on the wire). In MACVLANMODE and cloud modes (NVGRE/VXLAN), this field is reserved and should be set to 0x0.

8.2.2.15.6 Flow Director Filters VLAN and FLEX Bytes - FDIRVLAN (0x0000EE24)

Field	Bit(s)	Init.	Type	Description
VLAN	15:0	0x0	RW	**VLAN Tag** This field is defined in Little Endian (MS byte is first on the wire). The *CFI* bit must be set to zero and it is not checked by hardware.
FLEX	31:16	0x0	RW	**Flexible** Flexible tuple data as defined by the *FLEX_OFFSET* field in the FDIRCTRL register, where the field is defined in Big Endian (LS byte is first on the wire).

8.2.2.15.7 Flow Director Filters Hash Signature - FDIRHASH (0x0000EE28)

Field	Bit(s)	Init.	Type	Description
HASH	14:0	0x0	RW	**Hash** Bucket hash value that identifies a filter's linked list
BUCKET_VALID	15	0b	RW	**Bucket Valid** The *Valid* bit is set by hardware whenever there is at least one filter assigned to this hash.
SIGNATURE_SW_INDEX	30:16	0x0	RW	**Signature/Software Index** Flow Director Filter Signature for Signature filters and SW-index for perfect match filters. In perfect match mode, the two MS bits should be zero.
RESERVED	31	0b	RSV	Reserved.

8.2.2.15.8 Flow Director Filters Command Register - FDIRCMD (0x0000EE2C)

Field	Bit(s)	Init.	Type	Description
CMD	1:0	0x0	RW	**Command** Flow Director filter programming command. 00b = No Action 01b = Add Flow 10b = Remove Flow 11b = Query Command Following a command completion, hardware clears the *CMD* field. In a query command, all other parameters are valid when the *CMD* field is zero.
FILTER_VALID	2	0b	RW	**Filter Valid** Valid filter is found by the query command. This bit is set by the device following a "Query Command" completion.
FILTER_UPDATE	3	0b	RW	**Filter Update** This bit is relevant only for "Add Flow" command, and must be set to zero in any other commands. 0b = The filter parameters do not override existing ones if exist while setting only the collision bit. 1b = The new filter parameters override existing ones if exist keeping the collision bit as is.
IPV6DMATCH	4	0b	RW	**IPv6 Destination Match** IP destination match to IP6AT filter. This bit is meaningful only for perfect match IPv6 filters. Otherwise it should be cleared by software at programming time. 0b = The Destination IPv6 address should not match the IP6AT. 1b = The Destination IPv6 address should match the IP6AT. This field might never match local VM to VM traffic.
L4TYPE	6:5	00b	RW	**L4 Packet Type** Defines the packet as one of the following L4 types: 00b = Reserved. 01b = UDP 10b = TCP 11b = SCTP
IPV6	7	0b	RW	**IPV6** 0b = IPv4 packet type. 1b = IPv6 packet type. Relevant only if FDIRM.*L3P* is not set.
CLEARHT	8	0b	RW	**Clear Flow Director Head and Tail Registers** This bit is set only as part of Flow Director initialization. During nominal operation it must be kept at 0b.
DROP	9	0b	RW	**Packet Drop Action** Receive packets that match a filter with active *DROP* bit and do not exceed the maximum recommended linked list length defined in FDIRCTRL.*MAX_LENGTH* field are posted to the global queue defined by FDIRCTRL.*DROP_QUEUE*. Receive packets that match a filter with active *DROP* bit and exceeds the maximum recommended linked list length defined in FDIRCTRL.*MAX_LENGTH* field are posted to the queue defined by *RX_QUEUE* field in this register. The receive descriptor of such packets is reported with active FDIRErr(0) flag indicating that the *MAX_LENGTH* was exceeded. ***Note:*** As in some cases, packets matching this filter may still be forwarded, even when the *DROP* bit is set. The *QUEUE_EN* flag must be set, and *RX_QUEUE* in this register must be point to a valid queue. Otherwise, the result is unexpected. The *DROP* flag is useful only for Perfect Match mode and it should not be set in Signature mode.
RESERVED	10	0b	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
LAST	11	0b	RW	**Last** Last filter indication in the linked list. At Flow programming, the software should set the *LAST* bit to 1b. Hardware might modify this bit when adding or removing flows from the same linked list.
COLLISION	12	0b	RW	**Collision Indication** This field is set to 1b when software programs the same filter multiple times. In Signature-based filtering, it is set when software programs a filter with the same Hash and Signature multiple times. It should be cleared by software when it adds a Flow. It might be set by hardware when two flows collide with the same Hash and Signature. During reception, this bit is reported on the Rx descriptor of packets that match the filter.
RESERVED	14:13	00b	RSV	Reserved.
QUEUE_EN	15	0b	RW	**Queue Enable** Enable routing matched packet to the queue defined by the *RX_QUEUE*. ***Note:*** Packets redirection to the FDIRCTRL.*DROP_QUEUE* is not gated by the *QUEUE_EN* bit.
RX_QUEUE	22:16	0x0	RW	**Rx Queue Index** This field defines the absolute Rx queue index in all modes of operation (regardless of DCB and VT enablement).
TUNNEL_FILTER	23	0b	RSV	**Tunnel Filter** If set, the filter is matched by NVGRE or VXLAN packets only. In this case, the L3 and L4 parameters programmed should be of the tunneled (inner) header. This bit is valid, and should be set, only in IP filtering mode (FDIRCTRL.*FILTERMODE* = 000b)
POOL	29:24	0x0	RW	**Pool Number** Meaningful when VT mode is enabled. When VT is not enabled, this field must be set to zero by software.
RESERVED	30	0b	RSV	Reserved.
FD_EXCEPTION	31	0b	RW	**Flow Director Exception** If set after a command was given, indicates that a Flow Director exception occurred. This bit is a reflection of the EICR.*FLOW_DIRECTOR* bit (Section 8.2.2.6.1). Should be set to zero when a new command is given.

8.2.2.15.9 Flow Director Filters Free - FDIRFREE (0x0000EE38)

Field	Bit(s)	Init.	Type	Description
FREE	15:0	0x0	RW	**Free** Number of free (non programmed) filters in the Flow Director filters logic.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.15.10 Flow Director Filters DIPv4 Mask - FDIRDIP4M (0x0000EE3C)

Field	Bit(s)	Init.	Type	Description
IPM	31:0	0x0	RW	**Mask Destination IPv4 address** Each cleared bit means that the associated bit of the destination IPv4 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the destination IPv4 address is ignored (masked out). The LS bit of this register matches the first byte on the wire.

8.2.2.15.11 Flow Director Filters Source IPv4 Mask - FDIRSIP4M (0x0000EE40)

Field	Bit(s)	Init.	Type	Description
IPM	31:0	0x0	RW	**Mask Source IPv4 address** Each cleared bit means that the associated bit of the source IPv4 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the source IPv4 address is ignored (masked out).The LS bit of this register matches the first byte on the wire.

8.2.2.15.12 Flow Director Filters TCP Mask - FDIRTCPM (0x0000EE44)

Field	Bit(s)	Init.	Type	Description
SPORTM	15:0	0x0	RW	**Mask TCP Source Port** Each cleared bit means that the associated bit of the TCP source port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the TCP source port is ignored (masked out). This field is laid out as follows: • Bit[0] in the mask affects Bit[15] of the source port as defined in FDIRPORT.*SOURCE* (Section 8.2.2.15.5). • Bit[1] in the mask affects Bit[14] in FDIRPORT.*SOURCE.* • ...and so on. • Bit[15] in the mask affects Bit[0] in FDIRPORT.*SOURCE*.
DPORTM	31:16	0x0	RW	**Mask TCP Destination Port** Each cleared bit means that the associated bit of the TCP destination port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the TCP destination Port is ignored (masked out). This field is laid out as follows: • Bit[0] in the mask affects Bit[15] of the source port as defined in FDIRPORT.*DESTINATION* (Section 8.2.2.15.5). • Bit[1] in the mask affects Bit[14] in FDIRPORT.*DESTINATION.* • ...and so on. • Bit[15] in the mask affects Bit[0] in FDIRPORT.*DESTINATION*.

8.2.2.15.13 Flow Director Filters UDP Mask - FDIRUDPM (0x0000EE48)

Field	Bit(s)	Init.	Type	Description
SPORTM	15:0	0x0	RW	**Mask UDP Source Port** Each cleared bit means that the associated bit of the UDP source port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the UDP source port is ignored (masked out). This field is laid out as follows: • Bit[0] in the mask affects Bit[15] of the source port as defined in FDIRPORT.*SOURCE* (Section 8.2.2.15.5). • Bit[1] in the mask affects Bit[14] in FDIRPORT.*SOURCE.* • ...and so on. • Bit[15] in the mask affects Bit[0] in FDIRPORT.*SOURCE*.

Field	Bit(s)	Init.	Type	Description
DPORTM	31:16	0x0	RW	**Mask UDP Destination Port** Each cleared bit means that the associated bit of the UDP destination port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the UDP destination port is ignored (masked out). This field is laid out as follows: • Bit[0] in the mask affects Bit[15] of the source port as defined in FDIRPORT.*DESTINATION* (Section 8.2.2.15.5). • Bit[1] in the mask affects Bit[14] in FDIRPORT.*DESTINATION.* • ...and so on. • Bit[15] in the mask affects Bit[0] in FDIRPORT.*DESTINATION*.

8.2.2.15.14 Flow Director Filters Length - FDIRLEN (0x0000EE4C)

Field	Bit(s)	Init.	Type	Description
MAXLEN	5:0	0x0	RC	**Maximum Length** Longest linked list of filters in the table. This field records the length of the longest linked list that is updated since the last time this register was read by the software. The longest bucket reported by this field includes *MAXLEN* + 1 filters.
RESERVED	7:6	00b	RSV	Reserved.
BUCKET_LENGTH	13:8	0x0	RC	**Bucket Length** The length of the linked list indicated by a query command. This field is valid following a query command completion.
RESERVED	15:14	00b	RSV	Reserved.
MAXHASH	30:16	0x0	RC	**Maximum Hash** The Lookup HASH value of the added filter that updated the value of the *MAXLEN* field in this register.
RESERVED	31	0b	RSV	Reserved.

8.2.2.15.15 Flow Director Filters Usage Statistics - FDIRUSTAT (0x0000EE50)

Field	Bit(s)	Init.	Type	Description
ADD	15:0	0x0	RC	**Added** Number of added filters. This field counts the number of added filters to the Flow Director filters logic. The counter is stuck at 0xFFFF and cleared on read.
REMOVE	31:16	0x0	RC	**Removed** Number of removed filters. This field counts the number of removed filters to the Flow Director filters logic. The counter is stuck at 0xFFFF and cleared on read.

8.2.2.15.16 Flow Director Filters Failed Usage Statistics - FDIRFSTAT (0x0000EE54)

Field	Bit(s)	Init.	Type	Description
FADD	7:0	0x0	RC	**Failed Added** Number of failed added filters due to no space in the filter table. The counter is stuck at 0xFF and cleared on read.
FREMOVE	15:8	0x0	RC	**Failed Removed** Number of failed removed filters. The counter is stuck at 0xFF and cleared on read.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.15.17 Flow Director Filters Match Statistics - FDIRMATCH (0x0000EE58)

Field	Bit(s)	Init.	Type	Description
PCNT	31:0	0x0	RC	**Packet Count** Number of packets that matched any Flow Director filter. The counter is stacked at 0xFF...F and cleared on read. ***Note:*** This counter might include packets that match the L2 filters, 5 tuple filters, or Syn filters, even if they are enabled for queue assignment.

8.2.2.15.18 Flow Director Filters Miss Match Statistics - FDIRMISS (0x0000EE5C)

Field	Bit(s)	Init.	Type	Description
PCNT	31:0	0x0	RC	**Packet Count** Number of packets that miss-matched any Flow Director filter. The counter is stacked at 0xFF...F and cleared on read.

8.2.2.15.19 Flow Director Filters Lookup Table HASH Key - FDIRHKEY (0x0000EE68)

Field	Bit(s)	Init.	Type	Description
KEY	31:0	0x80000001	RW	**Key** Programmable hash lookup table key.

8.2.2.15.20 Flow Director Filters Signature HASH Key - FDIRSKEY (0x0000EE6C)

Field	Bit(s)	Init.	Type	Description
KEY	31:0	0x80800101	RW	**Key** Programmable signature key.

8.2.2.15.21 Flow Director Filters Other Mask - FDIRM (0x0000EE70)

Field	Bit(s)	Init.	Type	Description
VLANID	0	0b	RW	**Mask VLAN ID Tag** When cleared, the 12 bits of the VLAN ID tag are meaningful for the filtering functionality.
VLANP	1	0b	RW	**Mask VLAN Priority Tag** When cleared, the 3 bits of the VLAN Priority are meaningful for the filtering functionality.
POOL	2	0b	RW	**Mask Pool** When cleared, the target Pool number is meaningful for the filtering functionality.
L4P	3	0b	RW	**Mask L4 Protocol** When cleared, the UDP/TCP/SCTP protocol type is meaningful for the filtering functionality. ***Note:*** For the Flow Director filtering aspects, SCTP is treated as if it is TCP.

Field	Bit(s)	Init.	Type	Description
FLEX	4	0b	RW	**Mask Flexible Tuple** When cleared, the 2 bytes of the Flexible Tuple are meaningful for the filtering functionality.
DIPV6	5	0b	RW	**Mask Destination IPv6** When cleared, the compare against the IP6AT filter is meaningful for IPv6 packets.
L3P	6	0b	RW	**Mask IP Type Comparison** To block all IP comparison, the following should also be set: *DIPV6* field, FDIRIP6M register, FDIRSIP4M register, and FDIRDIP4M register (see Section 8.2.2.15.22, Section 8.2.2.15.11, and Section 8.2.2.15.10, respectively.
RESERVED	31:7	0x0	RSV	Reserved.

8.2.2.15.22 Flow Director Filters IPv6 Mask - FDIRIP6M (0x0000EE74)

Field	Bit(s)	Init.	Type	Description
SIPM	15:0	0x0	RW	**Mask Source IPv6 Address** Each cleared bit means that the associated byte of the source IPv6 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated byte of the source IPv6 address is ignored (masked out). The LS bit of this register matches the first byte on the wire. When working in MACVLAN mode, this field defines the mask to be used to compare the MAC Address When working in VXLAN or NVGRE mode, should be used to allow comparison of TNI/VNI and inner MAC Address:.
DIPM	31:16	0x0	RW	**Mask Destination IPv6 Address** Each cleared bit means that the associated byte of the destination IPv6 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated byte of the destination IPv6 address is ignored (masked out). The whole field is meaningful only for the Hash function and the Signature based filters. The DIPv6 bit in the FDIRM register is meaningful for perfect match filters. The LS bit of this register matches the first byte on the wire.

8.2.2.15.23 Flow Director Filters SCTP Mask - FDIRSCTPM (0x0000EE78)

Field	Bit(s)	Init.	Type	Description
SPORTM	15:0	0x0	RW	**Mask SCTP Source Port** Each cleared bit means that the associated bit of the SCTP source port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the SCTP source port is ignored (masked out). This field is laid out as follows: • Bit[0] in the mask affects Bit[15] of the source port as defined in FDIRPORT.*SOURCE* (Section 8.2.2.15.5). • Bit[1] in the mask affects Bit[14] in FDIRPORT.*SOURCE.* • ...and so on. • Bit[15] in the mask affects Bit[0] in FDIRPORT.*SOURCE*.
DPORTM	31:16	0x0	RW	**Mask SCTP Destination Port** Each cleared bit means that the associated bit of the SCTP destination port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the SCTP destination port is ignored (masked out). This field is laid out as follows: • Bit[0] in the mask affects Bit[15] of the source port as defined in FDIRPORT.*DESTINATION* (Section 8.2.2.15.5). • Bit[1] in the mask affects Bit[14] in FDIRPORT.*DESTINATION.* • ...and so on. • Bit[15] in the mask affects Bit[0] in FDIRPORT.*DESTINATION*.

8.2.2.16 PF - MAC Registers

8.2.2.16.1 Highlander Control 0 Register - HLREG0 (0x00004240)

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	11b	RSV	Reserved.
JUMBOEN	2	0b	RW	**Jumbo Frame Enable** Allows frames up to the size specified in Bits[31:16] of the MAXFRS register (Section 8.2.2.16.6). 0b = Disable jumbo frames (default). 1b = Enable jumbo frames.
RESERVED	9:3	0x7F	RSV	Reserved. Must be set to 0x7F (1111111b).
TXPADEN	10	1b	RW	**Tx Pad Frame Enable** Pad short Tx frames to 64 bytes if requested by user. 0b = Transmit short frames with no padding. 1b = Pad frames (default).
RESERVED	14:11	0x5	RSV	Reserved. Must be set to 0x5 (0101b).
LPBK	15	0b	RW	**Loopback** Turn on loopback where transmit data is sent back through receiver. 0b = Loopback disabled (default). 1b = Loopback enabled.
MDCSPD	16	1b	RW	**MDC Speed** High or low speed MDC clock frequency to PCS, XGXS, WIS, etc. MDCSPD — Freq at 10 GbE — Freq at 1 GbE — Freq at 100 Mb/s 0b — 2.4 MHz — 240 KHz — 240 KHz 1b — 24 MHz — 2.4 MHz — 240 KHz
CONTMDC	17	0b	RW	**Continuous MDC** Turn off MDC between MDIO packets. 0b = MDC off between packets (default). 1b = Continuous MDC.
RESERVED	19:18	00b	RSV	Reserved.
PREPEND	23:20	0x0	RW	**Prepend Value** Number of 32-bit words, starting after the preamble and SFD, to exclude from the CRC generator and checker (default = 0x0).
RESERVED	26:24	000b	RSV	Reserved.
RXLNGTHERREN	27	1b	RW	**Rx Length Error Reporting** 0b = Disable reporting of all rx_length_err events. 1b = Enable reporting of rx_length_err events if length field < 0x0600.
RXPADSTRIPEN	28	0b	RW	**Rx Padding Strip Enable** 0b = Do not strip padding from Rx packets with length field < 64 (default). 1b = Strip padding from Rx packets with length field < 64 (debug only). ***Note:*** This functionality should be used as debug mode only. If Rx Pad Stripping is enabled, the Rx CRC Stripping needs to be enabled as well.
RESERVED	31:29	000b	RSV	Reserved.

8.2.2.16.2 Highlander Status 1 Register - HLREG1 (0x00004244)

Field	Bit(s)	Init.	Type	Description
RESERVED	4:0	0x1	RSV	Reserved.
RXERRSYM	5	0b	RO	**Rx Error Symbol** Error symbol during Rx packet (latch high, clear on read). 0b = No error symbol (default). 1b = Error symbol received.
RXILLSYM	6	0b	RO	**Rx Illegal Symbol** Illegal symbol during Rx packet (latch high, clear on read). 0b = No illegal symbol received (default). 1b = Illegal symbol received.
RXIDLERR	7	0b	RO	**Rx Idle Error** Non-idle symbol during idle period (latch high, clear on read). 0b = No idle errors received (default). 1b = Idle error received.
RXLCLFLT	8	0b	RO	**Rx Local Fault** Fault reported from PMD, PMA, or PCS (latch high, clear on read). 0b = No local fault (default). 1b = Local fault is or was active.
RXRMTFLT	9	0b	RO	**Rx Remote Fault** Link partner reported remote fault (latch high, clear on read). 0b = No remote fault (default). 1b = Remote fault is or was active.
RESERVED	31:10	0x0	RSV	Reserved.

8.2.2.16.3 Pause and Pace Register - PAP (0x00004248)

Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0xFFFF	RSV	Reserved.
PACE	19:16	0x0	RW	**Pace** 0000b = 10 Gb/s (LAN) 0001b = 1 Gb/s 0010b = 2 Gb/s 0011b = 3 Gb/s 0100b = 4 Gb/s 0101b = 5 Gb/s 0110b = 6 Gb/s 0111b = 7 Gb/s 1000b = 8 Gb/s 1001b = 9 Gb/s 1010b = Reserved 1011b = Reserved 1100b = Reserved 1101b = Reserved 1110b = Reserved 1111b = 9.294196 Gb/s (WAN)
RESERVED	31:20	0x0	RSV	Reserved.

8.2.2.16.4 MDI Single Command and Address - MSCA (0x0000425C)

Field	Bit(s)	Init.	Type	Description
MDIADD	15:0	0x0	RW	**MDI Address** Address used for MDI accesses (default = 0x0000).
DEVADD	20:16	0x0	RW	**Device Address** Five bits representing device address.
PORTADD	25:21	0x0	RW	**Port Address** The address of the PHY port. This field must be set to the port number as reflected in the STATUS.*LAN_ID* field (Section 8.2.2.1.2).
OPCODE	27:26	00b	RW	**Op Code** Two bits identifying operation to be performed (default - 00b). 00b = Address cycle. 01b = Write operation. 10b = Read, increment address. 11b = Read operation.
RESERVED	29:28	00b	RSV	Reserved. Reads as 00b. Must be written as 00b.
MDICMD	30	0b	RW	**MDI Command** Perform the MDI operation in this register. Cleared when done. 0b = MDI ready / Operation complete (default). 1b = Perform operation / Operation in progress.
RESERVED	31	0b	RSV	Reserved.

8.2.2.16.5 MDI Single Read and Write Data - MSRWD (0x00004260)

Field	Bit(s)	Init.	Type	Description
MDIWRDATA	15:0	0x0	RW	**MDI Write Data** Write data.
MDIRDDATA	31:16	0x0	RW	**MDI Read Data** Read data (RO).

8.2.2.16.6 Max Frame Size - MAXFRS (0x00004268)

Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0x0	RSV	Reserved.
MFS	31:16	0x5EE	RW	**Maximum Frame Size** This field defines the maximum frame size (MFS) in bytes units from Ethernet MAC Addresses up to and including the CRC. Frames received that are larger than this value are dropped. This field is meaningful when jumbo frames are enabled (HLREG0.*JUMBOEN* = 1b). When Jumbo frames are not enabled the device uses a hard-wired value of 1518 for this field. The MFS does not include the 4 bytes of the VLAN header. Packets with VLAN header might be as large as *MFS* + 4. When Double VLAN is enabled the device adds 8 to the MFS for any packet. When E-tags are enabled, the device adds 12 to the MFS for any packet. This value has no effect on transmit frames. It is the responsibility of software to limit the size of transmit frames. ***Note:*** Packets to/from MC are limited to 2 KB even when jumbo frames are enabled.

8.2.2.16.7 Link Status Register - LINKS (0x000042A4)

Field	Bit(s)	Init.	Type	Description
FIFO_UNDERRUN	0	0b	RO	**FIFO Underrun** Indicates underrun condition in MAC elastic FIFO.
FIFO_OVERRUN	1	0b	RO	**FIFO Overrun** Indicates overrun condition in MAC elastic FIFO.
RF_STATE	2	0b	RO	**Remote Fault State** MAC is in Remote Fault state.
LF_STATE	3	0b	RO	**Local Fault State** MAC is in Local Fault state
RESERVED	6:4	000b	RSV	Reserved.
LINK_STATUS	7	0b	RO	**Link Status** 0b = Link is currently down, or link was down since last time read. 1b = Link is up, and there was no link down from last time read. Self-cleared upon read if the link is low, and set if the link is up.
RESERVED	26:8	0x0	RSV	Reserved.
NON_STANDARD_SPEED	27	0b	RO	**Non-Standard Speed** If set, the *LINK_SPEED* field reflects non standard speeds (2.5/5 GbE)
LINK_SPEED	29:28	00b	RO	**Link Speed** MAC link speed status. If NON_STANDARD_SPEED = 0: 00b = Reserved. 01b = 100 Mb/s 10b = 1 GbE 11b = 10 GbE If NON_STANDARD_SPEED = 1: 00b = Reserved. 01b = 5 GbE 10b = Reserved. 11b = 2.5 GbE
LINK_UP	30	0b	RO	**Link Up** 0b = Link is down. 1b = Link is up.
RESERVED	31	0b	RSV	Reserved.

8.2.2.16.8 MAC Manageability Control Register - MMNGC (0x000042D0)

Field	Bit(s)	Init.	Type	Description
MNG_VETO	0	0b	RO	**Management Veto** *MNG_VETO* (default 0) access read/write by the management controller, read only to the host. 0b = No specific constraints on link from management controller. 1b = Hold off any low-power link mode changes. This is done to avoid link loss and interrupting management traffic/activity.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.16.9 MAC Control Register - MACC (0x00004330)

Field	Bit(s)	Init.	Type	Description
FLU	0	0b	RW	**Force Link Up** 0b = Normal mode. 1b = MAC is forced to the link up state regardless to the PHY link status.
MAC_RX2TX_LPBK_EN	1	0b	RW	**MAC Rx to Tx Loopback Enable** 0b = 0No loopback, normal mode. 1b = Loopback enabled. Transmit path is driven from the receive path, at the MAC internal XGMII interface.
SWIZZLE_TX_DATA	2	0b	RW	**Swizzle Tx Data** Swizzle the bytes in all four MAC internal Tx XGMII lanes. 0b = Swizzle disabled, normal mode. 1b = Swizzle enabled.
SWAP_TX_DATA	3	0b	RW	**Swap Tx Data** Swap the MAC internal Tx XGMII lanes. 0b = Swap disabled, normal mode. 1b = Swap enabled.
SWAP_TX_CONTROL	4	0b	RW	**Swap Tx Control** Swap the MAC internal Tx XGMII controls. 0b = Swap disabled, normal mode. 1b = Swap enabled.
SWIZZLE_RX_DATA	5	0b	RW	**Swizzle Rx Data** Swizzle the bytes in all four MAC internal Rx XGMII lanes. 0b = Swizzle disabled, normal mode. 1b = Swizzle enabled.
SWAP_RX_DATA	6	0b	RW	**Swap Rx Data** Swap the MAC internal Rx XGMII lanes. 0b = Swap disabled, normal mode. 1b = Swap enabled.
SWAP_RX_CONTROL	7	0b	RW	**Swap Rx Control** Swap the MAC internal Rx XGMII controls. 0b = Swap disabled, normal mode. 1b = Swap enabled.
FIFOHT	11:8	0xD	RW	**FIFO High Threshold** Determines the high threshold of the MAC elastic FIFO.
FIFOLT	15:12	0x3	RW	**FIFO Low Threshold** Determines the low threshold of the MAC elastic FIFO.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.17 PF - Statistics Registers

All Statistics registers are cleared on read. In addition, they stick at 0xFF...F when the maximum value is reached.

For the receive statistics, a packet is indicated as received if it passes the device filters and is placed into the packet buffer memory. A packet does not have to be DMA'd to host memory to be counted as received.

Due to divergent paths between interrupt generation and logging of relevant statistics counts, it might be possible to generate an interrupt to the system for a noteworthy event prior to the associated statistics count actually being incremented. This is extremely unlikely due to expected delays associated with the system interrupt collection and ISR delay, but might be an explanation for interrupt statistics values that do not quite make sense. Hardware guarantees that any event noteworthy of inclusion in a statistics count is reflected in the appropriate count within 1 μs; a small time-delay prior to reading the statistics might be required to avoid a potential mismatch between and interrupt and its cause.

If RSC is enabled, statistics are collected before RSC is applied to the packets. If TSO is enabled, statistics are collected after segmentation.

All byte (octet) counters composed of two registers can be fetched by two consecutive 32-bit accesses while reading the low 32-bit register first, or a single 64-bit access.

All receive statistic counters count the packets and bytes before coalescing by the RSC logic or FCoE DDP logic. All receive statistic counters in the filter unit might count packets that might be dropped by the packet buffer or receive DMA. Same comment is valid for the byte counters associated with these packet counters: PRC64, PRC127, PRC255, PRC511, PRC1023, PRC1522, BPRC, MPRC, GPRC, RXNFGPC, RUC, and ROC.

8.2.2.17.1 Queue Packets Received Count - QPRC[n] (0x00001030 + 0x40*n, n=0...15)

Field	Bit(s)	Init.	Type	Description
PRC	31:0	0x0	RC	**Packets Received Count** Number of packets received for the Queue. FCoE packets are counted in the QRPC even if they are posted only to the DDP queue (with no traces in the legacy queue).

8.2.2.17.2 Queue Bytes Received Count Low - QBRC_L[n] (0x00001034 + 0x40*n, n=0...15)

Field	Bit(s)	Init.	Type	Description
BRC_L	31:0	0x0	RC	**Bytes Received Count Low** Lower 32 bits of the statistic counter. The QBRC_L[n] and QBRC_H[n] registers make up a logical 36-bit counter of received bytes that were posted to the programmed Rx queues of the packets counted by the QPRC[n]. The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP.

8.2.2.17.3 Queue Bytes Received Count High - QBRC_H[n] (0x00001038 + 0x40*n, n=0...15)

Field	Bit(s)	Init.	Type	Description
BRC_H	3:0	0x0	RC	**Bytes Received Count High** Higher 4 bits of the statistic counter described in QBRC_L (Section 8.2.2.17.2).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.4 Queue Packets Received Drop Count - QPRDC[n] (0x00001430 + 0x40*n, n=0...15)

Field	Bit(s)	Init.	Type	Description
PRDC	31:0	0x0	RC	**Packets Received Drop Count** Number of receive packets dropped for the queue. Packets are dropped per queue in one of three cases: • Rx Queue is disabled in the RXDCTL[n] register (this includes packets sent to this queue due to the flow director drop no match mechanism or drop action). • No free descriptors in the Rx queue while hardware is set to *DROP_EN* in the SRRCTL[n] register or in the PFQDE register. • Packet size is larger than RLPML while *RLPML_EN* is set in the RXDCTL[n] register.

8.2.2.17.5 Receive Queue Statistic Mapping Registers - RQSMR[n] (0x00002300 + 0x4*n, n=0...31)

These registers define the mapping of the receive queues to the per-queue statistics. Several queues can be mapped to a single statistic register. Each statistic register counts the number of packets and bytes of all the queues that are mapped to that statistics. The registers counting Rx queue statistics are: QPRC, QBRC, and QPRDC. For example, setting RQSMR[0].*Q_MAP*[0] to "3" maps Rx queue 0 to the counters QPRC[3], QBRC[3], and QPRDC[3]. Setting RQSMR[2].*Q_MAP*[1] to "5" maps Rx queue 9 to the QPRC[5], QBRC[5], and QPRDC[5].

Field	Bit(s)	Init.	Type	Description
Q_MAP_0	3:0	0x0	RW	**Queue Map 0** For each register 'n', *Q_MAP*[0] defines the per queue statistic registers that are mapped to Rx queue '4*n+0'.
RESERVED	7:4	0x0	RSV	Reserved.
Q_MAP_1	11:8	0x0	RW	**Queue Map 1** For each register 'n', *Q_MAP*[1] defines the per queue statistic registers that are mapped to Rx queue '4*n+1'.
RESERVED	15:12	0x0	RSV	Reserved.
Q_MAP_2	19:16	0x0	RW	**Queue Map 2** For each register 'n', *Q_MAP*[2] defines the per queue statistic registers that are mapped to Rx queue '4*n+2'.
RESERVED	23:20	0x0	RSV	Reserved.
Q_MAP_3	27:24	0x0	RW	**Queue Map 3** For each register 'n', *Q_MAP*[3] defines the per queue statistic registers that are mapped to Rx queue '4*n+3'.
RESERVED	31:28	0x0	RSV	Reserved.

8.2.2.17.6 FCoE Rx Packets Dropped Count - FCOERPDC (0x0000241C)

Field	Bit(s)	Init.	Type	Description
RPDC	31:0	0x0	RC	**Received Packets Dropped Count** Number of Rx packets dropped due to lack of descriptors.

8.2.2.17.7 Fiber Channel Last Error Count - FCLAST (0x00002424)

Field	Bit(s)	Init.	Type	Description
LAST_CNT	15:0	0x0	RC	**Last Count** Number of packets received to valid FCoE contexts while their user buffers are exhausted.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.17.8 FCoE Packets Received Count - FCOEPRC (0x00002428)

Field	Bit(s)	Init.	Type	Description
PRC	31:0	0x0	RC	**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.

8.2.2.17.9 FCOE DWord Received Count - FCOEDWRC (0x0000242C)

Field	Bit(s)	Init.	Type	Description
DWRC	31:0	0x0	RC	**DWord Received Count** Number of DWords count in good received packets with no Ethernet CRC nor FC CRC errors and that passed successfully DDP. The counter relates to FCoE packets starting at the FC header up to and including the FC CRC (it excludes the Ethernet encapsulation).

8.2.2.17.10 Rx DMA Statistic Counter Control - RXDSTATCTRL (0x00002F40)

Field	Bit(s)	Init.	Type	Description
QSEL	4:0	0x0	RW	**Queue Select** Controls which Rx queues are considered for the DMA Good Rx and DMA Duplicated counters as follows: 00000b...01111b = The counters relates to the same queues that are directed to the QPRC[QSEL] counter as defined by the RQSMR[n] registers. 10000b = The counters relates to all Rx queues. Else = Reserved.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.2.17.11 DMA Good Rx Packet Counter - RXDGPC (0x00002F50)

Field	Bit(s)	Init.	Type	Description
GPC	31:0	0x0	RC	**Good Packet Counter** Number of good (non-erred) Rx packets from the Network posted to the host memory. In case of packet replication (or mirrored) the counter counts each packet only once. ***Note:*** The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register (Section 8.2.2.17.10).

8.2.2.17.12 DMA Good Rx Byte Counter Low - RXDGBCL (0x00002F54)

Field	Bit(s)	Init.	Type	Description
GBCL	31:0	0x0	RC	**Good Byte Counter Low** Low 32 bits of the 36-bit byte counter of good (non-erred) Rx packets that match the RXDGPC register (Section 8.2.2.17.11). The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP.

8.2.2.17.13 DMA Good Rx Byte Counter High - RXDGBCH (0x00002F58)

Field	Bit(s)	Init.	Type	Description
GBCH	3:0	0x0	RC	**Good Byte Counter High** High 4 bits of the 36-bit byte counter associated with the RXDGBCL register (Section 8.2.2.17.12).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.14 DMA Duplicated Good Rx Packet Counter - RXDDPC (0x00002F5C)

Field	Bit(s)	Init.	Type	Description
GPC	31:0	0x0	RC	**Good Packet Counter** Number of replicated or mirrored packets that meet the RXDGPC conditions. The sum of RXDDPC and RXDGPC is the total good (non-erred) Rx packets from the Network that are posted to the host. ***Note:*** The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register (Section 8.2.2.17.10).

8.2.2.17.15 DMA Duplicated Good Rx Byte Counter Low - RXDDBCL (0x00002F60)

Field	Bit(s)	Init.	Type	Description
GBCL	31:0	0x0	RC	**Good Byte Counter Low** Low 32 bits of the 36-bit byte counter of good (non-erred) Rx packets that match the RXDDPC register (Section 8.2.2.17.14). The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP.

8.2.2.17.16 DMA Duplicated Good Rx Byte Counter High - RXDDBCH (0x00002F64)

Field	Bit(s)	Init.	Type	Description
GBCH	3:0	0x0	RC	**Good Byte Counter High** High 4 bits of the 36-bit byte counter associated with the RXDDBCL register (Section 8.2.2.17.15).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.17 DMA Good Rx LPBK Packet Counter - RXLPBKPC (0x00002F68)

Field	Bit(s)	Init.	Type	Description
GPC	31:0	0x0	RC	**Good Packet Counter** Number of good (non-erred) Rx packets from a local VM posted to the host memory. In case of packet replication (or mirrored) the counter counts each packet only once. ***Note:*** The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register (Section 8.2.2.17.10). The counter is not affected by RSC and FCoE DDP, since both functions are not supported for LPBK traffic.

8.2.2.17.18 DMA Good Rx LPBK Byte Counter Low - RXLPBKBCL (0x00002F6C)

Field	Bit(s)	Init.	Type	Description
GBCL	31:0	0x0	RC	**Good Byte Counter Low** Low 32 bits of the 36-bit byte counter of good (non-erred) Rx packets that match the RXLPBKPC register (Section 8.2.2.17.17). The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP.

8.2.2.17.19 DMA Good Rx LPBK Byte Counter High - RXLPBKBCH (0x00002F70)

Field	Bit(s)	Init.	Type	Description
GBCH	3:0	0x0	RC	**Good Byte Counter High** High 4 bits of the 36-bit byte counter associated with the RXLPBKBCL register (Section 8.2.2.17.18).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.20 DMA Duplicated Good Rx LPBK Packet Counter - RXDLPBKPC (0x00002F74)

Field	Bit(s)	Init.	Type	Description
GPC	31:0	0x0	RC	**Good Packet Counter** Number of replicated or mirrored packets that meet the RXLPBKPC conditions (Section 8.2.2.17.17). The sum of RXDLPBKPC and RXLPBKPC is the total good (non-erred) Rx packets from a local VM posted to the host. ***Note:*** The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register (Section 8.2.2.17.10).

8.2.2.17.21 DMA Duplicated Good Rx LPBK Byte Counter Low - RXDLPBKBCL (0x00002F78)

Field	Bit(s)	Init.	Type	Description
GBCL	31:0	0x0	RC	**Good Byte Counter Low** Low 32 bits of the 36-bit byte counter of good (non-erred) Rx packets that match the RXDLPBKPC (Section 8.2.2.17.20). The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP.

8.2.2.17.22 DMA Duplicated Good Rx LPBK Byte Counter High - RXDLPBKBCH (0x00002F7C)

Field	Bit(s)	Init.	Type	Description
GBCH	3:0	0x0	RC	**Good Byte Counter High** High 4 bits of the 36-bit byte counter associated to RXDLPBKBCL (Section 8.2.2.17.21).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.23 BMC2OS Packets Received by Host - B2OGPRC (0x00002F90)

Field	Bit(s)	Init.	Type	Description
B2OGPRC	31:0	0x0	RC	**BMC-to-OS Good Packets Received Counter** Counts the total number of packets originating from the BMC that reached the host. When the internal switch is enabled, each replication of a BMC to host packet is counted. The counter is cleared when read by driver. The counter is also cleared by PCIe reset and software reset. When reaching maximum value, counter does not wrap-around.

8.2.2.17.24 CRC Error Count - CRCERRS (0x00004000)

Field	Bit(s)	Init.	Type	Description
CEC	31:0	0x0	RC	**CRC Error Count** Counts the number of receive packets with CRC errors. For a packet to be counted in this register, it must be 64 bytes or greater (from <Destination Address> through <CRC>, inclusively) in length. This registers counts all packets received, regardless of L2 filtering and receive enable. This register does not count packets with bad SFD, error control byte, or illegal code byte.

8.2.2.17.25 Illegal Byte Error Count - ILLERRC (0x00004004)

Field	Bit(s)	Init.	Type	Description
IBEC	31:0	0x0	RC	**Illegal Byte Error Count** Counts the number of receive packets with illegal bytes errors (i.e., there is an illegal symbol in the packet). This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.2.17.26 Error Byte Packet Count - ERRBC (0x00004008)

Field	Bit(s)	Init.	Type	Description
EBC	31:0	0x0	RC	**Error Byte Count** Counts the number of receive packets with Error bytes (i.e., there is an error symbol in the packet). This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.2.17.27 MAC Short Packet Discard Count - MSPDC (0x00004010)

Field	Bit(s)	Init.	Type	Description
MSPDC	31:0	0x0	RC	**MAC Short Packet Discard Count** Counts the number of MAC short Packet Discard packets received.

8.2.2.17.28 Bad SFD Count - MBSDC (0x00004018)

Field	Bit(s)	Init.	Type	Description
MBSDC	31:0	0x0	RW	**MAC Bad SFD Count** Counts the number of packets received with bad SFD. Does not count short packets not received

8.2.2.17.29 MAC Local Fault Count - MLFC (0x00004034)

Field	Bit(s)	Init.	Type	Description
MLFC	31:0	0x0	RC	**MAC Local Fault Count** Counts the number of faults in the local MAC. This register is valid only when the link speed is 10 GbE.

8.2.2.17.30 MAC Remote Fault Count - MRFC (0x00004038)

Field	Bit(s)	Init.	Type	Description
MRFC	31:0	0x0	RC	**MAC Remote Fault Count** Counts the number of faults in the remote MAC. This register is valid only when the link speed is 10GbE.

8.2.2.17.31 Receive Length Error Count - RLEC (0x00004040)

Field	Bit(s)	Init.	Type	Description
RLEC	31:0	0x0	RC	**Receive Length Error Count** Counts the 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. To enable the receive length error count, the HLREG0.*RXLNGTHERREN* bit must be set to 1b. (see Section 8.2.2.16.1). This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.2.17.32 Packets Received [64 Bytes] Count - PRC64 (0x0000405C)

Field	Bit(s)	Init.	Type	Description
PRC64	31:0	0x0	RW	**Packets Received Count (64 Bytes)** Number of good packets received that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless of receive enablement, and does not include received flow control packets.

8.2.2.17.33 Packets Received [65-127 Bytes] Count - PRC127 (0x00004060)

Field	Bit(s)	Init.	Type	Description
PRC127	31:0	0x0	RW	**Packets Received Count (65-127 Bytes)** Number of packets received that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless of receive enablement, and does not include received flow control packets.

8.2.2.17.34 Packets Received [128-255 Bytes] Count - PRC255 (0x00004064)

Field	Bit(s)	Init.	Type	Description
PRC255	31:0	0x0	RW	**Packets Received Count (128-255 Bytes)** Number of packets received that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless of receive enablement, and does not include received flow control packets.

8.2.2.17.35 Packets Received [256-511 Bytes] Count - PRC511 (0x00004068)

Field	Bit(s)	Init.	Type	Description
PRC511	31:0	0x0	RW	**Packets Received Count (256-511 Bytes)** Number of packets received that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless of receive enablement, and does not include received flow control packets.

8.2.2.17.36 Packets Received [512-1023 Bytes] Count - PRC1023 (0x0000406C)

Field	Bit(s)	Init.	Type	Description
PRC1023	31:0	0x0	RW	**Packets Received Count (512-1023 Bytes)** Number of packets received that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless of receive enablement, and does not include received flow control packets.

8.2.2.17.37 Packets Received [1024 to Max Bytes] Count - PRC1522 (0x00004070)

Field	Bit(s)	Init.	Type	Description
PRC1522	31:0	0x0	RW	**Packets Received Count (1024-Max Bytes)** Number of packets received that are 1024-Max bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets. The maximum is dependent on the current receiver configuration and the type of packet being received. If a packet is counted in Receive Oversized Count, it is not counted in this register. Due to changes in the standard for maximum frame size for VLAN tagged frames in 802.3, this device accepts packets which have a maximum length of 1522 bytes. The RMON statistics associated with this range has been extended to count 1522 byte long packets.

8.2.2.17.38 Good Packets Received Count - GPRC (0x00004074)

Field	Bit(s)	Init.	Type	Description
GPRC	31:0	0x0	RO	**Good Packets Received Count** Number of good (non-erred) Rx packets (from the network) that pass L2 filtering and have legal length. This registers counts packets regardless of receive enable, and does not count flow control packets.

8.2.2.17.39 Broadcast Packets Received Count - BPRC (0x00004078)

Field	Bit(s)	Init.	Type	Description
BPRC	31:0	0x0	RO	**Broadcast Packets Received Count** Number of good (non-erred) broadcast packets received. This register does not count received broadcast packets when the broadcast address filter is disabled. The counter counts packets regardless of receive enablement.

8.2.2.17.40 Multicast Packets Received Count - MPRC (0x0000407C)

Field	Bit(s)	Init.	Type	Description
MPRC	31:0	0x0	RO	**Multicast Packets Received Count** Number of good (non-erred) multicast packets received that pass L2 filtering (excluding Broadcast packets). This register does not count received flow control packets. This register counts packets regardless of receive enablement.

8.2.2.17.41 Good Packets Transmitted Count - GPTC (0x00004080)

Field	Bit(s)	Init.	Type	Description
GPTC	31:0	0x0	RC	**Good Packets Transmitted Count** Number of good packets transmitted. This register counts good (non-erred) transmitted packets. A good transmit packet is considered one that is 64 or more bytes (from <Destination Address> through <CRC>, inclusively) in length. The register counts transmitted clear packets, secure packets and FC packets.

8.2.2.17.42 Good Octets Received Count Low - GORCL (0x00004088)

Field	Bit(s)	Init.	Type	Description
CNT_L	31:0	0x0	RC	**Count Low** Lower 32 bits of the "Good Octets Received" counter. The GORCL and GORCH registers make up a logical 36-bit octet counter of the packets counted by the GPRC (see Section 8.2.2.17.38). This register includes bytes received in a packet from the *Destination Address* field through the *CRC* field, inclusively.

8.2.2.17.43 Good Octets Received Count High - GORCH (0x0000408C)

Field	Bit(s)	Init.	Type	Description
CNT_H	3:0	0x0	RC	**Count High** Higher 4 bits of the "Good Octets Received" counter associated with the GORCL register (Section 8.2.2.17.42).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.44 Good Octets Transmitted Count Low - GOTCL (0x00004090)

Field	Bit(s)	Init.	Type	Description
CNT_L	31:0	0x0	RC	**Count Low** Lower 32 bits of the "Good Octets Transmitted" counter. The GOTCL and GOTCH registers make up a logical 36-bit counter of successfully transmitted octets (in packets counted by GPTC – see Section 8.2.2.17.41). This register includes transmitted bytes in a packet from the *Destination Address* field through the *CRC* field, inclusively.

8.2.2.17.45 Good Octets Transmitted Count High - GOTCH (0x00004094)

Field	Bit(s)	Init.	Type	Description
CNT_H	3:0	0x0	RC	**Count High** Higher 4 bits of the "Good Octets Transmitted" counter associated with the GOTCL register (Section 8.2.2.17.44).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.46 Receive Undersize Count - RUC (0x000040A4)

Field	Bit(s)	Init.	Type	Description
RUC	31:0	0x0	RC	**Receive Undersize Error** This register counts the number of received frames that are shorter than minimum size (64 bytes from <Destination Address> through <CRC>, inclusively), and had a valid CRC. This register counts packets regardless of L2 filtering and receive enablement. This register does not count packets with SFD errors or packets discarded by the MAC layer (e.g. packets smaller than 12 bytes).

8.2.2.17.47 Receive Fragment Count - RFC (0x000040A8)

Field	Bit(s)	Init.	Type	Description
RFC	31:0	0x0	RC	**Receive Fragment Count** Number of receive fragment errors (frame shorter than 64 bytes from <Destination Address> through <CRC>, inclusively) that have bad CRC (this is slightly different from the Receive Undersize Count (RUC) register). This register counts packets regardless of L2 filtering and receive enablement.

8.2.2.17.48 Receive Oversize Count - ROC (0x000040AC)

Field	Bit(s)	Init.	Type	Description
ROC	31:0	0x0	RC	**Receive Oversize Error** This register counts the number of received frames that are longer than maximum size as defined by MAXFRS.*MFS* (from <Destination Address> through <CRC>, inclusively) and have valid CRC. This register counts packets regardless of L2 filtering and receive enablement. This register does not count packets with SFD errors or packets discarded by the MAC layer (e.g. packets smaller than 12 bytes).

8.2.2.17.49 Receive Jabber Count - RJC (0x000040B0)

Field	Bit(s)	Init.	Type	Description
RJC	31:0	0x0	RC	**Receive Jabber Count** This register counts the number of received packets regardless of L2 filtering and receive enablement, and are greater than maximum size and have bad CRC (this is slightly different from the Receive Oversize Count register). The packets length is counted from <Destination Address> through <CRC>, inclusively. This register counts packets regardless of L2 filtering and receive enablement.

8.2.2.17.50 Management Packets Received Count - MNGPRC (0x000040B4)

Field	Bit(s)	Init.	Type	Description
MNGPRC	31:0	0x0	RO	**Management Packets Received Count** This register counts the total number of packets received that pass the management filters. Management packets include RMCP and ARP packets. SFD errors and short packets are not counted, except that packets dropped because the management receives a jumbo packet or because the receive FIFO is full are counted.

8.2.2.17.51 Management Packets Dropped Count - MNGPDC (0x000040B8)

Field	Bit(s)	Init.	Type	Description
MPDC	31:0	0x0	RO	**Management Packets Dropped Count** This register counts the total number of packets received that pass the management filters and then are dropped because of one of the following: • The management receive FIFO is full or disabled. • A packet bigger than the maximal allowed size is received. • A packet length mismatch. The maximum allowed size is 1518 + the size of any recognized L2 header (VLAN, extended VLAN, E-tag, etc.). Management packets include any packet directed to the manageability console (such as RMCP and ARP packets).

8.2.2.17.52 Total Octets Received Low - TORL (0x000040C0)

Field	Bit(s)	Init.	Type	Description
CNT_L	31:0	0x0	RC	**Counter Low** Lower 32 bits of the "Total Octets Received" counter. The TORL and TORH registers make up a logical 36-bit counter of the total received octets (in the packets counted by the TPR counter – see Section 8.2.2.17.54). This register includes bytes received in a packet from the *Destination Address* field through the *CRC* field, inclusively.

8.2.2.17.53 Total Octets Received High - TORH (0x000040C4)

Field	Bit(s)	Init.	Type	Description
CNT_H	3:0	0x0	RC	**Counter High** Higher 4 bits of the "Total Octets Received" counter associated with the TORL register (Section 8.2.2.17.52).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.54 Total Packets Received - TPR (0x000040D0)

Field	Bit(s)	Init.	Type	Description
TPR	31:0	0x0	RC	**Total Packets Received** This register counts the total number of all packets received. All packets received are counted in this register, regardless of their length, whether they are erred, regardless on L2 filtering and receive being enabled but excluding Flow Control packets. The TPR might count packets interrupted by link disconnect although they have a CRC error. This register does not count packets with SFD errors, or packets discarded by the MAC layer (e.g. packets smaller than 12 bytes).

8.2.2.17.55 Total Packets Transmitted - TPT (0x000040D4)

Field	Bit(s)	Init.	Type	Description
TPT	31:0	0x0	RC	**Total Packets Transmitted** This register counts the total number of all packets transmitted. This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets.

8.2.2.17.56 Packets Transmitted [64 Bytes] Count - PTC64 (0x000040D8)

Field	Bit(s)	Init.	Type	Description
PTC64	31:0	0x0	RC	**Packets Transmitted Count (64 bytes)** Number of packets transmitted that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets.

8.2.2.17.57 Packets Transmitted [65-127 Bytes] Count - PTC127 (0x000040DC)

Field	Bit(s)	Init.	Type	Description
PTC127	31:0	0x0	RC	**Packets Transmitted Count (65-127 bytes)** Number of packets transmitted that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets.

8.2.2.17.58 Packets Transmitted [128-255 Bytes] Count - PTC255 (0x000040E0)

Field	Bit(s)	Init.	Type	Description
PTC255	31:0	0x0	RC	**Packets Transmitted Count (128-255 bytes)** Number of packets transmitted that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets.

8.2.2.17.59 Packets Transmitted [256-511 Bytes] Count - PTC511 (0x000040E4)

Field	Bit(s)	Init.	Type	Description
PTC511	31:0	0x0	RC	**Packets Transmitted Count (256-511 bytes)** Number of packets transmitted that are 265-511 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets.

8.2.2.17.60 Packets Transmitted [512-1023 Bytes] Count - PTC1023 (0x000040E8)

Field	Bit(s)	Init.	Type	Description
PTC1023	31:0	0x0	RC	**Packets Transmitted Count (512-1023 bytes)** Number of packets transmitted that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets.

8.2.2.17.61 Packets Transmitted [Greater than 1024 Bytes] Count - PTC1522 (0x000040EC)

Field	Bit(s)	Init.	Type	Description
PTC1522	31:0	0x0	RC	**Packets Transmitted Count (Greater Than 1024 bytes)** Number of packets transmitted that are 1024 or more bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, and manageability packets. Due to changes in the standard for maximum frame size for VLAN tagged frames in 802.3, this device transmits packets which have a maximum length of 1522 bytes. The RMON statistics associated with this range has been extended to count 1522-byte long packets. This register counts all packets, including standard and secure packets.

8.2.2.17.62 Multicast Packets Transmitted Count - MPTC (0x000040F0)

Field	Bit(s)	Init.	Type	Description
MPTC	31:0	0x0	RC	**Multicast Packets Transmitted Count** This register counts the number of multicast packets transmitted. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets.

8.2.2.17.63 Broadcast Packets Transmitted Count - BPTC (0x000040F4)

Field	Bit(s)	Init.	Type	Description
BPTC	31:0	0x0	RC	**Broadcast Packets Transmitted Count** This register counts the number of broadcast packets transmitted. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets

8.2.2.17.64 XSUM Error Count - XEC (0x00004120)

Field	Bit(s)	Init.	Type	Description
XEC	31:0	0x0	RC	**ESUM Error Count** Number of receive IPv4, TCP, UDP or SCTP XSUM errors. ***Note:*** XSUM errors are not counted when a packet has any MAC error (CRC, length, under-size, over-size, byte error, or symbol error).

8.2.2.17.65 Priority XON Received Count - PXONRXCNT[n] (0x00004140 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
XONRXC	15:0	0x0	RC	**XON Receive Count** Number of XON packets received per UP. Sticks at 0xFFFF.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.17.66 Priority XOFF Received Count - PXOFFRXCNT[n] (0x00004160 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
XOFFRXC	15:0	0x0	RC	**XOFF Receive Count** Number of XOFF packets received per UP. Sticks at 0xFFFF.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.17.67 Total Unicast Packets Received (BMC copy) - BUPRC (0x00004180)

Note: This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BUPRC	31:0	0x0	RC	**BMC Unicast Packets Received Count** This register counts the number of good (no errors) unicast packets received from the network. This register does not count unicast packets received that fail to pass address filtering. This register does not count packets counted by the Missed Packet Count (MPC) register. This register does not count flow control packets. Packets sent to the manageability engine are included in this counter.

8.2.2.17.68 BMC Total Multicast Packets Received - BMPRC (0x00004184)

Note: This register counts the same events as the MPRC register (Section 8.2.2.17.40) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BMPRC	31:0	0x0	RC	**BMC Multicast Packets Received Count** Number of good (non-erred) multicast packets received that pass L2 filtering (excluding Broadcast packets). This register does not count received flow control packets. This registers counts packets regardless of receive enablement.

8.2.2.17.69 Total Broadcast Packets Received (BMC copy) - BBPRC (0x00004188)

Note: This register counts the same events as the BPRC register (Section 8.2.2.17.39) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BBPRC	31:0	0x0	RC	**BMC Broadcast Packets Received Count** Number of good (non-erred) broadcast packets received to BMC. This register does not count received broadcast packets when the broadcast address filter is disabled. The counter counts packets regardless on receive enablement.

8.2.2.17.70 Total Unicast Packets Transmitted (BMC copy) - BUPTC (0x0000418C)

Note: This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BUPTC	31:0	0x0	RC	**BMC Unicast Packets Transmitted Count** Number of unicast packets transmitted.

8.2.2.17.71 BMC Total Multicast Packets Transmitted - BMPTC (0x00004190)

Note: This register counts the same events as the MPTC register (Section 8.2.2.17.62) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BMPTC	31:0	0x0	RC	**BMC Multicast Packets Transmitted Count** Number of multicast packets transmitted. This register counts the number of multicast packets transmitted. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets.

8.2.2.17.72 Total Broadcast Packets Transmitted (BMC copy) - BBPTC (0x00004194)

Note: This register counts the same events as the BPTC register (Section 8.2.2.17.63) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BBPTC	31:0	0x0	RC	**BMC Broadcast Packets Transmitted Count** Number of broadcast packets transmitted count. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets

8.2.2.17.73 BMC FCS Receive Errors - BCRCERRS (0x00004198)

Note: This register counts the same events as the CRCERRS register (Section 8.2.2.17.24) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BCEC	31:0	0x0	RC	**BMC CRC Error Count** Counts the number of receive packets with CRC errors. For a packet to be counted in this register, it must be 64 bytes or greater (from <Destination Address> through <CRC>, inclusively) in length. This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.2.17.74 BMC Pause XON Frames Received - BXONRXC (0x0000419C)

Note: This register counts the same events as the LXONRXCNT register (Section 8.2.2.17.75) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BXONRXC	15:0	0x0	RC	**BMC XON Received Count** Number of XON packets received. Sticks at 0xFFFF. XON packets can use the global address, or the station address. This register counts any XON packet whether it is a legacy XON or a priority XON. Each XON packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only the LXOFFRXCNT counter is incremented.
RESERVED	31:16	0x0	RC	Reserved

8.2.2.17.75 Link XON Received Count - LXONRXCNT (0x000041A4)

Field	Bit(s)	Init.	Type	Description
XONRXC	15:0	0x0	RC	**XON Received Count** Number of XON packets Received. Sticks at 0xFFFF. XON packets can use the global address, or the station address. This register counts any XON packet whether it is a legacy XON or a priority XON. Each XON packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only the LXOFFRXCNT counter is incremented.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.17.76 Link XOFF Received Count - LXOFFRXCNT (0x000041A8)

Field	Bit(s)	Init.	Type	Description
XOFFRXC	15:0	0x0	RC	**XOFF Received Count** Number of XOFF packets Received. Sticks at 0xFFFF. XOFF packets can use the global address, or the station address. This register counts any XOFF packet whether it is a legacy XOFF or a priority XOFF. Each XOFF packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only this counter is incremented.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.17.77 Good Rx Non-Filtered Packet Counter - RXNFGPC (0x000041B0)

Field	Bit(s)	Init.	Type	Description
GPC	31:0	0x0	RC	**Good Packet Counter** Number of good (non-erred with legal length) Rx packets (from the network) regardless of packet filtering and receive enablement.

8.2.2.17.78 Good Rx Non-Filter Byte Counter Low - RXNFGBCL (0x000041B4)

Field	Bit(s)	Init.	Type	Description
BCL	31:0	0x0	RC	**Byte Counter Low** Low 32 bits of the 36-bit byte counter of good (non-erred) Rx packets that match the RXNFGPC register (Section 8.2.2.17.77). The counter counts all bytes from *Destination Address* field through the *CRC* field, inclusively.

8.2.2.17.79 Good Rx Non-Filter Byte Counter High - RXNFGBCH (0x000041B8)

Field	Bit(s)	Init.	Type	Description
BCH	3:0	0x0	RC	**Byte Counter High** High 4 bits of the 36-bit byte counter associated with the RXFGBCL register (Section 8.2.2.17.78).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.80 BMC2OS Packets Sent by BMC - B2OSPC (0x000041C0)

Field	Bit(s)	Init.	Type	Description
B2OSPC	31:0	0x0	RC	**BMC-to-OS Packet Count** BMC2OS packets sent by BMC. This register counts the total number of transmitted packets sent from the manageability path that were sent to host. This includes packets received by the host and packet dropped in the device due to congestion conditions. Counter is cleared when read by driver. Counter is also cleared by PCIe reset and software reset. When reaching maximum value, counter does not wrap-around.

8.2.2.17.81 OS2BMC Packets Received by BMC - O2BGPTC (0x000041C4)

Field	Bit(s)	Init.	Type	Description
O2BGPTC	31:0	0x0	RC	**OS-to-BMC Good Packets Transmitted Count** This register counts the total number of packets originating from the host that reached the NC-SI interface. This includes packets sent from the host to the BMC or consumed by the internal manageability engine. Counter is cleared when read by driver. Counter is also cleared by PCIe reset and software reset. When reaching maximum value, counter does not wrap-around.

8.2.2.17.82 BMC Pause XOFF Frames Received - BXOFFRXC (0x000041E0)

Note: This register counts the same events as the LXOFFRXCNT register (Section 8.2.2.17.76) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BXOFFRXC	15:0	0x0	RC	**BMC XOFF Received Count** Number of XOFF packets received. Sticks at 0xFFFF. XOFF packets can use the global address, or the station address. This register counts any XOFF packet whether it is a legacy XOFF or a priority XOFF. Each XOFF packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only this counter is incremented.
RESERVED	31:16	0x0	RC	Reserved

8.2.2.17.83 BMC Pause XON Frames Transmitted - BXONTXC (0x000041E4)

Field	Bit(s)	Init.	Type	Description
BXONTXC	15:0	0x0	RC	**BMC XON Transmitted Count** Number of XON packets Transmitted. Sticks at 0xFFFF. *BXONTXC* is incremented by one for each Link XON packet when MFLCN.*RFCE* is set and for each Priority XON packet when corresponding MFLCN.*RPFCE* bit is set (see Section 8.2.2.3.6).
RESERVED	31:16	0x0	RC	Reserved.

8.2.2.17.84 BMC Pause XOFF Frames Transmitted - BXOFFTXC (0x000041E8)

Field	Bit(s)	Init.	Type	Description
BXOFFTXC	15:0	0x0	RC	**BMC XOFF Transmitted Count** Number of XOFF packets Transmitted. Sticks at 0xFFFF. *BXOFFTXC* is incremented by one for each Link XOFF packet when MFLCN.*RFCE* is set and for each Priority XOFF packet when corresponding MFLCN.*RPFCE* bit is set (see Section 8.2.2.3.6).
RESERVED	31:16	0x0	RC	Reserved.

8.2.2.17.85 Sideband Receive Dropped Packet Count - B2OSDPC (0x000041F0)

Note: Although the name is B2OSDPC, this counter includes all drops from the BMC, even if sent to LAN. This counter is shared between all ports, and counts events of packets dropped regardless of their destination port.

Field	Bit(s)	Init.	Type	Description
B2OSDPC	31:0	0x0	RW	**BMC-to-OS Dropped Packet Count** Counts the number of packets received from the BMC interface receive that were dropped. The following causes can create a drop: • Memory buffer full. • Pause packet. • PT packet without SA match or requires enables. • MCTP drop (unsupported message type). • Length error. • Reject by specific interface (SMBus abort, MCTP reject, RMII reject).

8.2.2.17.86 Fiber Channel CRC Error Count - FCCRC (0x00005118)

Field	Bit(s)	Init.	Type	Description
CRC_CNT	15:0	0x0	RC	**FC CRC Count** Counts the number of packets with good Ethernet CRC and bad FC CRC.
RESERVED	31:16	X	RSV	Reserved.

8.2.2.17.87 Queue Packets Transmitted Count - QPTC_ALIAS[n] (0x00006030 + 0x40*n, n=0...15; RC)

Field definitions are the same as those defined in Section 8.2.2.17.89.

8.2.2.17.88 Transmit Queue Statistic Mapping Registers - TQSM[n] (0x00008600 + 0x4*n, n=0...31)

These registers define the mapping of the transmit queues to the per-queue statistics. Several queues can be mapped to a single statistic register. Each statistic register counts the number of packets and bytes of all the queues that are mapped to that statistics. The registers counting Tx queue statistics are QPTC and QBTC (refer to Section 8.2.2.17.89, Section 8.2.2.17.90, and Section 8.2.2.17.91).

Field	Bit(s)	Init.	Type	Description
Q_MAP_0	3:0	0x0	RW	**Queue Map 0** For each register 'n', *Q_MAP*[0] defines the per queue statistic registers that are mapped to Tx queue '4*n+0'.
RESERVED	7:4	0x0	RSV	Reserved.
Q_MAP_1	11:8	0x0	RW	**Queue Map 1** For each register 'n', *Q_MAP*[1] defines the per queue statistic registers that are mapped to Tx queue '4*n+1'.
RESERVED	15:12	0x0	RSV	Reserved.
Q_MAP_2	19:16	0x0	RW	**Queue Map 2** For each register 'n', *Q_MAP*[2] defines the per queue statistic registers that are mapped to Tx queue '4*n+2'.
RESERVED	23:20	0x0	RSV	Reserved.
Q_MAP_3	27:24	0x0	RW	**Queue Map 3** For each register 'n', *Q_MAP*[3] defines the per queue statistic registers that are mapped to Tx queue '4*n+3'.
RESERVED	31:28	0x0	RSV	Reserved.

8.2.2.17.89 Queue Packets Transmitted Count - QPTC[n] (0x00008680 + 0x4*n, n=0...15)

Note: Additional address(es): 0x06030 + 0x40*n, n=0...15

Field	Bit(s)	Init.	Type	Description
PTC	31:0	0x0	RC	**Packets Transmitted Count** Number of packets transmitted for the Queue. A packet is considered as transmitted if it is forwarded to the MAC unit for transmission to the network and/or is accepted by the internal Tx-to-Rx switch enablement logic. Packets dropped due to anti-spoofing filtering, or traffic sent to the BMC and blocked due to security violations, or loopback packets that are rejected by the Tx-to-Rx switch, are not counted.

8.2.2.17.90 Queue Bytes Transmitted Count Low - QBTC_L[n] (0x00008700 + 0x8*n, n=0...15)

Field	Bit(s)	Init.	Type	Description
BTC_L	31:0	0x0	RC	**Bytes Transmitted Count Low** Lower 32 bits of the statistic counter. The QBTC_L and QBTC_H registers make up a logical 36-bit counter of transmitted bytes of the packets counted by the matched QPTC counter. These registers count all bytes in the packets from the *Destination Address* field through the *CRC* field, inclusively. These registers must be accessed as two consecutive 32-bit entities while QBTC_L register is read first, or a single 64-bit read cycle. Each register is read-cleared. In addition, it sticks at 0xFF...F to avoid overflow.

8.2.2.17.91 Queue Bytes Transmitted Count High - QBTC_H[n] (0x00008704 + 0x8*n, n=0...15)

Field	Bit(s)	Init.	Type	Description
BTC_H	3:0	0x0	RC	**Bytes Transmitted Count High** Higher 4 bits of the statistic counter described in QBTC_L (Section 8.2.2.17.90).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.92 Switch Security Violation Packet Count - SSVPC (0x00008780)

Field	Bit(s)	Init.	Type	Description
SSVPC	31:0	0x0	RC	**Switch Security Violation Packet Count** This register counts all Tx packets dropped. For example, packets dropped due to switch security violations such as SA or VLAN anti-spoof filtering, or packet that has (inner) VLAN that contradicts with PFVMVIR register definitions. Valid only in VMDq or IOV mode. This counter includes also traffic sent to the BMC and blocked due to security violations. or packets dropped due to malicious events detection.

8.2.2.17.93 FCoE Packets Transmitted Count - FCOEPTC (0x00008784)

Field	Bit(s)	Init.	Type	Description
PTC	31:0	0x0	RC	**Packets Transmitted Count** Number of FCoE packets transmitted. The counter does not include packets dropped due to anti-spoofing filtering or VLAN tag validation. This rule is applicable if FCoE traffic is sent by a VF.

8.2.2.17.94 FCoE DWord Transmitted Count - FCOEDWTC (0x00008788)

Field	Bit(s)	Init.	Type	Description
DWTC	31:0	0x0	RC	**DWord Transmitted Count** 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).

8.2.2.17.95 DMA Good Tx Packet Counter - TXDGPC (0x000087A0)

Field	Bit(s)	Init.	Type	Description
GPTC	31:0	0x0	RC	**Good Packets Transmitted Count** Number of Tx packets from host memory. This counter includes packets that are transmitted to the external network as well as packets that are transmitted only to local VMs. The later occurs only in VT mode when the local switch is enabled. Dropped packets counted in SSVPC register are not counted here.

8.2.2.17.96 DMA Good Tx Byte Counter Low - TXDGBCL (0x000087A4)

Field	Bit(s)	Init.	Type	Description
BCL	31:0	0x0	RC	**Byte Counter Low** The low 32 bits of the 36-bit byte counter of the Tx packets that match the TXDGPC register (Section 8.2.2.17.95). The counter counts all bytes posted by the host and the VLAN (if bytes are added by hardware). Dropped packets counted in SSVPC register are not counted here.

8.2.2.17.97 DMA Good Tx Byte Counter High - TXDGBCH (0x000087A8)

Field	Bit(s)	Init.	Type	Description
BCH	3:0	0x0	RC	**Byte Counter High** High 4 bits of the 36-bit byte counter associated with the TXDGBCL register (Section 8.2.2.17.96).
RESERVED	31:4	0x0	RSV	Reserved.

8.2.2.17.98 OS2BMC Packets Transmitted by Host - O2BSPC (0x000087B0)

Field	Bit(s)	Init.	Type	Description
O2BPC	31:0	0x0	RC	**OS-to-BMC Packet Count** OS2BMC packets transmitted count. Counts the total number of packets originating from the functions that were sent to the manageability path. This includes packets received by the BMC and packet dropped in the X550 due to congestion conditions or due to anti-spoof check. Counter is cleared when read by driver. Counter is also cleared by PCIe reset and software reset. When reaching maximum value counter does not wrap-around.

8.2.2.18 PF - Wake-Up and Proxy Control Registers

8.2.2.18.1 Wake-Up Control Register - WUC (0x00005800)

The *PME_EN* and *PME_STATUS* bits are reset when LAN_PWR_GOOD is 0. When AUX_PWR=0, these bits are also reset by the assertion of PE_RST_N.

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
PME_EN	1	0b	RW	**PME Enable** This bit is used by the software device driver to read the *PME_En* bit of the Power Management Control/Status Register (PMCSR) without writing to the PCIe configuration space. Writing a 1b to this bit clears it. ***Note:*** Software should not modify this bit while PME enablement is active.
PME_STATUS	2	0b	RW1C	**PME Status** This bit is set when the X550 receives a wake-up event. It is the same as the *PME_Status* bit in the Power Management Control/Status Register (PMCSR). Writing a 1b to this bit clears it. The *PME_Status* bit in the PMCSR register is also cleared.
RESERVED	3	0b	RSV	Reserved.
WKEN	4	1b	RW	**Wake Enable** This bit can be cleared to disable the PE_WAKE_N pin assertion (= 0) even if APM is enabled in the NVM. In this case, PMCSR and WUS wake-up statuses are invalid. ***Note:*** This bit should not be cleared while in ACPI mode.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.2.18.2 Wake-Up Filter Control Register - WUFC (0x00005808)

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, and a value of 0b means the filter is turned off.

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**
EX	2	0b	RW	**Directed Exact Wake-up Enable**
MC	3	0b	RW	**Directed Multicast Wake-up Enable** Setting this bit does not enable broadcast packets, which are enabled by the *BC* bit in this register.
BC	4	0b	RW	**Broadcast Wake-up Enable**
ARP	5	0b	RW	**ARP/IPv4 Request Packet Wake-up Enable**
IPV4	6	0b	RW	**Directed IPv4 Packet Wake-up Enable**
IPV6	7	0b	RW	**Directed IPv6 Packet Wake-up Enable**
RESERVED	14:8	0x0	RSV	Reserved.
NOTCO	15	0b	RW	**No TCO** Ignore TCO/managements packets for wake-up. 0b = Ignore all TCO/management packets for wake-up, except packets that meet the criteria defined in the MNGONLY register via the *Host Enable* field (for example, criteria intended for the host as well as to the MC). 1b = Ignore all TCO/management packets for wake-up. While in normal operation, this ignore is forwarded to the host as well as to the MC.
FLX0	16	0b	RW	**Flexible Filter 0 Enable** Controls the usage of the FHFT[0] register (0x9000).

Field	Bit(s)	Init.	Type	Description
FLX1	17	0b	RW	**Flexible Filter 1 Enable** Controls the usage of the FHFT[1] register (0x9100).
FLX2	18	0b	RW	**Flexible Filter 2 Enable** Controls the usage of the FHFT[2] register (0x9200).
FLX3	19	0b	RW	**Flexible Filter 3 Enable** Controls the usage of the FHFT[3] register (0x9300).
FLX4	20	0b	RW	**Flexible Filter 4 Enable** Controls the usage of the FHFT[0] register (0x9600).
FLX5	21	0b	RW	**Flexible Filter 5 Enable** Controls the usage of the FHFT[5] register (0x9700).
FLX6	22	0b	RW	**Flexible Filter 6 Enable** Controls the usage of the FHFT[6] register (0x9800).
FLX7	23	0b	RW	**Flexible Filter 7 Enable** Controls the usage of the FHFT[7] register (0x9900).
RESERVED	30:24	0x0	RSV	Reserved.
FW_RST_WK	31	0b	RW	**Enable Wake on Firmware Reset Assertion** When set, a firmware reset causes a system wake-up that enables the software driver to resend proxying information to firmware.

8.2.2.18.3 Wake-Up Status Register - WUS (0x00005810)

This register is used to record statistics about all wake-up packets received. If a packet matches multiple criteria, multiple bits are set by hardware. Software writing a 1b to any bit clears that bit.

This register is not cleared when PE_RST_N is asserted. It is only cleared when LAN_PWR_GOOD is de-asserted, or when cleared by the driver.

Field	Bit(s)	Init.	Type	Description
LNKC	0	0b	RW1C	**Link Status Changed** This bit can be set even if another event is already set.
MAG	1	0b	RW1C	**Magic Packet Received**
EX	2	0b	RW1C	**Directed Exact Packet Received** The packet's address matched one of the 16 pre-programmed exact values in the Receive Address registers.
MC	3	0b	RW1C	**Directed Multicast Packet Received** The packet was a multicast packet that's hashed to a value that corresponds to a 1 bit in the Multicast Table Array.
BC	4	0b	RW1C	**Broadcast Packet Received**
ARP	5	0b	RW1C	**ARP/IPv4 Request Packet Received**
IPV4	6	0b	RW1C	**Directed IPv4 Packet Received**
IPV6	7	0b	RW1C	**Directed IPv6 Packet Received**
MNG	8	0b	RW1C	**Manageability** Indicates that a manageability event that should cause a PME to happen.
RESERVED	15:9	0x0	RSV	Reserved.
FLX0	16	0b	RW1C	**Flexible Filter 0 Match**
FLX1	17	0b	RW1C	**Flexible Filter 1 Match**
FLX2	18	0b	RW1C	**Flexible Filter 2 Match**
FLX3	19	0b	RW1C	**Flexible Filter 3 Match**
FLX4	20	0b	RW1C	**Flexible Filter 4 Match**

Field	Bit(s)	Init.	Type	Description
FLX5	21	0b	RW1C	**Flexible Filter 5 Match**
FLX6	22	0b	RW1C	**Flexible Filter 6 Match**
FLX7	23	0b	RW1C	**Flexible Filter 7 Match**
RESERVED	30:24	0x0	RSV	Reserved.
FW_RST_WK	31	0b	RW1C	**Wake Due to Firmware Reset Assertion Event** When set to 1b, indicates that asserting a firmware reset causes the system wake-up so the software driver can re-send proxying information to firmware. This bit can be set even if another event is already set.

8.2.2.18.4 IP Address Valid - IPAV (0x00005838)

This register indicates whether the IP Addresses in the IP Address Table are valid.

Field	Bit(s)	Init.	Type	Description
V40	0	0b	RW	**IPv4 Address 0 Valid** Loaded from NVM.
V41	1	0b	RW	**IPv4 Address 1 Valid**
V42	2	0b	RW	**IPv4 Address 2 Valid**
V43	3	0b	RW	**IPv4 Address 3 Valid**
RESERVED	15:4	0x0	RSV	Reserved.
V60	16	0b	RW	**IPv6 Address 0 Valid**
V61	17	0b	RW	**IPv6 Address 1 Valid**
V62	18	0b	RW	**IPv6 Address 2 Valid**
V63	19	0b	RW	**IPv6 Address 3 Valid**
RESERVED	31:20	0x0	RSV	Reserved.

8.2.2.18.5 IPv4 Address Table - IP4AT[n] (0x00005840 + 0x8*n, n=0...3)

4 x IPv4 addresses for ARP/IPv4 Request packet and Directed IPv4 packet wake-up. IPv4[0] is loaded from MIPAF words in the NVM.

Field	Bit(s)	Init.	Type	Description
IPV4ADDR	31:0	X	RW	**IPv4 Address** IPv4 Address 'n', where 'n' = 0...3.

8.2.2.18.6 IPv6 Address Table - IP6AT[n] (0x00005880 + 0x4*n, n=0...3)

First IPv6 addresses for Neighbor Discovery packet filtering and Directed IPv6 packet wake-up.

Field	Bit(s)	Init.	Type	Description
IPV6ADDR	31:0	X	RW	**IPv6 Address** 4 x Register IPv6 filter. Register 'n' contains bytes '4*n' up to '4*n+3' of the IPv6 address. LS byte of register '0' is first on the wire.

8.2.2.18.7 Wake-Up Packet Length - WUPL (0x00005900)

This register is de-featured and software should not access it (read or write).

8.2.2.18.8 IPv6 Address Table Extended - IP6AT_EXT[n] (0x00005990 + 0x4*n, n=0...11)

3 x IPv6 addresses for Neighbor Discovery packet filtering.

Field	Bit(s)	Init.	Type	Description
IPV6ADDR	31:0	X	RW	**IPv6 Address** 4 x Register IPv6 filter. Register 'n' contains bytes '4*n' up to '4*n+3' of the IPv6 address. LS byte of register '0' is first on the wire.

8.2.2.18.9 Wake-Up Packet Memory (128 Bytes) - WUPM[n] (0x00005A00 + 0x4*n, n=0...31)

Field	Bit(s)	Init.	Type	Description
WUPD	31:0	X	RO	**Wake-Up Packet Data** This register is used to store the first 128 bytes of the wake-up packet for software retrieval after the system wakes up. It should not be cleared by any reset including master reset.

8.2.2.18.10 Proxying Status Register - PROXYS (0x00005F60)

This register is used to record statistics about all proxying packets received. If a packet matches multiple criteria, 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 de-asserted, or when cleared by the software device driver.

Note: If additional packets are received that match one of the wake-up filters, after the original wake-up packet is received, the PROXYS register is updated with the matching filters accordingly.

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	00b	RSV	Reserved.
EX	2	0b	RW1C	**Exact Packet Received**
RESERVED	4:3	00b	RSV	Reserved.
ARP_DIRECTED	5	0b	RW1C	**ARP Directed Received** ARP request packet with IP4AT filter match received. When set to 1b, indicates a match on any ARP request packet that passed main filtering and Target IP Address also matches one of the valid IP4AT filters.
RESERVED	8:6	000b	RSV	Reserved.
NS	9	0b	RW1C	**IPV6 Neighbor Solicitation Received** When set to 1b, indicates a match on NS packet that passed main filtering.
NS_DIRECTED	10	0b	RW1C	**IPV6 Neighbor Solicitation with Directed DA Match Received.** When set to 1b, indicates a match on NS packet and Target IP Address also matches IP6AT filter.
ARP	11	0b	RW1C	**ARP Request Packet Received** When set to 1b, indicates a match on ARP request packet that passed main filtering.

Field	Bit(s)	Init.	Type	Description
MLD	12	0b	RW1C	**IPv6 Multicast Listener Discovery (MLD) Packet Received** When set to 1b, indicates a match on any of the following MLD packet types that passed main filtering: • Multicast Listener Query (ICMPv6 Type = decimal 130). Defined in MLDv1 and MLDv2. • Multicast Listener Report (ICMPv6 Type = decimal 131). Defined in MLDv1 and MLDv2. • Version 2 Multicast Listener Report Message (ICMPv6 Type = decimal 143). Defined in MLDv2 only.
RESERVED	31:13	0x0	RSV	Reserved.

8.2.2.18.11 Proxying Filter Control Register - PROXYFC (0x00005F64)

This register is used to enable each of the pre-defined filters for Proxying support. This register is not cleared when RST# is asserted. It is only cleared when LAN_PWR_GOOD is de-asserted, or when cleared by the software device driver.

Field	Bit(s)	Init.	Type	Description
PPROXYE	0	0b	RW	**Port Proxying Enable** When set to 1b, proxying of packets is enabled when device is in D3 low power state. Proxy information and requirements is passed by software driver to firmware via the host interface.
RESERVED	1	0b	RSV	Reserved.
EX	2	0b	RW	**Exact Proxying Enable**
RESERVED	4:3	00b	RSV	Reserved.
ARP_DIRECTED	5	0b	RW	**ARP Request Packet and IP4AT MATCH Proxy Enable** If set to 1b, forward to Management for proxying on match of any ARP request packet that passed main filtering and Target IP Address also matches one of the valid IP4AT filters.
RESERVED	8:6	000b	RSV	Reserved.
NS	9	0b	RW	**IPV6 Neighbor Solicitation Proxy Enable** If set to 1b, forward to Management for proxying on match of any NS packet (ICMPv6 type 135) that passed main filtering.
NS_DIRECTED	10	0b	RW	**IPV6 Neighbor Solicitation and Directed DA Match Proxy Enable** If set to 1b, forward to Management for proxying on match of NS packet and Target IP Address also matches valid IP6AT filter.
ARP	11	0b	RW	**ARP Request Packet Proxy Enable** If set to 1b, forward to Management for proxying on match of any ARP request packet that passed main filtering.
MLD	12	0b	RW	**IPv6 Multicast Listener Discovery (MLD) Proxy Enable** If set to 1b, forward to Management for Proxying on match of any of the following MLD packet types that passed main filtering: • Multicast Listener Query (ICMPv6 Type = decimal 130). Defined in MLDv1 and MLDv2. • Multicast Listener Report (ICMPv6 Type = decimal 131). Defined in MLDv1 and MLDv2. • Version 2 Multicast Listener Report Message (ICMPv6 Type = decimal 143). Defined in MLDv2 only.
RESERVED	14:13	00b	RSV	Reserved.
NOTCO	15	0b	RW	**Ignore TCO/Management Packets For Proxying** 0b = Ignore only TCO/management packets for Proxying that meet the criteria defined in the MNGONLY register (i.e., are intended only for the BMC and not the Host). 1b = Ignore any TCO/management packets for proxying, even if in normal operation it is forwarded to the Host in addition to the BMC.

Field	Bit(s)	Init.	Type	Description
RESERVED	31:16	0x0	RSV	Reserved.

8.2.2.18.12 Filter DW Even - FHFT_FILTER_DW_EVEN[n,m] (0x00009000 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009600 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3)

Notes:

- The mask field must be 8 bytes aligned even if the length field is not 8 bytes aligned, as hardware implementation compares 8 bytes at a time. Therefore, 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 0b, indicating no comparison should be done.
- In case the actual length defined by the length field register and the mask bits is not 8 bytes aligned, there might be a case that a packet which is shorter than the actual required length passes the flexible filter. This might happen due to comparison of up to 7 bytes that come after the packet, but are not a real part of the packet.
- The last DW of each filter contains a length field defining the number of bytes from the beginning of the packet compared by this filter. If actual packet length is less than length specified by this field, the filter fails. Otherwise, it depends on the result of actual byte comparison. The value should not be greater than 128.

Field	Bit(s)	Init.	Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	**FHFT Filter DW Even** Even DW of the flex filter.

8.2.2.18.13 Filter DW Odd - FHFT_FILTER_DW_ODD[n,m] (0x00009004 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009604 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3)

Field	Bit(s)	Init.	Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	**FHFT Filter DW Odd** Odd DW of the flex filter.

8.2.2.18.14 Filter Mask - FHFT_FILTER_MASK[n,m] (0x00009008 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009608 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3)

Field	Bit(s)	Init.	Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	**FHFT Filter Mask** Bit mask for the 8 bytes of the filter (bit per byte).
RESERVED	31:8	0x0	RSV	Reserved.

8.2.2.18.15 Filter Length - FHFT_FILTER_LENGTH[n,m] (0x0000900C + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x0000960C + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3)

Field	Bit(s)	Init.	Type	Description
LENGTH	7:0	0x0	RW	**Flex Filter Length** The *LENGTH* field is only valid on the last DW of the filter. All other length fields are reserved.
RESERVED	31:8	0x0	RSV	Reserved.

8.2.2.19 PF - Management Filters Registers

The Management Filters registers are RO for the host. These registers are initialized at LAN Power Good and can be loaded from the NVM by the manageability firmware.

8.2.2.19.1 Management VLAN TAG Value - MAVTV[n] (0x00005010 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
VID	11:0	0x0	RW	**VLAN ID** Contains the VLAN ID that should be compared with the incoming packet.
RESERVED	31:12	0x0	RSV	Reserved.

8.2.2.19.2 Management Flex UDP/TCP Ports - MFUTP[n] (0x00005030 + 0x4*n, n=0...7)

Note: Each 32-bit register (n=0,...,7) refers to two port filters (register 0 refers to ports 0 and 1, register 2 refers to ports 2 and 3, etc.). SCTP packets do not match the MFUTP filters. MFUTP filters are programmed in Network order.

Field	Bit(s)	Init.	Type	Description
MFUTP_2N	15:0	0x0	RW	**Management Flex UDP/TCP Port 2n** (2n)-th Management Flex UDP/TCP port.
MFUTP_2N_1	31:16	0x0	RW	**Management Flex UDP/TCP Port 2n+1** (2n+1)-th Management Flex UDP/TCP port.

8.2.2.19.3 BMC IP Address Register - BMCIP[n] (0x00005050 + 0x4*n, n=0...3)

These registers contain the BMC IP Address table.

Field	Bit(s)	Init.	Type	Description
IPADDR	31:0	0x0	RW	**IP Address** 4 bytes of 16-bytes destination IP Address of the BMC. • n=0 contains the MSB for an IPv6 IP Address. • n=3 contains an IPv4 IP Address or the LSB for an IPv6 IP Address. For an IPv4 address, BMCIP 0...2 must be written with zeros. ***Note:*** This field is defined in Big Endian (LS byte is first on the wire).

8.2.2.19.4 BMC IP Valid Register - BMCIPVAL (0x00005060)

This register indicates the type of IP Address stored in the IPVAL register, and indicates if a valid address is stored.

Field	Bit(s)	Init.	Type	Description
IPADDR_TYPE	0	0b	RW	**IP Address Type** 0b = IPv4 1b = IPv6
IPADDR_VALID	1	0b	RW	**IP Address Valid** 0b = IP Address in BMCIP is not valid. 1b = IP Address in BMCIP is valid.
RESERVED	31:2	0x0	RSV	Reserved.

8.2.2.19.5 Manageability Decision Filters Ext - MDEF_EXT[n] (0x00005160 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
L2_ETHERTYPE_AND	3:0	0x0	RW	**L2 EtherType AND** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section).
L2_ETHERTYPE_OR	7:4	0x0	RW	**L2 EtherType OR** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section).
FLEX_PORT	23:8	0x0	RW	**Flex Port** Controls the inclusion of Flex port filtering in the manageability filter decision (OR section). Bit 8 corresponds to flex port 0, etc.
FLEX_TCO	24	0b	RW	**Flex TCO** Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 24 corresponds to Flex TCO filter 0. ***Note:*** Supported only for Network traffic.
ND_135	25	0b	RW	**Neighbor Discovery 135** Controls the inclusion of Neighbor Solicitation neighbor Discovery filtering in the manageability filter decision (OR section). ***Notes:*** 1. Supported only for Network traffic. 2. Neighbor Discovery types supported by this bit is 0x87 (135d) - Neighbor Solicitation
ND_136	26	0b	RW	**Neighbor Discovery 136** Controls the inclusion of Neighbor Advertisement Neighbor Discovery filtering in the manageability filter decision (OR section). ***Notes:*** 1. Supported only for Network traffic. 2. Neighbor Discovery types supported by this bit is 0x88 (136d) - Neighbor Advertisement
ND_137	27	0b	RW	**Neighbor Discovery 137** Controls the inclusion of Redirect neighbor Discovery filtering in the manageability filter decision (OR section). ***Notes:*** 1. Supported only for Network traffic. 2. Neighbor Discovery types supported by this bit is 0x89 (137d) - Redirect
RESERVED	28	0b	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
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	**Apply to Network Traffic** 0b = Do not apply this decision filter to traffic received from the network. 1b = Apply this decision filter to traffic received from the network.
APPLY_TO_HOST_TRAFFIC	31	0b	RW	**Apply to Host Traffic** 0b = This decision filter does not apply to traffic received from the host. 1b = This decision filter applies to traffic received from the host.

8.2.2.19.6 Management Ethernet Type Filters - METF[n] (0x00005190 + 0x4*n, n=0...3)

Field	Bit(s)	Init.	Type	Description
ETYPE	15:0	0x0	RW	**EtherType** EtherType value to be compared against the *L2 EtherType* field in the Rx packet. ***Note:*** Appears in Little Endian order (high byte first on the wire).
RESERVED	29:16	0x0	RSV	Reserved.
POLARITY	30	0b	RW	**Polarity** 0b = Positive filter — Filter enters the decision filters if a match occurred. 1b = Negative filter — Filter enters the decision filters if a match did not occur.
RESERVED	31	0b	RSV	Reserved.

8.2.2.19.7 Management Control Register - MANC (0x00005820)

Field	Bit(s)	Init.	Type	Description
FC_DISCARD	0	0b	RW	**Flow Control Discard** 0b = Apply filtering rules to packets with Flow Control EtherType. 1b = Discard packets with Flow Control EtherType. ***Note:*** Flow Control EtherType is 0x8808
NCSI_DISCARD	1	0b	RW	**NC-SI Discard** 0b = Apply filtering rules to packets with NC-SI EtherType. 1b = Discard packets with NC-SI EtherType. ***Note:*** NC-SI EtherType is 0x88F8
RESERVED	16:2	0x0	RSV	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 *EN_BMC2OS* or *EN_BMC2NET* bits are set.
RESERVED	22:18	0x0	RSV	Reserved.
EN_XSUM_FILTER	23	0b	RW	**Enable XSUM Filter** When this bit is set, enables XSUM filtering to Manageability, meaning only packets that pass L3/L4 checksum are sent to the manageability block. ***Note:*** This capability is not provided for tunneled packets.
EN_IPV4_FILTER	24	0b	RW	**Enable IPv4 address Filters** 0b = These bits store a single IPv6 filter. 1b = The last 128 bits of the MIPAF register (Section 8.2.2.19.10) are used to store 4 IPv4 addresses for IPv4 filtering.

Field	Bit(s)	Init.	Type	Description
FIXED_NET_TYPE	25	0b	RW	**Fixed Net Type** 0b = Both tagged and un-tagged packets might be forwarded to manageability engine. 1b = Only packets matching the net type defined by the *NET_TYPE* field pass to manageability.
NET_TYPE	26	0b	RW	**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 cannot communicate with the OS. 1b = The BMC can communicate with the OS. When cleared, the BMC traffic is not forwarded to the OS, even if the Host MAC Address filter and VLANs (RAH/L, MTA, VFTA and PFVLVF registers) indicate that it should. When cleared the OS traffic is not forwarded to the BMC even if the decision filters indicates it should. This bit does not impact the BMC to Network traffic. ***Note:*** This bit can change while the host is sending or receiving traffic.
EN_BMC2NET	29	0b	RW	**Enable BMC-to-Network and Network-to-BMC traffic** 0b = The BMC cannot 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:*** This bit can change while the host is sending or receiving traffic.
PROXYEN	30	0b	RW	**Proxy Enable** Set by firmware to indicate that proxy offload is supported.
RESERVED	31	0b	RSV	Reserved.

8.2.2.19.8 Manageability Only Traffic - MNGONLY (0x00005864)

Field	Bit(s)	Init.	Type	Description
EXCLUSIVE_TO_MNG	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] (Section 8.2.2.19.9).
RESERVED	31:8	0x0	RSV	Reserved

8.2.2.19.9 Manageability Decision Filters - MDEF[n] (0x00005890 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
EXACT_AND	3:0	0x0	RW	**Exact AND** Controls the inclusion of Exact MAC Address 0 to 3. In the manageability filter decision (AND section). Bit 0 corresponds to exact MAC Address 0 (MMAL0 and MMAH0), etc.
BROADCAST_AND	4	0b	RW	**Broadcast AND** Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section).

Field	Bit(s)	Init.	Type	Description
VLAN_AND	12:5	0x0	RW	**VLAN AND** Controls the inclusion of VLAN tag 0 to 7, respectively. In the manageability filter decision (AND section). Bit 5 corresponds to VLAN tag 0, etc.
IPV4_ADDRESS	16:13	0x0	RW	**IPv4 Address** Controls the inclusion of IPV4 address 0 to 3 (MIPAF[3,n]) respectively in the manageability filter decision (AND section). Bit 13 corresponds to IPV4 address 0, etc. ***Notes:*** 1. This field is relevant only if MANC.*EN_IPv4_FILTER* is set (Section 8.2.2.19.7). 2. Supported only for Network traffic.
IPV6_ADDRESS	20:17	0x0	RW	**IPv6 Address** Controls the inclusion of IPV6 address 0 to 3 respectively (MIPAF[0:3,n]) in the manageability filter decision (AND section). Bit 17 corresponds to IPV6 address 0, etc ***Notes:*** 1. Bit 20 is relevant only if MANC.*EN_IPv4_FILTER* is cleared (Section 8.2.2.19.7). 2. Supported only for Network traffic.
EXACT_OR	24:21	0x0	RW	**Exact OR** Controls the inclusion of exact MAC Address 0 to 3. In the manageability filter decision (OR section). Bit 21 corresponds to exact MAC Address 0 (MMAL0 and MMAH0), etc.
BROADCAST_OR	25	0b	RW	**Broadcast OR** Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section).
MULTICAST_AND	26	0b	RW	**Multicast OR** Controls the inclusion of Multicast address filtering in the manageability filter decision (AND section). Broadcast packets are not included by this bit.
ARP_REQUEST	27	0b	RW	**ARP Request** Controls the inclusion of ARP Request filtering in the manageability filter decision (OR section). ***Note:*** Supported only for Network traffic.
ARP_RESPONSE	28	0b	RW	**ARP Response** Controls the inclusion of ARP Response filtering in the manageability filter decision (OR section). ***Note:*** Supported only for Network traffic.
ND_134	29	0b	RW	**Neighbor Discovery 134** Controls the inclusion of Router Advertisement neighbor Discovery filtering in the manageability filter decision (OR section). ***Notes:*** 1. Supported only for Network traffic. 2. Neighbor Discovery types supported by this bit is 0x86 (134d) - Router Advertisement
PORT_0X298	30	0b	RW	**Port 0x298** Controls the inclusion of Port 0x298 filtering in the manageability filter decision (OR section). ***Note:*** Supported only for Network traffic.
PORT_0X26F	31	0b	RW	**Port 0x26F** Controls the inclusion of Port 0x26F filtering in the manageability filter decision (OR section). ***Note:*** Supported only for Network traffic.

8.2.2.19.10 Manageability IP Address Filter - MIPAF[n,m] (0x000058B0 + 0x4*n + 0x10*m, n=0...3, m=0...3)

Field	Bit(s)	Init.	Type	Description
IP_ADDR	31:0	X	RW	**Manageability IP Address Filters** For each n, m, m=0...3, n=0...3, while MANC.*EN_IPv4_FILTER* = 0, MIPAF[m,n] register holds DW 'n' of IPv6 filter 'm' (4 x IPv6 filters). For each n, m, m=0...3, n=0...3 while MANC.*EN_IPv4_FILTER* = 1, MIPAF[m,n] registers for m=0,1,2 is the same as the previous case (3 x IPv6 filters). And MIPAF[3,n] registers holds IPv4 filter 'n' (4 x IPv4 filters). ***Note:*** These registers appear in Big Endian order (LS byte, LS address is first on the wire).

8.2.2.19.11 Manageability Ethernet MAC Address Low - MMAL[n] (0x00005910 + 0x8*n, n=0...3)

Field	Bit(s)	Init.	Type	Description
MMAL	31:0	X	RW	**Manageability Ethernet MAC Address Low** The lower 32 bits of the 48-bit Ethernet MAC Address. ***Note:*** Appears in Big Endian order (LS byte of MMAL is first on the wire).

8.2.2.19.12 Manageability Ethernet MAC Address High - MMAH[n] (0x00005914 + 0x8*n, n=0...3)

Field	Bit(s)	Init.	Type	Description
MMAH	15:0	X	RW	**Manageability Ethernet MAC Address High** The upper 16 bits of the 48-bit Ethernet MAC 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.

8.2.2.19.13 FTFT Filter DW Even words - FTFT_FILTER_EVEN[n] (0x00009400 + 0x10*n, n=0...15)

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 non-masked bytes in the FTFT register.

Each 128-byte filter is composed of 32 DW entries FTFT_FILTER_ODD and FTFT_FILTER_EVEN), where each two DWs are accompanied by an 8-bit mask (FTFT_FILTER_MASK), one bit per filter byte. The mask field is set so that bit 0 in the mask masks byte 0, bit 1 masks byte 1, 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 FTFT_FILTER_LENGTH register indicates the number of bytes to compare.

Notes:

- The mask field must be 8 bytes aligned even if the length field is not 8 bytes aligned, as hardware implementation compares 8 bytes at a time. Therefore, 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 0b, indicating no comparison should be done.

- In case the actual length defined by the length field register and the mask bits is not 8 bytes aligned, there might be a case that a packet which is shorter than the actual required length passes the flexible filter. This might happen due to comparison of up to 7 bytes that come after the packet, but are not a real part of the packet.
- The last DW of each filter contains a length field defining the number of bytes from the beginning of the packet compared by this filter. If actual packet length is less than length specified by this field, the filter fails. Otherwise, it depends on the result of actual byte comparison. The value should not be greater than 128.
- FTFT registers are configured by firmware.

Field	Bit(s)	Init.	Type	Description
FTFT_FILTER0_DW0	31:0	X	RW	**FTFT Filter DW Even** Even DW filter value.

8.2.2.19.14 FTFT Filter DW Odd words - FTFT_FILTER_ODD[n] (0x00009404 + 0x10*n, n=0...15)

Field	Bit(s)	Init.	Type	Description
FTFT_FILTER0_DW1	31:0	X	RW	**FTFT Filter DW Odd** Odd DW filter value.

8.2.2.19.15 FTFT Filter Mask - FTFT_FILTER0_MASK[n] (0x00009408 + 0x10*n, n=0...15)

Field	Bit(s)	Init.	Type	Description
FTFT_FILTER0_MASK	7:0	X	RW	**FTFT Filter Mask** Mask for the 8 bytes of the filter (bit per byte)
RESERVED	31:8	0x0	RSV	Reserved.

8.2.2.19.16 FTFT Filter Length - FTFT_FILTER0_LENGTH[n] (0x0000940C + 0x10*n, n=0...15)

Field	Bit(s)	Init.	Type	Description
LENGTH	7:0	0x0	RW	**Flex Filter Length** *LENGTH* field is valid only on the last DW of the Filter all other fields are reserved
RESERVED	31:8	0x0	RSV	Reserved.

8.2.2.20 PF - Manageability (ARC Subsystem) HOST Interface Registers

Host interface to the ARC subsystem is described in the Section 11.8, "Manageability Host Interface".

8.2.2.20.1 Software Semaphore Register - SWSM (0x00010140)

Note: This register is shared by both LAN ports.

Field	Bit(s)	Init.	Type	Description
SMBI	0	0b	RW	**Semaphore Bit** This bit is set by hardware when this register is read by the device driver (one of the two PCIe functions), and cleared when the host driver writes 0b to it. The first time this register is read, the value is 0b. In the next read the value is 1b (hardware mechanism). The value remains 1b until the device driver clears it. This bit can be used as a semaphore between the two device's drivers, and is cleared on PCIe reset.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.20.2 Firmware Semaphore Register - FWSM (0x00010148)

Note: This register is shared by both LAN ports. This register should be written only by the manageability firmware. The device driver should only read this register. The firmware ignores the NVM semaphore in operating system hung states. Bits[15:0] are cleared on firmware reset.

Field	Bit(s)	Init.	Type	Description
TS_NVM_MODE	0	0b	RW	**Thermal Sensor NVM Mode** Indication to host on the Thermal Sensor NVM-based mode. 0b = TS NVM-based mode is disabled. 1b = TS NVM-based mode is enabled.
FLASH_UPDATE_INITIALIZED	1	0b	RW	**Flash Update Initialized** If set, the Flash update functionality is active.
PASS_THROUGH_INITIALIZED	2	0b	RW	**Pass-Through Initialized** If set, the pass-through functionality is active.
RESERVED	3	0b	RSV	Reserved.
HOST_INTERFACE_INITIALIZED	4	0b	RW	**Host Interface Initialized** If set, the host interface functionality is active.
OPERATION_MODE	5	0b	RW	**NVM Recovery Mode** Defines the operation mode of firmware: 0b = Normal operation mode. 1b = NVM recovery mode.
RESERVED	14:6	0x0	RSV	Reserved.
FW_VAL_BIT	15	0b	RW	**Firmware Valid Bit** Hardware clears this bit in reset de-assertion so software can know firmware modes (bits 1-4) are invalid. Firmware should set this bit to 1b when it is ready (end of boot sequence).
RESERVED	18:16	000b	RW	Reserved.

Field	Bit(s)	Init.	Type	Description
EXT_ERR_IND	24:19	0x0	RW	**External Error Indication** Firmware uses this register to store the reason that the firmware has reset/clock gated (such as NVM, patch corruption). Possible values are: 0x00 = No error. 0x01 = Unspecified error. 0x02 = No manageability (No NVM). 0x03 = TCO isolate mode active. 0x05 = Shadow RAM dump fault. 0x06 = Bad Flash contents. (for FW/RO update) 0x10 = NVM CRC error in "Test Configuration" module 0x11 = NVM CRC error in "Common FW Parameters" module 0x12 = NVM CRC error in "PT LAN 0 configuration" module 0x13 = NVM CRC error in "SideBand configuration" module 0x14 = NVM CRC error in "Flexible TCO filter configuration" module 0x15 = NVM CRC error in "PT LAN 1 configuration" module 0x16 = NVM CRC error in "OEM support structure" module 0x20 = Management memory parity/ECC error. 0x21 = SR ECC error 0x22 = NVM Firmware Module Header CRC error. 0x23 = PHY Auto-load section error. 0x3F = Reserved (max error value). All other values are reserved.
RESERVED	25	0b	RSV	Reserved.
PHY0_CONFIG__ERR_IND	26	0b	RW	**PHY0 Configuration Error Indication** Set to 1b by firmware when it fails to configure PHY of LAN0. Cleared by firmware upon successful configuration of PHY of LAN0.
PHY1_CONFIG__ERR_IND	27	0b	RW	**PHY1 Configuration Error Indication** Set to 1b by firmware when it fails to configure PHY of LAN1. Cleared by firmware upon successful configuration of PHY of LAN1
RESERVED	30:28	000b	RSV	Reserved.
FACTORY_MAC_ADDRESS_RESTORED	31	0b	RW	**Factory MAC Address Restored** When set, it indicates to software that the factory MAC Address and the current MAC Addresses are identical after the last power-up event. This bit is cleared by the device at power up.

8.2.2.20.3 Software-Firmware Synchronization - SW_FW_SYNC (0x00010160)

Each bit represents different software semaphore agreed between software and firmware as follows: Bits[4:0] and Bits[10:12] are owned by software, while Bits[9:5] and Bits[13:15] are owned by firmware. Hardware does not lock access to these bits.

Note: This register is shared by both LAN ports. For more details on software and firmware synchronization, see Section 11.8.4.

Field	Bit(s)	Init.	Type	Description
SW_NVM_SM	0	0b	RW	**Software NVM Semaphore** If set, NVM access is owned by software.
SW_PHY0_SM	1	0b	RW	**Software PHY 0 Semaphore** If set, PHY 0 access is owned by software.
SW_PHY1_SM	2	0b	RW	**Software PHY 1 Semaphore** If set, PHY 1access is owned by software.
SW_MAC_CSR_SM	3	0b	RW	**Software MAC CSR Semaphore** If set, MAC CSR access is owned by software.
RESERVED	4	0b	RSV	Reserved.
FW_NVM_SM	5	0b	RW	**Firmware NVM Semaphore** If set, NVM access is owned by firmware.
FW_PHY0_SM	6	0b	RW	**Firmware PHY 0 Semaphore** If set, PHY 0 access is owned by firmware.
FW_PHY1_SM	7	0b	RW	**Firmware PHY1 Semaphore** If set, PHY 1access is owned by firmware.
FW_MAC_CSR_SM	8	0b	RW	**Firmware MAC CSR Semaphore** If set, MAC CSR access is owned by firmware.
NVM_UPDATE_STARTED	9	0b	RW	**NVM Update Started** If set, NVM update started, software should not write to NVM.
SW_MNG_SM	10	0b	RW	**Software Manageability Semaphore** If set, Manageability host interface is owned by software.
SW_I2C0_SM	11	0b	RW	**Software I^2C 0 Semaphore** If set, I^2C of port 0 is owned by software.
SW_I2C1_SM	12	0b	RW	**Software I^2C 1 Semaphore** If set, I^2C of port 1 is owned by software.
FW_I2C0_SM	13	0b	RW	**Firmware I^2C 0 Semaphore** If set, I^2C of port 0 is owned by firmware.
FW_I2C1_SM	14	0b	RW	**Firmware I^2C 1 Semaphore** If set, I^2C of port 1 is owned by firmware.
RESERVED	30:15	0x0	RSV	Reserved.
REGSMP	31	0b	RW	**Register Semaphore** This bit is used to semaphore the access to this register between the firmware and software with no hardware enforcement. When the bit value is 0b and the register is read, the returned value is zero and the bit setting reverts to 1b (for the next read cycle). Writing 0b to this bit clears it. A software or firmware driver that reads this register and gets the value of zero for this bit, locks the access to this register until it clears this bit.

8.2.2.20.4 Host ARC Data RAM - ARCRAM[n] (0x00015800 + 0x4*n, n=0...447)

Field	Bit(s)	Init.	Type	Description
RAM_SPACE	31:0	X	RW	**RAM Space** RAM Space area that spans on the CSR space: 0x15800 - 0x15EFC.

8.2.2.20.5 HOST Interface Control Register - HICR (0x00015F00)

Field	Bit(s)	Init.	Type	Description
EN	0	0b	RW	**Enable** When set, indicates that a RAM area is provided for device driver accesses. This bit is read-only for the device driver.
C	1	0b	RW	**Command** The device driver sets this bit when it has finished putting a command block in the ARC internal data RAM. This bit should be cleared by the firmware when the command's processing is completed. Setting this bit causes an interrupt to the ARC.
SV	2	0b	RW	**Status Valid** Indicates that there is a valid status in CSR area that the device driver can read. 0b = Status not valid. 1b = Status valid. The value of the bit is valid only when the *C* bit is cleared. Only the device driver reads this bit.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.2.20.6 Firmware Resets Count - FWRESETCNT (0x00015F40)

Field	Bit(s)	Init.	Type	Description
FWRESETCNT	31:0	0x0	RO	**Firmware Resets Count** Updated by hardware. Saturates at 0xFFFF,FFFF.

8.2.2.21 PF - Time Sync (IEEE 1588) Registers

8.2.2.21.1 Time Sync SDP Configuration Register - TSSDP (0x0000003C)

This register defines the assignment of SDP pins to the time sync auxiliary capabilities.

Field	Bit(s)	Init.	Type	Description
AUX0_SDP_SEL	1:0	00b	RW	**Aux 0 SDP Select** Select one of the SDPs to serve as the trigger for auxiliary timestamp in AUXSTMPL0 and AUXSTMPH0 registers (see Section 8.2.2.21.20 and Section 8.2.2.21.21, respectively). 00b = SDP0 is assigned. 01b = SDP1 is assigned. 10b = SDP2 is assigned. 11b = SDP3 is assigned.
AUX0_TS_SDP_EN	2	0b	RW	**Aux 0 Timestamp SDP Enable** When set, indicates that one of the SDPs can be used as an external trigger to Aux timestamp 0. ***Note:*** If this bit is set to one of the SDP pins, the corresponding pin should be configured to input mode using SPD_DIR.
AUX1_SDP_SEL	4:3	00b	RW	**Aux 1 SDP Select** Select one of the SDPs to serve as the trigger for auxiliary timestamp 1 in AUXSTMPL1 and AUXSTMPH1 registers (see Section 8.2.2.21.22 and Section 8.2.2.21.23, respectively). 00b = SDP0 is assigned. 01b = SDP1 is assigned. 10b = SDP2 is assigned. 11b = SDP3 is assigned.
AUX1_TS_SDP_EN	5	0b	RW	**Aux 1 Timestamp SDP Enable** When set indicates that one of the SDPs can be used as an external trigger to Aux timestamp 1. ***Note:*** If this bit is set to one of the SDP pins, the corresponding pin should be configured to input mode using SPD_DIR.
TS_SDP0_SEL	7:6	00b	RW	**Timestamp SDP 0 Select** SDP0 allocation to Tsync event. When *TS_SDP0_EN* is set, these bits select the Tsync event that is routed to SDP0. 00b = Target Time 0 is output on SDP0. 01b = Target Time 1 is output on SDP0. 10b = Freq Clock 0 is output on SDP0. 11b = Freq Clock 1 is output on SDP0.
TS_SDP0_EN	8	0b	RW	**Timestamp SDP 0 Enable** When set, indicates that SDP0 is assigned to Tsync.
TS_SDP1_SEL	10:9	00b	RW	**Timestamp SDP 1 Select** SDP1 allocation to Tsync event. When *TS_SDP1_EN* is set, these bits select the Tsync event that is routed to SDP1. 00b = Target Time 0 is output on SDP1. 01b = Target Time 1 is output on SDP1. 10b = Freq Clock 0 is output on SDP1. 11b = Freq Clock 1 is output on SDP1.
TS_SDP1_EN	11	0b	RW	**Timestamp SDP 1 Enable** When set, indicates that SDP1 is assigned to Tsync.

Field	Bit(s)	Init.	Type	Description
TS_SDP2_SEL	13:12	00b	RW	**Timestamp SDP 2 Select** SDP2 allocation to Tsync event. When *TS_SDP2_EN* is set, these bits select the Tsync event that is routed to SDP2. 00b = Target Time 0 is output on SDP2. 01b = Target Time 1 is output on SDP2. 10b = Freq Clock 0 is output on SDP2. 11b = Freq Clock 1 is output on SDP2.
TS_SDP2_EN	14	0b	RW	**Timestamp SDP 2 Enable** When set, indicates that SDP2 is assigned to Tsync.
TS_SDP3_SEL	16:15	00b	RW	**Timestamp SDP 3 Select** SDP3 allocation to Tsync event. When *TS_SDP3_EN* is set, these bits select the Tsync event that is routed to SDP3. 00b = Target Time 0 is output on SDP3. 01b = Target Time 1 is output on SDP3. 10b = Freq Clock 0 is output on SDP3. 11b = Freq Clock 1 is output on SDP3.
TS_SDP3_EN	17	0b	RW	**Timestamp SDP 3 Enable** When set, indicates that SDP3 is assigned to Tsync.
RESERVED	31:18	0x0	RSV	Reserved. Write 0, ignore on read.

8.2.2.21.2 Rx Message Type Register Low - RXMTRL (0x00005120)

Field	Bit(s)	Init.	Type	Description
CTRLT	7:0	0x0	RW	**Control Timestamp** V1 control to timestamp.
MSGT	15:8	0x0	RW	**Message Timestamp** V2 message type to timestamp.
UDPT	31:16	0x13F	RW	**UDP Timestamp** UDP port number to timestamp.

8.2.2.21.3 Rx Time Sync Control Register - TSYNCRXCTL (0x00005188)

Field	Bit(s)	Init.	Type	Description
RXTT	0	0b	RO	**Rx Timestamp Valid** This bit is set to 1b when a valid value for Rx timestamp is captured in the Rx timestamp register. This bit is cleared by read of Rx timestamp register RXSTMPH (see Section 8.2.2.21.4).
TYPE	3:1	0x0	RW	**Type** Type of packets to timestamp. 000b = Timestamp L2 (V2) packets only (Sync or Delay_req depends on RXMTRL.*MSGT* and packets with message ID 2 and 3). 001b = Timestamp L4 (V1) packets only (Sync or Delay_req depends on RXMTRL.*CTRLT*). 010b = Timestamp V2 (L2 and L4) packets (Sync or Delay_req depends on RXMTRL.*MSGT* and packets with message ID 2 and 3). 100b = Timestamp all packets. 101b = Timestamp V2 packets in which message ID bit 3 is zero, which means timestamp all event packets. All other values are reserved.
EN	4	0b	RW	**Enable Rx Timestamp** 0b = Time stamping to TXSTMPH/L disabled. 1b = Time stamping to TXSTMPH/L enabled.

Field	Bit(s)	Init.	Type	Description
RESERVED	22:5	0x0	RSV	Reserved.
TSIP_UT_EN	23	0b	RW	**TSIP UT Enable** Defines if untagged packets are appended a timestamp.
TSIP_UP_EN	31:24	0x0	RW	**TSIP UP Enable** Defines which UP timestamp is appended to the received packet (per UP bitmap). For example, to require a timestamp on packets received with UP = 0 or UP = 3, bits 24 and 26 should be set.

8.2.2.21.4 Rx Timestamp High - RXSTMPH (0x000051A4)

Field	Bit(s)	Init.	Type	Description
RXSTMPH	31:0	0x0	RO	**Rx Timestamp High** Rx timestamp MSB value defined in seconds.

8.2.2.21.5 Rx Timestamp Low - RXSTMPL (0x000051E8)

Field	Bit(s)	Init.	Type	Description
RXSTMPL	29:0	0x0	RO	**Rx Timestamp Low** Rx timestamp LSB value defined in ns units. The value in this field wraps around at 999999999 decimal.
RESREVED	31:30	00b	RO	Reserved.

8.2.2.21.6 Tx Time Sync Control Register - TSYNCTXCTL (0x00008C00)

Field	Bit(s)	Init.	Type	Description
TXTT	0	0b	ROS	**Tx Timestamp Valid** This bit is set to 1b when a valid value for Tx timestamp is captured in the Tx timestamp register. This bit is cleared by read of Tx timestamp register TXSTMPH (see Section 8.2.2.21.8).
RESERVED	3:1	000b	RSV	Reserved.
EN	4	0b	RW	**Enable Tx Timestamp** 0x0 = time stamping disabled. 0x1 = time stamping enabled.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.2.21.7 Tx Timestamp Value Low - TXSTMPL (0x00008C04)

Field	Bit(s)	Init.	Type	Description
TXSTMPL	29:0	0x0	RO	**Tx Timestamp Low** Tx timestamp LSB value defined in ns units. The value in this field wraps around at 999999999 decimal.
RESERVED	31:30	00b	RSV	Reserved.

8.2.2.21.8 Tx Timestamp Value High - TXSTMPH (0x00008C08)

Field	Bit(s)	Init.	Type	Description
TXSTMPH	31:0	0x0	RO	**Tx Timestamp High** Tx timestamp MSB value defined in seconds.

8.2.2.21.9 System Time Register Low - SYSTIMEL (0x00008C0C)

Field	Bit(s)	Init.	Type	Description
STL	29:0	0x0	RW	**System Time Low** System time LSB register defined in ns units. Writes of values above one second (above 0x3B9AC9FF) are ignored.
RESERVED	31:30	00b	RW	Reserved.

8.2.2.21.10 System Time Register High - SYSTIMEH (0x00008C10)

Field	Bit(s)	Init.	Type	Description
STH	31:0	0x0	RW	**System Time High** System time MSB register defined in seconds.

8.2.2.21.11 Increment Attributes Register - TIMEINCA (0x00008C14)

Field	Bit(s)	Init.	Type	Description
INCVALUE	30:0	0x0	RW	**Increment Value** Increment value to the SYSTIME registers on each 12.5 ns. This field is used to correct a fixed clock drift and should be kept at zero for an ideal clock.
ISGN	31	0b	RW	**Increment Sign** 0b = Each 12.5 ns cycle add to SYSTIME a value of 12.5 ns + *INCVALUE* * 2^-32 ns. 1b = Each 12.5 ns cycle add to SYSTIME a value of 12.5 ns - *INCVALUE* * 2^-32 ns.

8.2.2.21.12 Time Adjustment Offset Register - TIMADJ (0x00008C18)

Field	Bit(s)	Init.	Type	Description
TADJL	30:0	0x0	RW	**Time Adjustment** Time adjustment value defined in ns units.
SIGN	31	0b	RW	**Sign** 0b = ”+” 1b = ”-”

8.2.2.21.13 TimeSync Auxiliary Control Register - TSAUXC (0x00008C20)

Field	Bit(s)	Init.	Type	Description
EN_TT0	0	0b	RW	**Enable Target Time 0** Enable bit is set by software to 1b to enable pulse or level change generation as a function of the *PLSG0* bit in this register. This bit is not auto-cleared by the hardware.
EN_TT1	1	0b	RW	**Enable Target Time 1** Enable bit is set by software to 1b to enable pulse or level change generation as a function of the *PLSG0* bit in this register. This bit is not auto-cleared by the hardware.
EN_CLK0	2	0b	RW	**Enable Configurable Frequency Clock 0** Clock is generated according to frequency defined in the FREQOUT0 register on the SDP pin (0 to 3) that has both: 1. TSSDP.*TS_SDPx_SEL* field with a value of 10b. 2. TSSDP.*TS_SDPx_EN* value of 1b.
SAMP_AUT0	3	0b	SC	**Sample Auxiliary Timestamp 0** When this flag is set, the SYSTIMEL/H registers are latched to the AUXSTMPL0/ AUXSTMPH0 registers. Then, the SAMP_AUT0 flag is auto-cleared by the hardware.
ST0	4	0b	RW	**Start Clock 0 Toggle on Target Time 0** Enable Clock 0 toggle only after target time 0 (as defined in the TRGTTIML0 and TRGTTIMH0 registers) has passed.
EN_CLK1	5	0b	RW	**Enable Configurable Frequency Clock 1** Clock is generated according to frequency defined in the FREQOUT1 register on the SDP pin (0 to 3) that has both: 1. TSSDP.*TS_SDPx_SEL* field with a value of 10b. 2. TSSDP.*TS_SDPx_EN* value of 1b.
SAMP_AUT1	6	0b	SC	**Sample Auxiliary Timestamp 1** When this flag is set, SYSTIMEL/H registers are latched to the AUXSTMPL1/ AUXSTMPH1 registers. Then, the SAMP_AUT1 flag is auto-cleared by the hardware.
ST1	7	0b	RW	**Start Clock 1 Toggle on Target Time 1** Enable Clock 1 toggle only after Target Time 1 (as defined in the TRGTTIML1 and TRGTTIMH1 registers) has passed.
EN_TS0	8	0b	RW	**Enable Hardware Timestamp 0** Enable time-stamping occurrence of change in SDP pin into the AUXSTMPL0 and AUXSTMPH0 registers. SDP pin (0 to 3) is selected for time-stamping if the SDP pin is selected via the TSSDP.*AUX0_SDP_SEL* field and the TSSDP.*AUX0_TS_SDP_EN* bit is set to 1b.
AUTT0	9	0b	RW	**Auxiliary Timestamp Taken 0** This is a read-only bit field. At 0b the Auxiliary Timestamp 0 is enabled. This bit is set to 1b by hardware following a level change of the input SDP. Reading the AUXSTMPH0 register clears this bit.
EN_TS1	10	0b	RW	Enable hardware time stamp 1
AUTT1	11	0b	RW	**Auxiliary Timestamp Taken 1** This is a read only bit field. At 0b the Auxiliary Time Stamp 1 is enabled. This bit is set to 1b by hardware following a level change of the input SDP. Reading the AUXSTMPH1 register clears this bit.
RESERVED	16:12	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
PLSG0	17	0b	RW	**Pulse Generate** Use Target Time 0 to generate start of pulse and Target Time 1 to generate end of pulse. SDP pin selected to drive pulse or level change is set according to the TSSDP.*TS_SDPx_SEL* field with a value of 00b, and TSSDP.*TS_SDPx_EN* bit with a value of 1b (see Section 8.2.2.21.1). 0b = Target Time 0 generates change in SDP level. 1b = Target time 0 generates start of pulse on SDP pin. ***Note:*** Pulse or level change is generated when the *EN_TT0* bit is set to 1b.
RESERVED	29:18	0x0	RSV	Reserved.
RESREVED	30	1b	RSV	Reserved.
DISABLE_SYSTIME	31	1b	RW	**Disable SYSTIME count operation** 0b = SYSTIME timer activated. 1b = SYSTIME timer disabled. Value of SYSTIMEH, SYSTIMEL and SYSTIMER remains constant.

8.2.2.21.14 Target Time Register 0 Low - TRGTTIMEL0 (0x00008C24)

Field	Bit(s)	Init.	Type	Description
TTL	29:0	0x0	RW	**Target Time Low** Target Time 0 LSB register defined in ns units.
RESERVED	31:30	00b	RW	Reserved.

8.2.2.21.15 Target Time Register 0 High - TRGTTIMEH0 (0x00008C28)

Field	Bit(s)	Init.	Type	Description
TTH	31:0	0x0	RW	**Target Time High** Target Time 0 MSB register defined in seconds.

8.2.2.21.16 Target Time Register 1 Low - TRGTTIMEL1 (0x00008C2C)

Field	Bit(s)	Init.	Type	Description
TTL	29:0	0x0	RW	**Target Time Low** Target Time 1 LSB register defined in ns units.
RESERVED	31:30	00b	RW	Reserved.

8.2.2.21.17 Target Time Register 1 High - TRGTTIMEH1 (0x00008C30)

Field	Bit(s)	Init.	Type	Description
TTH	31:0	0x0	RW	**Target Time High** Target Time 1 MSB register defined in seconds.

8.2.2.21.18 Frequency Out 0 Control Register - FREQOUT0 (0x00008C34)

Field	Bit(s)	Init.	Type	Description
CHCT	29:0	0x0	RW	**Clock Out Half Cycle Time** Defines the Half Cycle time of Clock 0 in ns units. When clock output is enabled, permitted values are any value larger than 25 and any value below 1 second.
RESERVED	31:30	00b	RSV	Reserved

8.2.2.21.19 Frequency Out 1 Control Register - FREQOUT1 (0x00008C38)

Field	Bit(s)	Init.	Type	Description
CHCT	29:0	0x0	RW	**Clock Out Half Cycle Time** Defines the Half Cycle time of Clock 1 in ns units. When clock output is enabled, permitted values are any value larger than 13 and up to including 999,999,900 decimal (slightly below 1 second).
RESERVED	31:30	00b	RSV	Reserved.

8.2.2.21.20 Auxiliary Timestamp 0 Register Low - AUXSTMPL0 (0x00008C3C)

Field	Bit(s)	Init.	Type	Description
TST_LOW	29:0	0x0	RO	**Timestamp Low** Auxiliary Time Stamp 0 LSB value defined in ns units.
RESERVED	31:30	00b	RO	Reserved

8.2.2.21.21 Auxiliary Timestamp 0 Register High - AUXSTMPH0 (0x00008C40)

Field	Bit(s)	Init.	Type	Description
TST_HI	31:0	0x0	RO	**Timestamp High** Auxiliary Time Stamp 0 MSB value defined in seconds.

8.2.2.21.22 Auxiliary Timestamp 1 Register Low - AUXSTMPL1 (0x00008C44)

Field	Bit(s)	Init.	Type	Description
TST_LOW	29:0	0x0	RO	**Timestamp Low** Auxiliary Time Stamp 1 LSB value defined in ns units.
RESERVED	31:30	00b	RO	Reserved

8.2.2.21.23 Auxiliary Timestamp 1 Register High - AUXSTMPH1 (0x00008C48)

Field	Bit(s)	Init.	Type	Description
TST_HI	31:0	0x0	RO	**Timestamp High** Auxiliary Time Stamp 1 MSB value defined in seconds.

8.2.2.21.24 System Time Register Residue - SYSTIMR (0x00008C58)

Field	Bit(s)	Init.	Type	Description
SYSTIMR	31:0	0x0	RW	**System Time Residue** System time Residue value defined in 2^(-32) ns resolution.

8.2.2.21.25 Time Sync Interrupt Cause Register - TSICR (0x00008C60)

Note: Value of register is always read as 0x0. Once EICR.*TIMESYNC* is set, internal value of register should be cleared by write 1 to all bits, or cleared by read to enable reception of an additional EICR.*TIMESYNC* interrupt.

Field	Bit(s)	Init.	Type	Description
SYS_WRAP	0	0b	RW1C	**SYSTIMEL Wraparound** Set when SYSTIME ns counter wraps around (SYSTIMEL). Wraparound occurrence can be used by software to update software time. This event should happen every second.
TXTS	1	0b	RW1C	**Transmit Timestamp** Set when new timestamp is loaded into TXSTMP register
RXTS	2	0b	RW1C	**Receive Timestamp** Set when new timestamp is loaded into RXSTMP register
TT0	3	0b	RW1C	**Target Time 0 Trigger** Set when Target Time 0 (TRGTTIML/H0) trigger occurs.
TT1	4	0b	RW1C	**Target Time 1 Trigger** Set when Target Time 1 (TRGTTIML/H1) trigger occurs.
AUTT0	5	0b	RW1C	**Auxiliary Timestamp 0 Taken** Set when new timestamp is loaded into AUXSTMP 0 (auxiliary timestamp 0) register.
AUTT1	6	0b	RW1C	**Auxiliary Timestamp 1 Taken** Set when new timestamp is loaded into AUXSTMP 1 (auxiliary timestamp 1) register.
TADJ	7	0b	RW1C	**Time Adjust 0 Done** Set when Time Adjust to clock out 0 or 1 completed
RESERVED	31:8	0x0	RSV	Reserved. Write 0, ignore on read.

8.2.2.21.26 Time Sync Interrupt Mask Register - TSIM (0x00008C68)

Field	Bit(s)	Init.	Type	Description
SYS_WRAP	0	0b	RW	**SYSTIMEL Wraparound Mask** 0b = No Interrupt generated when TSICR.*SYS_WRAP* is set. 1b = Interrupt generated when TSICR.*SYS_WRAP* is set.
TXTS	1	0b	RW	**Transmit Timestamp Mask** 0b = No Interrupt generated when TSICR.*TXTS* is set. 1b = Interrupt generated when TSICR.*TXTS* is set.
RXTS	2	0b	RW	**Receive Timestamp Mask** 0b = No Interrupt generated when TSICR.*RXTS* is set. 1b = Interrupt generated when TSICR.*RXTS* is set.
TT0	3	0b	RW	**Target Time 0 Trigger Mask** 0b = No Interrupt generated when TSICR.*TT0* is set. 1b = Interrupt generated when TSICR.*TT0* is set.
TT1	4	0b	RW	**Target Time 1 Trigger Mask** 0b = No Interrupt generated when TSICR.*TT1* is set. 1b = Interrupt generated when TSICR.*TT1* is set.
AUTT0	5	0b	RW	**Auxiliary Timestamp 0 Taken Mask** 0b = No Interrupt generated when TSICR.*AUTT0* is set. 1b = Interrupt generated when TSICR.*AUTT0* is set.
AUTT1	6	0b	RW	**Auxiliary Timestamp 1 Taken Mask** 0b = No Interrupt generated when TSICR.*AUTT1* is set. 1b = Interrupt generated when TSICR.*AUTT1* is set.
TADJ	7	0b	RW	**Time Adjust 0 Done Mask** 0b = No Interrupt generated when TSICR.*TADJ* is set. 1b = Interrupt generated when TSICR.*TADJ* is set.
RESERVED	31:8	0x0	RSV	Reserved. Write 0, ignore on read.

8.2.2.22 PF - Virtualization PF Registers

8.2.2.22.1 PF VFLR Events Clear - PFVFLREC[n] (0x00000700 + 0x4*n, n=0...1)

Field	Bit(s)	Init.	Type	Description
CLEAR_VFLE	31:0	0x0	RW1C	**Clear VLFR Events** When set, bit 'i' in register 'n' reflects an FLR event on VF# 32*n+i. These bits are accessible only to the PF and are cleared by writing 1.

8.2.2.22.2 PF Mailbox Interrupt Causes Register - PFMBICR[n] (0x00000710 + 0x4*n, n=0...3)

Each register handles 16 VFs as defined here.

Field	Bit(s)	Init.	Type	Description
VFREQ	15:0	0x0	RW1C	**VF Requested** Each bit in the *VFREQ* field is set when VF number (16*n+j) wrote a message in its mailbox, where 'n' is the register index (n=0...3) and 'j' is the index of the bits in the VFREQ (j=0...15).
VFACK	31:16	0x0	RW1C	**VF Acknowledged** Each bit in the *VFACK* field is set when VF number (16*n+j) acknowledged a PF message, where 'n' is the register index (n=0...3) and '16+j' is the index of the bits in the VFACK (j=0...15).

8.2.2.22.3 PF Mailbox Interrupt Mask Register - PFMBIMR[n] (0x00000720 + 0x4*n, n=0...1)

Field	Bit(s)	Init.	Type	Description
VFIM	31:0	0xFFFFFFFF	RW	**VF Interrupt Mask** Bit j - Mailbox indication from VF number (32*n+j) might cause an interrupt to the PF.

8.2.2.22.4 PF Queue Drop Enable Register - PFQDE (0x00002F04)

Field	Bit(s)	Init.	Type	Description
PFQDE	0	0b	RW	**PF Queue Drop Enable** Enable drop of packets from Rx queue *QUEUE_INDEX*. This bit overrides the SRRCTL.*DROP_EN* bit (Section 8.2.2.9.5) of each queue. In other words, if either of the bits is set, a packet received when no descriptor is available is dropped.
HIDE_VLAN	1	0b	RW	**Hide VLAN** If this bit is set, the RXDCTL.*VME* setting is ignored (Section 8.2.2.9.7), the VLAN is always stripped, a value of zero is written in the *RDESC.VLAN* tag and in the *RDESC.STATUS.VP* fields of the received descriptor.
STRIP_TAG	2	0b	RW	**Strip Tag** If this bit is set, the E-tag is stripped from the packet. If only one type of tag should be stripped, the other tag's EtherType should be invalidated using the ETAG_ETYPE.*VALID* field (Section 8.2.2.8.5).
RESERVED	7:3	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
QUEUE_INDEX	14:8	0x0	RW	**Queue Index** Indicates the queue referenced upon *WE*/*RE* commands.
RESERVED	15	0b	RSV	Reserved.
WE	16	0b	RW	**Write Enable** When this bit is set, the content of Bits[2:0] are written into the relevant queue context. Bit[3] is reserved. This bit should never be set together with the *RE* bit in this register.
RE	17	0b	RW	**Read Enable** When this bit is set, the content of Bits[2:0] are read from the relevant queue context. Bit[3] is reserved. This bit should never be set together with the *WE* bit in this register.
RESERVED	31:18	0x0	RSV	Reserved.

8.2.2.22.5 Last Malicious VM - Rx - LMVM_RX (0x00002FA4)

Field	Bit(s)	Init.	Type	Description
MALICIOUS_QUEUE	6:0	0x0	RW	**Malicious Queue** The Rx queue on which the malicious behavior reported in LVMMC was detected. The queue number is the absolute number in the PF space.
RESERVED	16:7	0x0	RSV	Reserved.
MAL_PF	17	0b	RW	**Malicious Driver on PF** Malicious driver behavior detected on current PF. 0b = Malicious event was on a queue belonging to a VF. 1b = Malicious event was on a queue belonging to the PF.
RESERVED	31:18	0x0	RSV	Reserved.

8.2.2.22.6 Last VM Misbehavior Cause - Rx - LVMMC_RX (0x00002FA8)

Bits in the LVMMC_RX register define the cause for blocking the malicious queue that was reported in the LMVM_RX.*MALICIOUS_QUEUE* field when RDRXCTL.*MDP_EN* is set. For details of the different bits, refer to Section 7.7.11.3, "Malicious Driver Detection".

Field	Bit(s)	Init.	Type	Description
INV_MACC	0	0b	RC	**Invalid Memory Access** A PCIe DMA access initiated by a VF ended with Unsupported Request (UR) or Completer Abort (CA), or a read access was blocked due to out of range address. When a Malicious DMA access is detected, the Rx queue (VF) that initiated the access is disabled and corresponding WQBR_RX bit is set.
INVALID_RXQ_CONTEXT	1	0b	RC	**Invalid Receive Queue Context** An invalid receive queue context was detected when queue was enabled, or an attempt to change the static part of the queue context on the fly was detected.
RESERVED	31:2	0x0	RSV	Reserved.

8.2.2.22.7 Wrong Queue Behavior Register - Rx - WQBR_RX[n] (0x00002FB0 + 0x4*n, n=0...3)

Field	Bit(s)	Init.	Type	Description
WVBR	31:0	0x0	RW1C	**Wrong Behavior Receive Queue** Bitmap indicating against which Rx queue a malicious action was taken. The queue is released only by a reset of the VF or by clearing the corresponding bit in WQBR_RX register.

8.2.2.22.8 PF Mailbox - PFMAILBOX[n] (0x00004B00 + 0x4*n, n=0...63)

Field	Bit(s)	Init.	Type	Description
STS	0	0b	WO	**Status** Status/Command from PF ready. Setting this bit, causes an interrupt to the relevant VF. This bit always read as zero. Setting this bit sets the *PFSTS* bit in the VFMAILBOX register (Section 8.3.2.1.5).
ACK	1	0b	WO	**Acknowledge** VF message received. Setting this bit, causes an interrupt to the relevant VF. This bit always read as zero. Setting this bit sets the *PFACK* bit in VFMAILBOX register.
VFU	2	0b	RW	**VFU** Buffer is taken by VF. This bit is RO for the PF and is a mirror of the *VFU* bit of the VFMAILBOX register.
PFU	3	0b	RW	**PFU** Buffer is taken by PF. This bit can be set only if the *VFU* bit is cleared and is mirrored in the *PFU* bit of the VFMAILBOX register.
RVFU	4	0b	WO	**Reset VFU** Setting this bit clears the *VFU* bit in the corresponding VFMAILBOX register. This bit should be used only if the VF driver is stuck. Setting this bit also resets the corresponding bits in the *VFREQ* and *VFACK* fields of the PFMBICR register (Section 8.2.2.22.2).
RESERVED	31:5	0x0	RSV	Reserved.

8.2.2.22.9 Filter Local Packets Low - PFFLPL (0x000050B0)

Field	Bit(s)	Init.	Type	Description
FLP	31:0	0x0	RW	**Filter Local Packets** Filter incoming packets whose MAC source address matches one of the LAN port DA MAC Addresses. If the SA of the received packet matches one of the DA in the RAH/RAL registers, the VM tied to this DA does not receive the packet. Other VMs can still receive it. (Bit per Pool)

8.2.2.22.10 Filter Local Packets High - PFFLPH (0x000050B4)

Field	Bit(s)	Init.	Type	Description
FLP	31:0	0x0	RW	**Filter Local Packets** Filter incoming packets whose MAC source address matches one of the LAN port DA MAC Addresses. If the SA of the received packet matches one of the DA in the RAH/RAL registers, the VM tied to this DA does not receive the packet. Other VMs can still receive it. (Bit per Pool)

8.2.2.22.11 PF VM Tx Switch Loopback Enable - PFVMTXSW[n] (0x00005180 + 0x4*n, n=0...1)

Field	Bit(s)	Init.	Type	Description
LLE	31:0	0x0	RW	**Local Loopback Enable** For each register 'n', and bit 'i' (where i=0..31), enables Local loopback for pool 32*n+1. When set, a packet originating from a specific pool and destined to the same pool is allowed to be looped back. If cleared, the packet is dropped.

8.2.2.22.12 PF Virtual Control Register - PFVTCTL (0x000051B0)

Field	Bit(s)	Init.	Type	Description
VT_ENA	0	0b	RW	**Virtualization Enabled Mode** 0b = Rx traffic is handled internally as if it belongs to VF zero while VF zero is enabled. 1b = The X550 supports either 16, 32, or 64 Pools. This bit should be set the same as MTQC.*VT_ENA* (see Section 8.2.2.10.14).
RESERVED	6:1	0x0	RSV	Reserved.
DEF_PL	12:7	0x0	RW	**Default Pool** Pool assignment for packets that do not pass any pool queuing decision. Enabled by the *DIS_DEF_POOL* bit.
RESERVED	15:13	000b	RSV	Reserved.
POOLING_MODE	17:16	00b	RW	**Pooling Mode** 00b = Pool select by MAC Address. 01b = Pool select by MAC Address or E-tag. 10b = Reserved. 11b = Reserved.
RESERVED	28:18	0x0	RSV	Reserved.
DIS_DEF_POOL	29	0b	RW	**Disable Default Pool** Determines the behavior of an Rx packet that does not match any Rx filter and is therefore not allocated a destination pool. 0b = Packet is assigned to the Default Pool (see DEF_PL above). 1b = Packet is dropped.
RPL_EN	30	0b	RW	**Replication Enable** Replication is enabled when set to 1b. When set, MRQC.*MRQE* should be set to one of the virtualization modes (1000b - 1111b)
RESERVED	31	0b	RSV	Reserved.

8.2.2.22.13 PF VF Receive Enable - PFVFRE[n] (0x000051E0 + 0x4*n, n=0...1)

This register is reset on common reset cases, and on per-function reset cases. Respective bits per VF are reset on VFLR, *BME* bit clear, or on VF software reset. For more information, see Section 8.2.2.22.16.

Field	Bit(s)	Init.	Type	Description
PFVFRE	31:0	0x0	RW	**PF VF Receive Enable** Bit j - Enables receiving packets to VF# (32*n+j). Each bit is cleared by the relevant VFLR.

8.2.2.22.14 PF VM VLAN Insert Register - PFVMVIR[n] (0x00008000 + 0x4*n, n=0...63)

Field	Bit(s)	Init.	Type	Description
PORT_VLAN_ID	15:0	0x0	RW	**Port VLAN ID** Port VLAN tag to insert if the *VLANA* field = 01b.
RESERVED	26:16	0x0	RSV	Reserved.
TAGA	28:27	00b	RW	**Tag Action** 00b = Do not insert any outer tag. 01b = Insert an E-tag. 10b = Reserved. 11b = Reserved.
RESERVED	29	0b	RSV	Reserved.
VLANA	31:30	00b	RW	**VLAN Action** 00b = Use descriptor command. 01b = Always insert Default VLAN. 10b = Never insert VLAN. 11b = Reserved.

8.2.2.22.15 Last VM Misbehavior Cause - Tx - LVMMC_TX (0x00008108)

Bits in this register define the cause for blocking the malicious queue that was reported in the LMVM_TX.*MALICIOUS_QUEUE* field (Section 8.2.2.22.17) when the DMATXCTL.*MDP_EN* bit is set (Section 8.2.2.10.2). For details of the different bits, refer to Section 7.7.11.3, "Malicious Driver Detection".

Note: Only the first malicious event is registered for each packet. Therefore, if a bit is not set, it does not mean that this event did not occur, only that another malicious behavior was detected first.

Field	Bit(s)	Init.	Type	Description
MAC_HEADER	0	0b	RC	**MAC Header** Illegal MAC header size.
IPV4_HEADER	1	0b	RC	**IPv4 Header** Illegal IPv4 header size.
IPV6_HEADER	2	0b	RC	**IPv6 Header** Illegal IPv6 header size.
WRONG_MAC_IP	3	0b	RC	**Wrong MAC IP** Wrong MAC +IP header size.
TCP_LSO	4	0b	RC	**TCP Long Send Operation** Illegal TCP header was detected in a large send operation.
IPSEC_OFFLOAD	5	0b	RC	**IPsec Offload** A VF requested an IPsec offload.
UDP_LSO	6	0b	RC	**UDP Long Send Operation** Illegal UDP header was detected in a large send operation.
STCP_CS	7	0b	RC	**STCP Checksum** Illegal STCP header was detected in an SCTP checksum offload operation.
SIZE	8	0b	RC	**Size** Illegal packet size (> 15.5K).
RESERVED	9	0b	RSV	Reserved.
OFF_ILL	10	0b	RC	**Offload Illegal** Illegal offload request.

Field	Bit(s)	Init.	Type	Description
SCTP_ALIGNED	11	0b	RC	**SCTP Aligned** SCTP CRC request of non 4 byte aligned data.
ZERO_MSS	12	0b	RC	**Zero MSS** A request for large send with zero MSS was detected.
CONTEXT_IN_PACKET	13	0b	RC	**Context in Packet** A context descriptor was detected in the middle of a packet.
LSO_MORE_THAN_4	14	0b	RC	**Large Send Operation More Than Four** Large send with more than 4 header buffers misbehavior was detected.
OOS_SSO	15	0b	RC	**Out of Synch Single Send Operation** Out of sync during Single Send.
OOS_LSO	16	0b	RC	**Out of Synch Large Send Operation** Out of sync during Large Send.
SSO_UDP	17	0b	RC	**Single Send Operation UDP** Wrong parameter of headers for UDP SSO.
SSO_TCP	18	0b	RC	**Single Send Operation TCP** Wrong parameter of headers for TCP SSO.
INV_MACC	19	0b	RC	**Invalid Memory Access** A PCIe DMA access initiated by a VF ended with Unsupported Request (UR) or Completer Abort (CA), or was prevented from being sent if out of range. When a Malicious DMA access is detected the Tx queue (VF) that initiated the access is disabled and corresponding WQBR_TX bit is set.
DESC_TYPE	20	0b	RC	**Descriptor Type** Wrong descriptor type (other than 2,3) or *CC* bit not set in virtualization mode.
WRONG_NULL	21	0b	RC	**Wrong Null** Null without EOP.
NO_EOP	22	0b	RC	**No End of Packet** Packet without EOP (i.e. bigger than the ring size).
CONTEXT_BURST	23	0b	RC	**Context Burst** Contiguous context descriptor burst that exceeds 3 Contexts.
RESERVED	24	0b	RSV	Reserved.
MAC_VLAN_SPOOF	25	0b	RC	**MAC/VLAN Spoof** A MAC spoof or VLAN spoof attempt was detected.
VLAN_IERR	26	0b	RC	**VLAN Insertion Error** *VLE* bit set in Tx descriptor when VMVIR[n].*VLANA* register field is not 0 causing drop of Tx packets.
LEGACY_DESC_IOV	27	0b	RC	**Legacy Descriptor IOV** A legacy descriptor in IOV was detected.
FCOE_OFFLOAD	28	0b	RC	**FCoE Offload** A VF requested an FCoE offload.
INVALID_TXQ_CONTEXT	29	0b	RC	**Invalid Transmit Queue Context** An invalid transmit queue context was detected when queue was enabled, or an attempt to change the static part of the queue context on the fly was detected.
RESERVED	31:30	00b	RSV	Reserved.

8.2.2.22.16 PF VF Transmit Enable - PFVFTE[n] (0x00008110 + 0x4*n, n=0...1)

This register is reset on common reset cases, and on per-function reset cases. Respective bits per VF are reset on VFLR, *BME* bit clear, or on VF software reset. For more information, see Section 8.2.2.22.13.

Field	Bit(s)	Init.	Type	Description
PFVFTE	31:0	0x0	RW	**PF VF Transmit Enable** Bit j - Enables transmitting packets from VF# (32*n+j). Each bit is cleared by the relevant VFLR.

8.2.2.22.17 Last Malicious VM - Tx - LMVM_TX (0x00008124)

Field	Bit(s)	Init.	Type	Description
MALICIOUS_QUEUE	6:0	0x0	RW	**Malicious Queue** The Tx queue on which the malicious behavior reported in LVMMC was detected. The queue number is the absolute number in the PF space.
RESERVED	16:7	0x0	RSV	Reserved.
MAL_PF	17	0b	RW	**Malicious PF** Malicious driver behavior detected on current PF. 0b = Malicious event was on a queue belonging to a VF. 1b = Malicious event was on a queue belonging to the PF.
RESERVED	31:18	0x0	RSV	Reserved.

8.2.2.22.18 Wrong Queue Behavior Register - Tx - WQBR_TX[n] (0x00008130 + 0x4*n, n=0...3)

Field	Bit(s)	Init.	Type	Description
WVBR	31:0	0x0	RW1C	**Wrong Behavior Transmit Queue** Bitmap indicating against which Tx queue an anti-spoof action or malicious action was taken. The queue is released only by a reset of the VF or by clearing the corresponding bit in the WQBR_TX register

8.2.2.22.19 PFVF Anti-Spoof Control - PFVFSPOOF[n] (0x00008200 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
MACAS	7:0	0x0	RW	**MAC Anti-Spoof** For each register 'n', and bit 'i' (where i=0..7), enables anti-spoofing filter on Ethernet MAC Addresses for VF(8*n+i).
VLANAS	15:8	0x0	RW	**VLAN Anti-Spoof** For each register 'n', and bit '8+i' (where i=0..7), enables anti-spoofing filter on VLAN tag for VF(8*n+i). ***Note:*** If *VLANAS* is set for a specific pool, the respective *MACAS* bit must be set as well.
ETHERTYPEAS	23:16	0x0	RW	**EtherType Anti-Spoof** For each register 'n', and bit '16+i' (where i=0..7), enables anti-spoofing filter on EtherType for VF(8*n+i).

Field	Bit(s)	Init.	Type	Description
ETHERTYPELB	31:24	0x0	RW	**EtherType Loopback** For each register 'n', and bit '24+i' (where i=0..7), enables loopback filter on EtherType for VF(8*n+i).

8.2.2.22.20 PF DMA Tx General Switch Control - PFDTXGSWC (0x00008220)

Field	Bit(s)	Init.	Type	Description
LBE	0	0b	RW	**Loopback Enable** Enables VMDQ loopback.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.2.22.21 PF VM 0:31 Error Count Mask - PFVMECM0 (0x00008790)

Field	Bit(s)	Init.	Type	Description
FILTER	31:0	0x0	RW	**Filter** Defines if a packet dropped from pools 0 to 31, respectively, is counted in the SSVPC counter.

8.2.2.22.22 PF VM 32:63 Error Count Mask - PFVMECM1 (0x00008794)

Field	Bit(s)	Init.	Type	Description
FILTER	31:0	0x0	RW	**Filter** Defines if a packet dropped from pools 32 to 63, respectively, is counted in the SSVPC counter.

8.2.2.22.23 PF VM L2Control Register - PFVML2FLT[n] (0x0000F000 + 0x4*n, n=0...63)

This register controls per VM Inexact L2 Filtering.

Field	Bit(s)	Init.	Type	Description
RESERVED	21:0	0x0	RSV	Reserved.
UPE	22	0b	RW	**Unicast Promiscuous Enable**
VPE	23	0b	RW	**VLAN Promiscuous Enable**
AUPE	24	0b	RW	**Accept Untagged Packets Enable** When set, packets without a VLAN tag can be forwarded to this queue, assuming they pass the Ethernet MAC Address queuing mechanism.
ROMPE	25	0b	RW	**Receive Overflow Multicast Packets** Accept packets that match the MTA table.
ROPE	26	0b	RW	**Receive MAC Filters Overflow** Accept packets that match the PFUTA table.
BAM	27	0b	RW	**Broadcast Accept Mode**
MPE	28	0b	RW	**Multicast Promiscuous Enable**
RESERVED	31:29	000b	RSV	Reserved.

8.2.2.22.24 PF VM VLAN Pool Filter - PFVLVF[n] (0x0000F100 + 0x4*n, n=0...63)

Note: Software should initialize these registers before transmit and receive are enabled.

Field	Bit(s)	Init.	Type	Description
VLAN_ID	11:0	X	RW	**VLAN ID** Defines a VLAN tag for pool VLAN filter n. The bitmap defines which pools belong to this VLAN. ***Note:*** Appears in Little Endian order (LS byte last on the wire).
RESERVED	30:12	X	RSV	Reserved.
VI_EN	31	0b	RW	**VLAN ID Enable** This filter is valid.

8.2.2.22.25 PF VM VLAN Pool Filter Bitmap - PFVLVFB[n] (0x0000F200 + 0x4*n, n=0...127)

Note: Software should initialize these registers before transmit and receive are enabled.

Field	Bit(s)	Init.	Type	Description
POOL_ENA	31:0	X	RW	**Pool Enable bit array** Each couple of registers '2*n' and '2*n+1' enables routing of packets that match a PFVLVF[n] filter to a Pool list (see Section 8.2.2.22.24). Each bit when set, enables packet reception with the associated Pools as follows: • Bit 'i' in register '2*n' is associated with POOL 'i'. • Bit 'i' in register '2*n+1' is associated with POOL '32+i'.

8.2.2.22.26 PF Unicast Table Array - PFUTA[n] (0x0000F400 + 0x4*n, n=0...127)

There is one register per 32 bits of the Unicast Address Table for a total of 128 registers (the PFUTA[127:0] designation). Software must mask to the desired bit on reads, and supply a 32-bit word on writes. The first bit of the address used to access the table is set according to the MCSTCTRL.*MO* field (Section 8.2.2.8.7). The 7 MS bits of the Ethernet MAC Address (out of the 12 bits) selects the register index, while the 5 LS bits (out of the 12 bits) selects the bit within a register.

Note: All accesses to this table must be 32 bit. The lookup algorithm is the same one used for the MTA table. This table should be zeroed by software before start of work.

Field	Bit(s)	Init.	Type	Description
BIT_VECTOR	31:0	X	RW	**Bit Vector** Word wide bit vector specifying 32 bits in the unicast destination address filter table.

8.2.2.22.27 PF Mirror Rule Control - PFMRCTL[n] (0x0000F600 + 0x4*n, n=0...3)

This register defines mirroring rules for each of four destination pools.

Field	Bit(s)	Init.	Type	Description
VPME	0	0b	RW	**Virtual Pool Mirroring Enable** Enables mirroring of certain pools as defined in the PFMRVM registers (Section 8.2.2.22.29).
UPME	1	0b	RW	**Uplink Port Mirroring Enable** Enables mirroring of all traffic received from the network.
DPME	2	0b	RW	**Downlink Port Mirroring Enable** Enables mirroring of all traffic transmitted to the network.
VLME	3	0b	RW	**VLAN Mirroring Enable** Enables mirroring of a set of given VLANs as defined in the PFMRVLAN registers (Section 8.2.2.22.28).
RESERVED	7:4	0x0	RSV	Reserved.
MP	13:8	0x0	RW	**Mirror Pool** Defines the destination pool for this mirror rule.
RESERVED	31:14	0x0	RSV	Reserved.

8.2.2.22.28 PF Mirror Rule VLAN - PFMRVLAN[n] (0x0000F610 + 0x4*n, n=0...7)

This register defines the VLAN values as listed in the PFVLVF table (Section 8.2.2.22.24) taking part in the VLAN mirror rule. Registers 0 and 4 correspond to rule 0, registers 1 and 5 correspond to rule 1, etc. Registers 0..3 correspond to the LSB in the PFVLVF table (e.g. register 0 corresponds to VLAN filters 31:0, while register 4 corresponds to VLAN filters 63:32).

Field	Bit(s)	Init.	Type	Description
VLAN	31:0	0x0	RW	**VLAN** Bitmap listing which VLANs participate in the mirror rule.

8.2.2.22.29 PF Mirror Rule Pool - PFMRVM[n] (0x0000F630 + 0x4*n, n=0...7)

This register defines which pools are being mirrored to the destination pool. Registers 0 and 4 correspond to rule 0, registers 1 and 5 correspond to rule 1, etc. Registers 0..3 correspond to the LSB in the pool list (e.g. register 0 corresponds to pools 31:0, while register 4 corresponds to pools 63:32).

Field	Bit(s)	Init.	Type	Description
POOL	31:0	0x0	RW	**Pool** Bitmap listing which pools participate in the mirror rule.

8.2.2.22.30 PF VM Tag Insert Register - PFVMTIR[n] (0x00017000 + 0x4*n, n=0...63)

Field	Bit(s)	Init.	Type	Description
PORT_TAG_ID	31:0	0x0	RW	**Port Tag ID** Port tag to insert if PFVMVIR.*TAGA* field = 01b. EtagEtype-PFVMTIR[31:0]-0x0000

8.2.2.23 PF - Power Management Registers

8.2.2.23.1 DMA Coalescing Control Register - DMACR (0x00002400)

Field	Bit(s)	Init.	Type	Description
DMACWT	15:0	0x20	RW	**DMA Coalescing Watchdog Timer** When in DMA coalescing, defines the upper limit in 40.960 µs units between arrival of coalescing exit event condition and actual exit from DMA coalescing. The programmed value should be non zero.
HIGH_PRI_TC	23:16	0x0	RW	**High Priority Traffic Class** This field defines which TC is considered high priority, and a packet received for this TC causes exit from DMA Coalescing. When a TC bit is set, it indicates this TC is high priority. Bit 16 TC0 Bit 17 TC1 ... Bit 23 TC7
RESERVED	27:24	0x0	RSV	Reserved.
EN_MNG_IND	28	1b	RW	**Enable Management Indications for DMAC operation** When set, DMA Coalescing functionality is affected from the MCTP management buffer status indications.
RESERVED	29	0b	RSV	Reserved.
LX_COALESCING_INDICATION	30	0b	RW	**Lx Coalescing Indication** Defines whether to move in/out of DMA coalescing when the PCIe link moves in/out of L1/L0s. 0b = DMA coalescing can also start in L0. DMA coalescing stops when any TLP transactions are executed on the PCIe. 1b = DMA coalescing conditions are met only when PCIe is in L1/ L0s.
DMAC_EN	31	0b	RW	**DMA Coalescing Enable** 0b = Disable DMA Coalescing. 1b = Enable DMA Coalescing.

8.2.2.23.2 DMA Coalescing Time to Lx Request - DMCTLX (0x00002404)

Field	Bit(s)	Init.	Type	Description
TTLX	11:0	0x20	RW	**Time to LX Request** Controls the time between detection of DMA Coalescing condition to actual entry into DMA Coalescing state. Timer counts is in 1.28 µs intervals. • For link speed of 10 GbE, minimum value should be 0x1. • For link speed of 1 GbE, minimum value should be 0x2. • For link speed of 100 Mb/s minimum value should be 0x20. This limitation is enforced by the hardware. ***Note:*** Timer adds delay to decision on when to enter DMA Coalescing state.
RESERVED	31:12	0x0	RSV	Reserved. Write 0. Ignore on read.

8.2.2.23.3 DMA Coalescing Threshold - DMCTH[n] (0x00003300 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Type	Description
DMACRXT	8:0	0x20	RW	**DMA Coalescing Receive Threshold Rx PB TC[n]** This value defines the DMA coalescing Receive threshold in 1 KB units. When amount of data in internal receive buffer exceeds *DMACRXT*[n] value, DMA coalescing is stopped. Value written to the field should take into account: • Latency tolerance requirements (LTR) sent over the PCIe and XOFF receive threshold values, to avoid needless generation of flow control packets when in DMA coalescing operating mode and flow control is enabled. • The maximum size of the receive buffer.
RESERVED	31:9	0x0	RSV	Reserved.

8.2.2.23.4 EEE Tx LPI Count - TLPIC (0x000041F4)

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	RC	**EEE Transmit LPI Count** Number of EEE Tx LPI events. Register cleared on read. Register does not wrap around at 0xFFFFFFFF.

8.2.2.23.5 EEE Rx LPI Count - RLPIC (0x000041F8)

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	RC	**EEE Receive LPI Count** Number of EEE Rx LPI events. Register cleared on read. Register does not wrap around at 0xFFFFFFFF.

8.2.2.23.6 Energy Efficient Ethernet (EEE) Setup Register - EEE_SU (0x00004380)

Field	Bit(s)	Init.	Type	Description
DTW_MIN	7:0	0x0	RW	**DTW Minimum** Time to add to minimum Tw_sys_tx value defined in IEEE802.3az clause 78.5. This additional time should be configured according to the link speed/type and is in addition to the specification definition: • For 1000BASE-T operation (16.5 µs) • For 100BASE-TX operation (30 µs) • For 10GBASE-T operation (7.36 µs for Case-1 and, 4.48 µs for Case-2) The minimum Tw_sys_tx value defines the duration where no data is transmitted following move out of EEE LPI Tx state. Time to add defined in this field is expressed in 0.1 µs resolution. ***Note:*** The idle time value defined by this field plus the Tw_sys_tx value defined in IEEE802.3az clause 78.5 is used when moving out of EEE Tx LPI state to transmit flow control frames even if value specified in EEER.*TW_SYSTEM* field is higher (see Section 8.2.2.23.8).
RESERVED	15:8	0x0	RSV	Reserved
TW_WAKE_MIN	21:16	0x0	RW	**TW Wake Minimum** Minimum time, expressed in 1 µs, 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	RW	Reserved
TX_LU_LPI_DLY	25:24	11b	RW	**Transmit Link-Up LPI Delay** Delay to enable entry of Tx EEE LPI state following Link-up indication. 00b = No delay 01b = 10 ms 10b = 100 ms 11b = 1 Second ***Note:*** IEEE802.3az clause 78.1.2.1 defines delay of 1 second following link-up.
TEEE_DLY	31:26	0x2	RW	**Tx EEE LPI Entry Delay** Defines the delay to EEE entry once conditions to enter EEE LPI are detected. Field resolution is 1 µs. ***Note:*** If conditions to enter LPI during the *TEEE_DLY* interval cease to exist, the device does not enter Tx LPI and continues normal operation.

8.2.2.23.7 Energy Efficient Ethernet (EEE) STATUS - EEE_STAT (0x00004398)

Field	Bit(s)	Init.	Type	Description
RESERVED	28:0	0x0	RSV	Reserved/ Write 0. 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 Status** 0b = Rx in Active state 1b = Rx in LPI state
TX_LPI_STATUS	31	0b	RO	**Tx Link in LPI Status** 0b = Tx in Active state 1b = Tx in LPI state

8.2.2.23.8 Energy Efficient Ethernet (EEE) Register - EEER (0x000043A0)

Field	Bit(s)	Init.	Type	Description
TW_SYSTEM	15:0	0x0	RW	**TW System** Time expressed in μs 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, the 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, the value of this field returns to default value, until software re-negotiates new tw_sys_tx value via EEE LLDP. 3. Fast Retrain and Local/Remote Fault indication are not considered link disconnect, and do not cause the field to return to the default value. 4. When transmitting flow control frames, the device waits the minimum time defined in the IEEE802.3az standard before transmitting the flow control packet. the device does not wait the Tw_system time following exit of LPI before transmitting the flow control frame.
TX_LPI_EN	16	0b	RW	**Transmit LPI Enable** Enable entry into EEE LPI on Tx path. 0b = Disable entry into EEE LPI on Tx path. 1b = Enable entry into EEE LPI on Tx path. ***Notes:*** 1. Even when *TX_LPI_EN* is 1b, the device does not enable entry into Tx LPI state for at least 1 second following the change of link_status to OK as defined in IEEE802.3az clause 78.1.2.1. 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.
RX_LPI_EN	17	1b	RW	**Receive LPI Enable** Enable entry into EEE LPI on Rx path 0b = Disable entry into EEE LPI on Rx path. 1b = Enable entry into EEE LPI on Rx path. ***Note:*** Even if the *RX_LPI_EN* bit is set, the device recognizes entry into Rx EEE LPI link state only if EEE support at the link speed was negotiated during auto-negotiation.
RESERVED	27:18	0x0	RSV	Reserved. Write 0. Ignore on read.
EEE_FRC_AN	28	0b	RW	**Force EEE Auto-Negotiation** When bit is set to 1b, enables EEE operation in internal MAC logic even if link partner does not support EEE. ***Note:*** Should be set to 1b to enable testing of EEE operation via MAC loopback.
RESERVED	31:29	000b	RSV	Reserved. Write 0. Ignore on read.

8.2.2.23.9 Latency Tolerance Reporting (LTR) Control - LTRC (0x00011708)

Field	Bit(s)	Init.	Type	Description
SLTRV	9:0	0x5	RW	**Snoop Latency Value** Along with the *Max Snoop Latency Scale* field (*SSCALE*), this register specifies the maximum snoop latency this function wants to request. Software should set this only if it is set to a lower number than the platform's maximum supported latency.
SSCALE	12:10	010b	RW	**Snoop Latency Scale** This field provides a scale for the value contained within the *Snoop Latency Value* field (*SLTRV*). 000b = Value times 1 ns. 001b = Value times 32 ns. 010b = Value times 1,024 ns. 011b = Value times 32,768 ns. 100b = Value times 1,048,576 ns. 101b = Value times 33,554,432 ns. All other values are not permitted It is advised that the software driver use the same scale as configured in the LTR capability.
RESERVED	14:13	00b	RSV	Reserved. Write 0. Ignore on read.
LTRS_REQUIREMENT	15	0b	RW	**LTR Snoop Requirement** 0b = No Latency requirements in Snoop memory access. 1b = Latency tolerance in Snoop memory access specified in the *SLTRV* field.
NSLTRV	25:16	0x5	RW	**No Snoop Latency Value** Along with the *Max No Snoop Latency Scale* field (*NSSCALE*), this field specifies the maximum no snoop latency this function wants to request. Software should set this only if it is set to a lower number than the platform's maximum supported latency.
NSSCALE	28:26	010b	RW	**No Snoop Latency Scale** This field provides a scale for the value contained within the *No Snoop Latency Value* field. 000b = Value times 1 ns. 001b = Value times 32 ns. 010b = Value times 1,024 ns. 011b = Value times 32,768 ns. 100b = Value times 1,048,576 ns. 101b = Value times 33,554,432 ns. All other values are reserved. It is advised that the software driver use the same scale as configured in the LTR capability.
RESERVED	29	0b	RSV	Reserved. Write 0. Ignore on read.
LTR_SEND	30	0b	RW	**LTR Send** Indication to send an LTR message. This bit is set by the software (indicates to hardware to send an LTR message update to the system), and cleared by hardware after the message is sent. This bit is not sent as part of the LTR message.
LTRNS_REQUIREMENT	31	0b	RW	**LTR No Snoop Requirement** 0b = No Latency requirements in No Snoop memory access. 1b = Latency tolerance in No Snoop memory access specified in the *NSLTRV* field.

8.2.2.23.10 DMA Coalescing Management Threshold - DMCMNGTH (0x00015F20)

Field	Bit(s)	Init.	Type	Description
RESERVED	3:0	0x0	RSV	Reserved.
DMCMNGTHR	19:4	0x100	RW	**DMA Coalescing Management Threshold** BMC Tx DMAC Threshold. This value defines the DMA coalescing management threshold in 16 byte units. When amount of empty space in internal transmit buffer exceeds *DMCMNGTHR* value, DMA coalescing is stopped and PCIe moves to L0 state. ***Note:*** If value is 0x0 condition to move out of DMA Coalescing due to passing the DMA Coalescing Management Threshold level is disabled.
RESERVED	31:20	0x0	RSV	Reserved.

8.2.2.24 PF - Security Registers

Security registers are mainly concerned with the internal settings of the AES crypto engine used by IPsec. They are owned by the PF in an IOV mode.

8.2.2.24.1 Security Tx Control - SECTXCTRL (0x00008800)

Field	Bit(s)	Init.	Type	Description
SECTX_DIS	0	1b	RW	**Security Transmit Disable** Tx security offload disable bit. 0b = The AES crypto engine used in Tx by LinkSec or IPsec offload is enabled. Normal operating mode when a security off load is enabled. 1b = The AES crypto engine used in Tx by LinkSec and IPsec offloads is disabled. This mode must be used to save the X550's power consumption when no security offload is enabled. This bit is RW/RO if fused-off.
TX_DIS	1	0b	RW	**Transmit Disable** Disable security Tx path. 0b = Tx data path is enabled. Normal operating mode. 1b = No new packet is fetched out from the Tx Packet Buffers, so that the Tx Security block can be internally emptied prior to changing the security mode. The SECTXSTAT.*SECTX_RDY* bit is de-asserted until the path is emptied by hardware (see Section 8.2.2.24.2).
STORE_FORWARD	2	0b	RW	**Store and Forward** Tx security buffer mode. 0b = Tx sec buffer is operated in pass-through mode. Operating mode when LinkSec is enabled or when no security off load is enabled. 1b = A complete frame is stored in the internal security Tx buffer prior to being forwarded to the MAC. Operating mode when IPsec off load is enabled (as requested to overwrite *ICV* field in AH frames). Note that it increases the Tx internal latencies (for all TCs).
RESERVED	31:3	0x0	RSV	Reserved.

8.2.2.24.2 Security Tx Status - SECTXSTAT (0x00008804)

Field	Bit(s)	Init.	Type	Description
SECTX_RDY	0	0b	RO	**Security Transmit Ready** Tx security block ready for mode change. 0b = Indicates that the internal data path from the Tx Packet Buffers to the Tx security block is not empty, and thus software cannot change the security mode. 1b = Indicates that the internal data path from the Tx Packet Buffers to the Tx security block has been emptied, and thus the security mode can be changed by software. This bit is polled by software once the SECTXCTRL.*TX_DIS* bit is set (Section 8.2.2.24.1).
SECTX_OFF_DIS	1	0b	RO	**Security Transmit Off Disable** Tx security offload disabled. When set, indicates that the Tx security offload feature is disabled by fuse or strapping pin.
ECC_TXERR	2	0b	RO	**ECC Transmit Error** Unrecoverable ECC error occurred in the Tx SA Table or SEC Tx FIFO. 0b = No ECC error occurred on the Tx SA Table since the last time device was reset. 1b = Indicates that an unrecoverable ECC error occurred when accessing internally the Tx SA Table. The ECC interrupt is set as well, until the device is reset by software.

Field	Bit(s)	Init.	Type	Description
RESERVED	31:3	0x0	RSV	Reserved.

8.2.2.24.3 Security Tx Buffer Almost Full - SECTXBUFFAF (0x00008808)

Field	Bit(s)	Init.	Type	Description
FULLTHRESH	9:0	0x250	RW	**Full Threshold** Tx security buffer almost full threshold (relatively to full capacity). The size of the security buffer is 0x274 lines of 16 bytes. In LinkSec offload, the buffer operates in path-through mode, and the recommended threshold is 0x250. It means that the almost full indication is generated very soon while only a fraction of a packet is stored in the buffer. In IPsec mode the buffer operates in a store-and-forward mode, and the recommended threshold is 0x15. It means that the almost full indication is generated only after the buffer contains at least a whole jumbo packet.
RESERVED	31:10	0x0	RSV	Reserved.

8.2.2.24.4 Security Tx Buffer Minimum IFG - SECTXMINIFG (0x00008810)

Field	Bit(s)	Init.	Type	Description
MINSECIFG	3:0	0x1	RW	**Minimum Security IFG** Minimum IFG between packets. It is the minimum gap between consecutive frames from the DBU-Tx required for the security block. The *MINSECIFG* is measured in Wake DMA clock units (equal to 6.4 ns in 10 GbE).
RESERVED	7:4	0x0	RSV	Reserved.
MRKRINSERT	18:8	0x10	RW	This field is used to configure the Security Tx Buffer. When in DCB mode, it should be modified as follows: • PFC disabled — Set to 0x1F. • PFC enabled and 9.5 KB Jumbo supported — Set to 0x640 (it represents a PFC XOFF recovery delay of ~10 µs in 10 GbE). • PFC enabled and 9.5 KB Jumbo not supported — Set to 0x1A0 (it represents a PFC XOFF recovery delay of ~2.6 µs in 10 GbE).
RESERVED	31:19	0x0	RSV	Reserved.

8.2.2.24.5 Security Rx Control - SECRXCTRL (0x00008D00)

Field	Bit(s)	Init.	Type	Description
SECRX_DIS	0	1b	RW	**Security Receive Disable** Rx security offload disable bit. 0b = The AES crypto engine used in Rx by LinkSec or IPsec offload is enabled. Normal operating mode when a security off load is enabled. 1b = The AES crypto engine used in Rx by LinkSec and IPsec offloads is disabled. This mode must be used to save the X550's power consumption when no security off load is enabled.
RX_DIS	1	0b	RW	**Transmit Disable** Disable security Rx path. 0b = Rx data path is enabled. Normal operating mode. 1b = Any new packet received from the Rx MAC is filtered out, so that the Rx Security block can be internally emptied prior to changing the security mode. The SECRXSTAT.*SECRX_RDY* bit is de-asserted until the path is emptied by hardware (see Section 8.2.2.24.6). This bit is RW/RO if fused-off.
RESERVED	31:2	0x0	RSV	Reserved.

8.2.2.24.6 Security Rx Status - SECRXSTAT (0x00008D04)

Field	Bit(s)	Init.	Type	Description
SECRX_RDY	0	0b	RO	**Security Receive Ready** Rx security block ready for mode change. 0b = Indicates that the internal data path from the Rx MAC to the Rx security block is not empty, and thus software cannot change the security mode. 1b = Indicates that the internal data path from the Rx MAC to the Rx security block has been emptied, and thus the security mode can be changed by software. This bit is polled by software once the SECRXCTRL.*RX_DIS* bit is set (Section 8.2.2.24.5).
SECRX_OFF_DIS	1	0b	RO	**Security Receive Off Disable** Rx security offload disabled. When set, it indicates that the Rx security offload feature is disabled by the internal fuse or by strapping pin.
ECC_RXERR	2	0b	RO	**ECC Receive Error** Unrecoverable ECC error occurred in an Rx SA Table. 0b = No ECC error occurred on the Rx SA Table since the last time device was reset. 1b = Indicates that an unrecoverable ECC error occurred when accessing internally one Rx SA Table. The ECC interrupt is set as well, until the device is reset by software.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.2.25 PF - IPsec Registers

IPsec registers are owned by the PF in an IOV mode. There is no added value here to encrypt the SA contents when being read by software, because the SA contents is available in clear text from system memory, like for any IPsec flow handled in software.

8.2.2.25.1 IPsec Tx Index - IPSTXIDX (0x00008900)

Notes: *WRITE* and *READ* bits must not be set at the same time by software.

IPS_TX_EN is RW, but it is RO if fused-off and/or if SECTXCTRL.*SECTX_DIS* is set to 1b.

Field	Bit(s)	Init.	Type	Description
IPS_TX_EN	0	0b	RW	**IPsec Tx Enable** IPsec Tx off-load enable bit. 0b = IPsec off-load ability is disabled for Tx path, regardless of the contents of the Tx SA table. 1b = IPsec of-load ability is enabled for Tx path.
RESERVED	2:1	00b	RSV	Reserved.
SA_IDX	12:3	0x0	RW	**SA Index** SA index for indirect access into the Tx SA table.
RESERVED	29:13	0x0	RSV	Reserved.
READ	30	0b	RW	**READ command** When set, the contents of the Tx SA table entry pointed by the *SA_IDX* field is loaded into the IPSTXKEY 0...3 and IPSTXSALT registers. Immediately self-cleared by hardware once the entry contents has been loaded into the registers.
WRITE	31	0b	RW	**WRITE command** When set, the contents of the IPSTXKEY 0...3 and IPSTXSALT registers are loaded into the Tx SA table entry pointed to by the *SA_IDX* field. Immediately self-cleared by hardware once the entry contents have been loaded into the memory.

8.2.2.25.2 IPsec Tx Salt Register - IPSTXSALT (0x00008904)

Field	Bit(s)	Init.	Type	Description
AES_128_SALT	31:0	0x0	RW	**AEA 128 Salt** 4 bytes salt that has been read/written from/to the Tx SA entry pointed to by *SA_IDX*.

8.2.2.25.3 IPsec Tx Key Registers - IPSTXKEY[n] (0x00008908 + 0x4*n, n=0...3)

Field	Bit(s)	Init.	Type	Description
AES_128_KEY	31:0	0x0	RW	**AES 128 Key** 4 bytes of a 16-byte key that has been read/written from/to the Tx SA entry pointed to by *SA_IDX*. n=0 contains the LSB of the key. n=3 contains the MSB of the key.

8.2.2.25.4 IPsec Rx Index - IPSRXIDX (0x00008E00)

Notes: *WRITE* and *READ* bits must not be set at the same time by software.

IPS_RX_EN is RW, but it is RO if fused-off and/or if SECRXCTRL.*SECRX_DIS* is set to 1b.

Software is not allowed to write/read access registers that belong to different Rx SA tables without writing the IPSRXIDX register in between for setting the *WRITE*/*READ* bit. Refer to Rx SA tables access rules described in Section 7.12.9.

Software should not make changes in the Rx SA tables while changing the *IPS_RX_EN* bit.

Field	Bit(s)	Init.	Type	Description
IPS_RX_EN	0	0b	RW	**IPsec Rx Enable** IPsec Rx off load enable bit. 0b = IPsec off load ability is disabled for Rx path, regardless of the contents of Rx SA tables. 1b = IPsec off load ability is enabled for Rx path.
TABLE	2:1	00b	RW	**TABLE select bits** 00b = No Rx SA table is accessed. 01b = IP Address table is accessed. 10b = SPI table is accessed. 11b = KEY table is accessed.
TB_IDX	12:3	0x0	RW	**Table Index** Table index bits for indirect access into the Rx SA table selected by *TABLE* bits. When accessing the IP Address table, only the 7 least significant bits of this field are meaningful.
RESERVED	29:13	0x0	RSV	Reserved.
READ	30	0b	RW	**READ command** When set, the contents of the Rx SA table entry as pointed to by the [*TABLE*, *TB_IDX*] fields is loaded into the corresponding registers. Immediately self-cleared by hardware once the entry contents have been loaded into the corresponding registers. For instance, if this bit is set together with *TABLE*=10b and *TB_IDX*=0x9, the SPI value stored in entry 9 is loaded into the IPSRXSPI 0...3 registers. Rx SA registers related to another Rx SA table (e.g. IPSRXKEY 0...3 registers) must not be read when TABLE=01b.
WRITE	31	0b	RW	**WRITE command** When set, the contents of the registers concerned by the Rx SA table pointed to by the *TABLE* field is loaded into the table entry pointed to by the *TB_IDX* field. Immediately self-cleared by hardware once the entry contents have been loaded into the memory. For instance, if this bit is set together with *TABLE*=10b and *TB_IDX*=0x9, the value written in IPSRXSPI 0...3 registers is loaded into the SPI table entry 9.

8.2.2.25.5 IPsec Rx IP Address Register - IPSRXIPADDR[n] (0x00008E04 + 0x4*n, n=0...3)

These registers are related to the IP Address table.

Field	Bit(s)	Init.	Type	Description
IPADDR	31:0	0x0	RW	**IP Address** 4 bytes of 16-byte destination IP Address for the associated Rx SA(s). n=0 contains the MSB for an IPv6 IP Address. n=3 contains an IPv4 IP Address or the LSB for an IPv6 IP Address. For an IPv4 address, IPSRXIPADDR 0...2 must be written with zeros. ***Note:*** This field is defined in Big Endian (LS byte is first on the wire).

8.2.2.25.6 IPsec Rx SPI Register - IPSRXSPI (0x00008E14)

This register is related to the Rx SPI table.

Field	Bit(s)	Init.	Type	Description
SPI	31:0	0x0	RW	**SPI** *SPI* field for the SPI entry. ***Note:*** This field is defined in Big Endian (LS byte is first on the wire).

8.2.2.25.7 IPsec Rx SPI Register IP Index - IPSRXIPIDX (0x00008E18)

This register is related to the Rx SPI table.

Field	Bit(s)	Init.	Type	Description
IP_IDX	6:0	0x0	RW	**IP Index** Index in the IP Address table where the destination IP Address associated to that SPI entry is found.
RESERVED	31:7	0x0	RSV	Reserved.

8.2.2.25.8 IPsec Rx Key Register - IPSRXKEY[n] (0x00008E1C + 0x4*n, n=0...3)

These registers are related to the Rx KEY table.

Field	Bit(s)	Init.	Type	Description
AES_128_KEY	31:0	0x0	RW	**AES 128 Key** 4 bytes of 16-byte key of the KEY entry. n=0 contains the LSB of the key. n=3 contains the MSB of the key.

8.2.2.25.9 IPsec Rx Salt Register - IPSRXSALT (0x00008E2C)

This register is related to the Rx KEY table.

Field	Bit(s)	Init.	Type	Description
AES_128_SALT	31:0	0x0	RW	**AES 128 Salt** 4 bytes salt associated to the KEY entry.

8.2.2.25.10 IPsec Rx Mode Register - IPSRXMOD (0x00008E30)

This register is related to the Rx KEY table.

Field	Bit(s)	Init.	Type	Description
VALID	0	0b	RW	**Valid bit** 0b = The KEY entry is not valid. 1b = The KEY entry is valid.
RESERVED	1	0b	RSV	Reserved.
PROTO	2	0b	RW	**IPsec Protocol select** 0b = The KEY entry off-loads AH packets. 1b = The KEY entry off-loads ESP packets.
DECRYPT	3	0b	RW	**Decryption bit** When set, hardware performs decryption off load for this KEY entry. Meaningful only if PROTO bit is set (i.e. ESP mode).
IPV6	4	0b	RW	**IPv6 type** When set, only matched IPv6 packet are off-loaded for that KEY entry. When cleared, only matched IPv4 packets are off-loaded for that KEY entry.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.2.26 VF Registers Mapping in the PF Space

8.2.2.26.1 VF Control Register - VFCTRL[n] (0x00000300 + 0x4*n, n=0...63; WO)

This register array is the mapping of the VFs VFCTRL registers.

Field definitions are the same as those defined in Section 8.3.2.1.1.

8.2.2.26.2 VF Extended Interrupt Cause - VFEICR[n] (0x00000B00 + 0x4*n, n=0...63; RC/W1C)

Field definitions are the same as those defined in Section 8.3.2.2.1.

8.2.2.26.3 VF Extended Interrupt Cause Set - VFEICS[n] (0x00000C00 + 0x4*n, n=0...63; WO)

Field definitions are the same as those defined in Section 8.3.2.2.2.

8.2.2.26.4 VF Extended Interrupt Mask Set/Read - VFEIMS[n] (0x00000D00 + 0x4*n, n=0...63; RWS)

Field definitions are the same as those defined in Section 8.3.2.2.3.

8.2.2.26.5 VF Extended Interrupt Mask Clear - VFEIMC[n] (0x00000E00 + 0x4*n, n=0...63; WO)

Field definitions are the same as those defined in Section 8.3.2.2.4.

8.2.2.26.6 VF Good Packets Received Count - VFGPRC[n] (0x0000101C + 0x40*n, n=0...63; RW)

PF mirror of the VFGPRC register.

Field definitions are the same as those defined in Section 8.3.2.6.1.

8.2.2.26.7 VF Good Octets Received Count Low - VFGORC_LSB[n] (0x00001020 + 0x40*n, n=0...63; RW)

PF mirror of the VFGORC_LSB register.

Field definitions are the same as those defined in Section 8.3.2.6.2.

8.2.2.26.8 VF Multiple Receive Queues Command Register - VFMRQC[n] (0x00003400 + 0x4*n, n=0...63; RW)

PF mirror of VFMRQC registers of the VFs.

Field definitions are the same as those defined in Section 8.3.2.3.10.

8.2.2.26.9 VF Mailbox - VFMAILBOX[n] (0x00004C00 + 0x4*n, n=0...63; RW)

This register array is the mapping of the VFMAILBOX registers of the VFs.

Field definitions are the same as those defined in Section 8.3.2.1.5.

8.2.2.26.10 VF Extended Interrupt Auto Mask Enable - VFEIAM[n] (0x00004D00 + 0x4*n, n=0...63; RW)

Field definitions are the same as those defined in Section 8.3.2.2.5.

8.2.2.26.11 VF Interrupt Vector Allocation Registers Misc - VFIVAR_MISC[n] (0x00004E00 + 0x4*n, n=0...63; RW)

These registers map the mailbox interrupt into MSI-X vector (PF mirror). For more details, refer to Section 7.3.4, "Mapping of Interrupt Causes".

Field definitions are the same as those defined in Section 8.3.2.2.7.

8.2.2.26.12 VF Good Packets Transmitted Count - VFGPTC[n] (0x00008300 + 0x4*n, n=0...63; RO)

PF mirror of the VFGPTC register.

Field definitions are the same as those defined in Section 8.3.2.6.5.

8.2.2.26.13 VF Good Octets Transmitted Count LSB - VFGOTC_LSB[n] (0x00008400 + 0x8*n, n=0...63; RO)

PF mirror of the VFGOTC_LSB register.

Field definitions are the same as those defined in Section 8.3.2.6.6.

8.2.2.26.14 VF Good Octets Transmitted Count MSB - VFGOTC_MSB[n] (0x00008404 + 0x8*n, n=0...63; RO)

PF mirror of the VFGOTC_MSB register.

Field definitions are the same as those defined in Section 8.3.2.6.7.

8.2.2.26.15 VF Multicast Packets Received Count - VFMPRC[n] (0x0000D01C + 0x40*n, n=0...63; RO)

PF mirror of the VFMPRC register.

Field definitions are the same as those defined in Section 8.3.2.6.4.

8.2.2.26.16 VF Good Octets Received Count High - VFGORC_MSB[n] (0x0000D020 + 0x40*n, n=0...63; RW)

PF mirror of the VFGORC_MSB register.

Field definitions are the same as those defined in Section 8.3.2.6.3.

8.2.2.26.17 VF Interrupt Vector Allocation Registers - VFIVAR[n] (0x00012500 + 0x4*n, n=0...63; RW)

These registers map VF interrupt causes into MSI-X vectors (PF mirror). For more details, refer to Section 7.3.4, "Mapping of Interrupt Causes".

Field definitions are the same as those defined in Section 8.3.2.2.6.

8.2.2.26.18 PF Mailbox Memory - PFMBMEM[n,m] (0x00013000 + 0x4*n + 0x40*m, n=0...15, m=0...63; RW)

Mailbox Memory for PF and VF Driver Communications. The mailbox size for each VM is 64 bytes accessed by 16 x 32-bit registers. Locations can be accessed as 32- or 64-bit words. This is the mapping of this memory in the PF space.

Field definitions are the same as those defined in Section 8.3.2.1.4.

8.2.3 BAR3 Registers Summary

Table 8-30. PF - MSI-X Table Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00000000 + 0x10*n, n=0...255	MSIXTADD[n]	MSI-X Table Entry Lower Address	8.2.4.1.1
0x00000004 + 0x10*n, n=0...255	MSIXTUADD[n]	MSI-X Table Entry Upper Address	8.2.4.1.2
0x00000008 + 0x10*n, n=0...255	MSIXTMSG[n]	MSI-X Table Entry Message	8.2.4.1.3
0x0000000C + 0x10*n, n=0...255	MSIXVCTRL[n]	MSI-X Table Entry Vector Control	8.2.4.1.4
0x00002000 + 0x4*n, n=0...7	MSIXPBA[n]	MSIXPBA Bit Description	8.2.4.1.5

8.2.4 Detailed Register Descriptions - PF BAR3

8.2.4.1 PF - MSI-X Table Registers

The MSI-X capability is described in Section 9.2.3.3. The MSI-X table is described in Section 9.2.3.3.4 and the Pending Bit Array (PBA) is described in Section 9.2.3.3.5. These registers are located in the MSI-X BAR.

8.2.4.1.1 MSI-X Table Entry Lower Address - MSIXTADD[n] (0x00000000 + 0x10*n, n=0...255)

Field	Bit(s)	Init.	Type	Description
MSIXTADD10	1:0	00b	RW	**Message Address 1:0** For proper DWord alignment, software must always write zeros to these two bits;. Otherwise, the result is undefined. The state of these bits after reset must be 0b. These bits are permitted to be read-only or read/write.
MSIXTADD	31:2	0x0	RW	**Message Address** System-specified message lower address. For MSI-X messages, the contents of this field from an MSI-X table entry specifies the lower portion of the DWord-aligned address (AD[31:02]) for the memory write transaction. This field is read/write.

8.2.4.1.2 MSI-X Table Entry Upper Address - MSIXTUADD[n] (0x00000004 + 0x10*n, n=0...255)

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.

8.2.4.1.3 MSI-X Table Entry Message - MSIXTMSG[n] (0x00000008 + 0x10*n, n=0...255)

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.

8.2.4.1.4 MSI-X Table Entry Vector Control - MSIXVCTRL[n] (0x0000000C + 0x10*n, n=0...255)

Field	Bit(s)	Init.	Type	Description
MSIXVCTRL	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). This bit is read/write.
RESERVED	31:1	0x0	RSV	Reserved. After reset, the state of these bits must be 0b. However, for potential future use, software must preserve the value of these reserved bits when modifying the value of other Vector Control bits. If software modifies the value of these reserved bits, the result is undefined.

8.2.4.1.5 MSIXPBA Bit Description - MSIXPBA[n] (0x00002000 + 0x4*n, n=0...7)

Note: Registers 2...7 are usable only by the VFs in IOV mode. These registers are not exposed to the operating system by the *Table Size* field in the *MSI-X Message Control* word.

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. Bit 'i' in register 'N' is associated with MSI-X vector 32*N+i, where N = 0...3.

8.3 Device Registers - VF

8.3.1 BAR0 Registers Summary

Table 8-31. VF - General Control Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00000000	VFCTRL	VF Control Register	8.3.2.1.1
0x00000008	VFSTATUS	VF Device Status Register	8.3.2.1.2
0x00000010	VFLINKS	Link Status Register	8.3.2.1.3
0x00000200 + 0x4*n, n=0...15	VFMBMEM[n]	VF Mailbox Memory	8.3.2.1.4
0x000002FC	VFMAILBOX	VF Mailbox	8.3.2.1.5

Table 8-32. VF - Interrupt Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00000100	VFEICR	VF Extended Interrupt Cause	8.3.2.2.1
0x00000104	VFEICS	VF Extended Interrupt Cause Set	8.3.2.2.2
0x00000108	VFEIMS	VF Extended Interrupt Mask Set/Read	8.3.2.2.3
0x0000010C	VFEIMC	VF Extended Interrupt Mask Clear	8.3.2.2.4
0x00000114	VFEIAM	VF Extended Interrupt Auto Mask Enable	8.3.2.2.5
0x00000120 + 0x4*n, n=0...3	VFIVAR[n]	VF Interrupt Vector Allocation Registers	8.3.2.2.6
0x00000140	VFIVAR_MISC	VF Interrupt Vector Allocation Registers Misc	8.3.2.2.7
0x00000148	VFPBACL	VF MSI-X PBA Clear	8.3.2.2.8
0x00000180 + 0x4*n, n=0...1	VFRSCINT[n]	VF RSC Enable Interrupt	8.3.2.2.9
0x00000820 + 0x4*n, n=0...1	VFEITR[n]	VF Extended Interrupt Throttle Registers	8.3.2.2.10

Table 8-33. VF - Receive Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00000300	VFPSRTYPE	VF Replication Packet Split Receive Type Register	8.3.2.3.1
0x00001000 + 0x40*n, n=0...7	VFRDBAL[n]	VF Receive Descriptor Base Address Low	8.3.2.3.2
0x00001004 + 0x40*n, n=0...7	VFRDBAH[n]	VF Receive Descriptor Base Address High	8.3.2.3.3
0x00001008 + 0x40*n, n=0...7	VFRDLEN[n]	VF Receive Descriptor Length	8.3.2.3.4
0x00001010 + 0x40*n, n=0...7	VFRDH[n]	VF Receive Descriptor Head	8.3.2.3.5
0x00001014 + 0x40*n, n=0...7	VFSRRCTL[n]	VF Split and Replication Receive Control Registers	8.3.2.3.6
0x00001018 + 0x40*n, n=0...7	VFRDT[n]	VF Receive Descriptor Tail	8.3.2.3.7
0x00001028 + 0x40*n, n=0...7	VFRXDCTL[n]	VF Receive Descriptor Control	8.3.2.3.8
0x0000102C + 0x40*n, n=0...7	VFRSCCTL[n]	VF RSC Control	8.3.2.3.9
0x00003000	VFMRQC	VF Multiple Receive Queues Command Register	8.3.2.3.10

Table 8-33. VF - Receive Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00003100 + 0x4*n, n=0...9	VFRSSRK[n]	VF RSS Random Key Register	8.3.2.3.11
0x00003200 + 0x4*n, n=0...15	VFRETA[n]	VF Redirection Table	8.3.2.3.12

Table 8-34. VF - Transmit Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00002000 + 0x40*n, n=0...7	VFTDBAL[n]	VF Transmit Descriptor Base Address Low	8.3.2.4.1
0x00002004 + 0x40*n, n=0...7	VFTDBAH[n]	VF Transmit Descriptor Base Address High	8.3.2.4.2
0x00002008 + 0x40*n, n=0...7	VFTDLEN[n]	VF Transmit Descriptor Length	8.3.2.4.3
0x00002010 + 0x40*n, n=0...7	VFTDH[n]	VF Transmit Descriptor Head	8.3.2.4.4
0x00002018 + 0x40*n, n=0...7	VFTDT[n]	VF Transmit Descriptor Tail	8.3.2.4.5
0x00002028 + 0x40*n, n=0...7	VFTXDCTL[n]	VF Transmit Descriptor Control	8.3.2.4.6
0x00002038 + 0x40*n, n=0...7	VFTDWBAL[n]	VF Tx Descriptor Completion Write Back Address Low	8.3.2.4.7
0x0000203C + 0x40*n, n=0...7	VFTDWBAH[n]	VF Tx Descriptor Completion Write Back Address High	8.3.2.4.8

Table 8-35. VF - TPH Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x0000100C + 0x40*n, n=0...7	VFTPH_RXCTRL[n]	VF Rx TPH Control Register	8.3.2.5.1
0x0000200C + 0x40*n, n=0...7	VFTPH_TXCTRL[n]	VF Tx TPH Control Registers	8.3.2.5.2

Table 8-36. VF - Statistic Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x0000101C	VFGPRC	VF Good Packets Received Count	8.3.2.6.1
0x00001020	VFGORC_LSB	VF Good Octets Received Count Low	8.3.2.6.2
0x00001024	VFGORC_MSB	VF Good Octets Received Count High	8.3.2.6.3
0x00001034	VFMPRC	VF Multicast Packets Received Count	8.3.2.6.4
0x0000201C	VFGPTC	VF Good Packets Transmitted Count	8.3.2.6.5
0x00002020	VFGOTC_LSB	VF Good Octets Transmitted Count LSB	8.3.2.6.6
0x00002024	VFGOTC_MSB	VF Good Octets Transmitted Count MSB	8.3.2.6.7

8.3.2 Detailed Register Descriptions - VF BAR0

8.3.2.1 VF - General Control Registers

8.3.2.1.1 VF Control Register - VFCTRL (0x00000000)

Field	Bit(s)	Init.	Type	Description
RESERVED	25:0	0x0	RSV	Reserved.
RST	26	0b	WO	**VF Reset** This bit performs a reset of the queue enable and the interrupt registers of the VF.
RESERVED	31:27	0x0	RSV	Reserved.

8.3.2.1.2 VF Device Status Register - VFSTATUS (0x00000008; RO)

This register is a mirror of the PF status register.

Field definitions are the same as those defined in Section 8.2.2.1.2.

8.3.2.1.3 Link Status Register - VFLINKS (0x00000010; RO)

This register is the mapping of the PF's LINKS register.

Field definitions are the same as those defined in Section 8.2.2.16.7.

8.3.2.1.4 VF Mailbox Memory - VFMBMEM[n] (0x00000200 + 0x4*n, n=0...15)

Mailbox Memory for PF and VF driver communications. The mailbox size for each VM is 64 bytes accessed by 16 x 32-bit registers. Locations can be accessed as 32- or 64-bit words.

Field	Bit(s)	Init.	Type	Description
MAILBOX_DATA	31:0	X	RW	**Mailbox Data** Mailbox data is composed of 16 x 4 byte registers.

8.3.2.1.5 VF Mailbox - VFMAILBOX (0x000002FC)

This register is cleared by VLFR (excluding the *RSTI* bit).

Field	Bit(s)	Init.	Type	Description
REQ	0	0b	WO	**Request** Request for PF Ready. Setting this bit, causes an interrupt to the PF. This bit always reads as 0b. Setting this bit sets the corresponding bit in the *VFREQ* field in PFMBICR register (Section 8.2.2.22.2).
ACK	1	0b	WO	**Acknowledge** PF message received. Setting this bit, causes an interrupt to the PF. This bit always reads as 0b. Setting this bit sets the corresponding bit in the *VFACK* field in PFMBICR register.

Field	Bit(s)	Init.	Type	Description
VFU	2	0b	RW	**VFU** Buffer is taken by VF. This bit can be set only if the *PFU* bit is cleared and is mirrored in the *VFU* bit of the PFMAILBOX register (Section 8.2.2.22.8).
PFU	3	0b	RW	**PFU** Buffer is taken by PF. This bit is RO for the VF and is a mirror of the *PFU* bit of the PFMAILBOX register.
PFSTS	4	0b	RC	**PF Status** PF wrote a message in the mailbox.
PFACK	5	0b	RC	**PF Acknowledge** PF acknowledged the VF previous message.
RSTI	6	1b	RO	**Reset Indication** Indicates that the PF reset the shared resources and the reset sequence is in progress. This bit is not affected by VFLR.
RSTD	7	0b	RC	**Reset Done** Indicates that a PF software reset completed.
RESERVED	31:8	0x0	RSV	Reserved.

8.3.2.2 VF - Interrupt Registers

8.3.2.2.1 VF Extended Interrupt Cause - VFEICR (0x00000100)

Field	Bit(s)	Init.	Type	Description
MSIX	2:0	000b	RW1C	**MSI-X** Indicates an interrupt cause mapped to MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.2.2.2 VF Extended Interrupt Cause Set - VFEICS (0x00000104)

Field	Bit(s)	Init.	Type	Description
MSIX	2:0	000b	WO	**MSI-X** Sets to the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.2.2.3 VF Extended Interrupt Mask Set/Read - VFEIMS (0x00000108)

Field	Bit(s)	Init.	Type	Description
MSIX	2:0	000b	RWS	**MSI-X** Sets the Mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.2.2.4 VF Extended Interrupt Mask Clear - VFEIMC (0x0000010C)

Field	Bit(s)	Init.	Type	Description
MSIX	2:0	000b	WO	**MSI-X** Clears the Mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.2.2.5 VF Extended Interrupt Auto Mask Enable - VFEIAM (0x00000114)

Field	Bit(s)	Init.	Type	Description
MSIX	2:0	000b	RW	**MSI-X** Auto-mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.2.2.6 VF Interrupt Vector Allocation Registers - VFIVAR[n] (0x00000120 + 0x4*n, n=0...3)

These registers map VF interrupt causes into MSI-X vectors. For more details, refer to Section 7.3.4, "Mapping of Interrupt Causes".

Field	Bit(s)	Init.	Type	Description
INT_ALLOC_0	0	0b	RW	**Interrupt Allocation 0** Defines the MSI-X vector (0 or 1) assigned to Rx queue '2*N' for IVAR 'N' register (N=0...3).
RESERVED	6:1	0x0	RSV	Reserved.
INT_ALLOC_VAL_0	7	0b	RW	**Interrupt Allocation Valid 0** Valid bit for *INT_ALLOC*[0].
INT_ALLOC_1	8	0b	RW	**Interrupt Allocation 1** Defines the MSI-X vector (0 or 1) assigned to Tx queue '2*N' for IVAR 'N' register (N=0...3).
RESERVED	14:9	0x0	RSV	Reserved.
INT_ALLOC_VAL_1	15	0b	RW	**Interrupt Allocation Valid 1** Valid bit for *INT_ALLOC*[1].
INT_ALLOC_2	16	0b	RW	**Interrupt Allocation 2** Defines the MSI-X vector (0 or 1) assigned to Rx queue '2*N+1' for IVAR 'N' register (N=0...3).
RESERVED	22:17	0x0	RSV	Reserved.
INT_ALLOC_VAL_2	23	0b	RW	**Interrupt Allocation Valid 2** Valid bit for *INT_ALLOC*[2].
INT_ALLOC_3	24	0b	RW	**Interrupt Allocation 3** Defines the MSI-X vector (0 or 1) assigned to Tx queue '2*N+1' for IVAR 'N' register (N=0...3).
RESERVED	30:25	0x0	RSV	Reserved.
INT_ALLOC_VAL_3	31	0b	RW	**Interrupt Allocation Valid 3** Valid bit for *INT_ALLOC*[3].

8.3.2.2.7 VF Interrupt Vector Allocation Registers Misc - VFIVAR_MISC (0x00000140)

This register maps the mailbox interrupt into MSI-X vector. For more details, refer to Section 7.3.4, "Mapping of Interrupt Causes".

Field	Bit(s)	Init.	Type	Description
INT_ALLOC_0	1:0	X	RW	**Interrupt Allocation 0** Defines the MSI-X vector assigned to the mailbox interrupt.
RESERVED	6:2	0x0	RSV	Reserved.
INT_ALLOC_VAL_0	7	0b	RW	**Interrupt Allocation Valid 0** Valid bit for *INT_ALLOC*[0].
RESERVED	31:8	0x0	RSV	Reserved.

8.3.2.2.8 VF MSI-X PBA Clear - VFPBACL (0x00000148)

Field	Bit(s)	Init.	Type	Description
PENBIT	2:0	0x0	RW1C	**MSI-X Pending Bits Clear** MSI-X Pending Bits Clear. Writing a 1b to any bit clears the corresponding *MSIXPBA* bit. Writing a 0b has no effect. Reading this register returns the PBA vector.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.2.2.9 VF RSC Enable Interrupt - VFRSCINT[n] (0x00000180 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 8.2.2.6.20.

8.3.2.2.10 VF Extended Interrupt Throttle Registers - VFEITR[n] (0x00000820 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 8.2.2.6.4.

8.3.2.3 VF - Receive Registers

8.3.2.3.1 VF Replication Packet Split Receive Type Register - VFPSRTYPE (0x00000300; RW)

Field definitions are the same as those defined in Section 8.2.2.8.17.

8.3.2.3.2 VF Receive Descriptor Base Address Low - VFRDBAL[n] (0x00001000 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.9.1.

8.3.2.3.3 VF Receive Descriptor Base Address High - VFRDBAH[n] (0x00001004 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.9.2.

8.3.2.3.4 VF Receive Descriptor Length - VFRDLEN[n] (0x00001008 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.9.3.

8.3.2.3.5 VF Receive Descriptor Head - VFRDH[n] (0x00001010 + 0x40*n, n=0...7; RO)

Field definitions are the same as those defined in Section 8.2.2.9.4.

8.3.2.3.6 VF Split and Replication Receive Control Registers - VFSRRCTL[n] (0x00001014 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.9.5.

8.3.2.3.7 VF Receive Descriptor Tail - VFRDT[n] (0x00001018 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.9.6.

8.3.2.3.8 VF Receive Descriptor Control - VFRXDCTL[n] (0x00001028 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.9.7.

8.3.2.3.9 VF RSC Control - VFRSCCTL[n] (0x0000102C + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.9.8.

8.3.2.3.10 VF Multiple Receive Queues Command Register - VFMRQC (0x00003000)

Field	Bit(s)	Init.	Type	Description
RSSE	0	0b	RW	**RSS Enable** This bit enables the VF RSS mechanism. 0b = VF RSS disabled. 1b = VF RSS enable.
RESERVED	15:1	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
RSS_FIELD_ENABLE	31:16	0x0	RW	**RSS Field Enable** Each bit, when set, enables a specific field selection to be used by the hash function. Several bits can be set at the same time. Bit[16] = Enable TcpIPv4 hash function. Bit[17] = Enable IPv4 hash function. Bit[18] = Reserved. Bit[19] = Reserved. Bit[20] = Enable IPv6 hash function. Bit[21] = Enable TcpIPv6 hash function. Bit[22] = Enable UdpIPV4. Bit[23] = Enable UdpIPV6. Bit[24] = Reserved. Bits[31:25] = Reserved; 0x0. ***Note:*** On Tunnel packets IPv4-IPv6, only the IPv4 header might be used for the RSS filtering.

8.3.2.3.11 VF RSS Random Key Register - VFRSSRK[n] (0x00003100 + 0x4*n, n=0...9; RW)

PF mirror of VFRSSK registers of the VFs.

Field definitions are the same as those defined in Section 8.2.2.8.25.

8.3.2.3.12 VF Redirection Table - VFRETA[n] (0x00003200 + 0x4*n, n=0...15; RW)

VF version of VFRETA registers. The redirection table has 64 entries in 16 registers.

Field definitions are the same as those defined in Section 8.2.2.8.26.

8.3.2.4 VF - Transmit Registers

8.3.2.4.1 VF Transmit Descriptor Base Address Low - VFTDBAL[n] (0x00002000 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.10.5.

8.3.2.4.2 VF Transmit Descriptor Base Address High - VFTDBAH[n] (0x00002004 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.10.6.

8.3.2.4.3 VF Transmit Descriptor Length - VFTDLEN[n] (0x00002008 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.10.7.

8.3.2.4.4 VF Transmit Descriptor Head - VFTDH[n] (0x00002010 + 0x40*n, n=0...7; RO)

Field definitions are the same as those defined in Section 8.2.2.10.8.

8.3.2.4.5 VF Transmit Descriptor Tail - VFTDT[n] (0x00002018 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.10.9.

8.3.2.4.6 VF Transmit Descriptor Control - VFTXDCTL[n] (0x00002028 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.10.10.

8.3.2.4.7 VF Tx Descriptor Completion Write Back Address Low - VFTDWBAL[n] (0x00002038 + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.10.11.

8.3.2.4.8 VF Tx Descriptor Completion Write Back Address High - VFTDWBAH[n] (0x0000203C + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.10.12.

8.3.2.5 VF - TPH Registers

8.3.2.5.1 VF Rx TPH Control Register - VFTPH_RXCTRL[n] (0x0000100C + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.12.1.

8.3.2.5.2 VF Tx TPH Control Registers - VFTPH_TXCTRL[n] (0x0000200C + 0x40*n, n=0...7; RW)

Field definitions are the same as those defined in Section 8.2.2.12.3.

8.3.2.6 VF - Statistics Registers

Registers in this section are RO by VF and RW by PF. Statistics are reset by PF clearing the register.

8.3.2.6.1 VF Good Packets Received Count - VFGPRC (0x0000101C)

Field	Bit(s)	Init.	Type	Description
VFGPRC	31:0	0x0	RW	**VF Good Packets Received Count** Number of good packets received for this VF (of any length). This counter includes loopback packets or replications of multicast packets. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.2.6.2 VF Good Octets Received Count Low - VFGORC_LSB (0x00001020)

Field	Bit(s)	Init.	Type	Description
VFGORC_LSB	31:0	0x0	RW	**VF Good Octets Received Count LSB** Number of good octets received (32 LSB of a 36-bit counter) by the queues allocated to this VF. The counter includes loopback packets or replications of multicast packets. This register includes bytes received in a packet from the *Destination Address* field through the *CRC* field, inclusively. Octets are counted on the VF interface rather than on the network interface. For example, LinkSec octets are not counted. Bytes of RSC are counted before coalescing. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.2.6.3 VF Good Octets Received Count High - VFGORC_MSB (0x00001024)

Field	Bit(s)	Init.	Type	Description
VFGORC_MSB	3:0	0x0	RW	**VF Good Octets Received Count MSB** Number of good octets received (4 MSB of a 36-bit counter) by the queues allocated to this VF (see Section 8.3.2.6.2 for more details). The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0.
RESERVED	31:4	0x0	RSV	Reserved.

8.3.2.6.4 VF Multicast Packets Received Count - VFMPRC (0x00001034)

Field	Bit(s)	Init.	Type	Description
VFMPRC	31:0	0x0	RO	**VF Multicast Packets Received Count** Number of multicast good packets received by this VF (of any length) that pass the Ethernet MAC Address filtering (excluding broadcast packets). The counter does not count received flow control packets. This register increments only if receives are enabled. This register does not count packets counted by the Missed Packet Count (MPC) register. This counter also includes loopback packets or replications of multicast packets. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.2.6.5 VF Good Packets Transmitted Count - VFGPTC (0x0000201C)

Field	Bit(s)	Init.	Type	Description
VFGPTC	31:0	0x0	RO	**VF Good Packets Transmitted Count** Number of good packets sent by the queues allocated to this VF. A packet is considered as transmitted if it is was forwarded to the MAC unit for transmission to the network and/or is accepted by the internal Tx to Rx switch enablement logic. Packets dropped due to anti-spoofing filtering or loopback packets that are rejected by the Tx to Rx switch are not counted. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.2.6.6 VF Good Octets Transmitted Count LSB - VFGOTC_LSB (0x00002020)

Field	Bit(s)	Init.	Type	Description
VFGOTC_LSB	31:0	0x0	RO	**VF Good Octets Transmitted Count LSB** Number of good octets transmitted (32 LSB of a 36-bit counter) by the queues allocated to this VF. This register includes bytes transmitted in a packet from the *Destination Address* field through the *CRC* field, inclusively. This register counts octets of the packets counted by the VFGPTC register. Octets are counted on the VF interface rather than on the network interface. For example, LinkSec octets are not counted. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.2.6.7 VF Good Octets Transmitted Count MSB - VFGOTC_MSB (0x00002024)

Field	Bit(s)	Init.	Type	Description
VFGOTC_MSB	3:0	0x0	RO	**VF Good Octets Transmitted Count MSB** Number of good octets transmitted (4 MSB of a 36-bit counter) by the queues allocated to this VF (see Section 8.3.2.6.6 for more details). The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0.
RESERVED	31:4	0x0	RSV	Reserved.

8.3.3 BAR3 Registers Summary

Table 8-37. VF - MSI-X Table Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Number
0x00000000 + 0x10*n, n=0...2	VFMSIXTADD[n]	MSI-X Table Entry Lower Address	8.3.4.1.1
0x00000004 + 0x10*n, n=0...2	VFMSIXTUADD[n]	MSI-X Table Entry Upper Address	8.3.4.1.2
0x00000008 + 0x10*n, n=0...2	VFMSIXTMSG[n]	MSI-X Table Entry Message	8.3.4.1.3
0x0000000C + 0x10*n, n=0...2	VFMSIXVCTRL[n]	MSI-X Table Entry Vector Control	8.3.4.1.4
0x00002000	VFMSIXPBA	MSIXPBA Bit Description	8.3.4.1.5

8.3.4 Detailed Register Descriptions - VF BAR3

8.3.4.1 VF - MSI-X Table Registers

8.3.4.1.1 MSI-X Table Entry Lower Address - VFMSIXTADD[n] (0x00000000 + 0x10*n, n=0...2)

Field	Bit(s)	Init.	Type	Description
MSIXTADD_10	1:0	00b	RW	**Message Address 1:0** For proper DWord alignment, software must always write zeros to these two bits. Otherwise, the result is undefined. The state of these bits after reset must be 0b. These bits are permitted to be read-only or read/write.
MSIXTADD	31:2	0x0	RW	**Message Address** System-specified message lower address. For MSI-X messages, the contents of this field from an MSI-X table entry specifies the lower portion of the DWord-aligned address (AD[31:02]) for the memory write transaction. This field is read/write.

8.3.4.1.2 MSI-X Table Entry Upper Address - VFMSIXTUADD[n] (0x00000004 + 0x10*n, n=0...2)

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.

8.3.4.1.3 MSI-X Table Entry Message - VFMSIXTMSG[n] (0x00000008 + 0x10*n, n=0...2)

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.

8.3.4.1.4 MSI-X Table Entry Vector Control - VFMSIXVCTRL[n] (0x0000000C + 0x10*n, n=0...2)

Field	Bit(s)	Init.	Type	Description
MSIXVCTRL	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). This bit is read/write.
RESERVED	31:1	0x0	RSV	Reserved. After reset, the state of these bits must be 0b. However, for potential future use, software must preserve the value of these reserved bits when modifying the value of other Vector Control bits. If software modifies the value of these reserved bits, the result is undefined.

8.3.4.1.5 MSIXPBA Bit Description - VFMSIXPBA (0x00002000)

Note: If a page size larger than 8 K is programmed in the IOV structure, the address of the MSIX PBA table moves to be page aligned.

Field	Bit(s)	Init.	Type	Description
PENDING_BITS	2:0	000b	RO	**Pending Bits** For each pending bit that is set, the function has a pending message for the associated MSI-X Table entry. Pending bits that have no associated MSI-X table entry are reserved.
RESERVED	31:3	0x0	RSV	Reserved.

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

Chapter 9 PCIe Programming Interface

9.1 Overview

The X550 is a multi-function device with the following functions:

- LAN 0
- LAN 1

Different parameters affect how LAN functions are exposed on PCIe.

Both functions contain the following regions of the PCI configuration space (some of them are enabled by NVM settings as detailed in the following sections):

- Mandatory PCI configuration registers - Section 9.2.2
- Legacy PCI capabilities - Section 9.2.3
  - Power management capabilities
  - MSI / MSI-X capabilities
  - Vital Product Data (VPD) capability
- PCIe extended capabilities - Section 9.2.4
  - Advanced Error Reporting (AER)
  - Serial ID
  - Alternate requester ID.
  - Single root IOV
  - Latency Tolerance Reporting
  - TPH Requester
  - Access Control Services
  - Secondary PCI Express

9.1.1 Register Attributes

Table 9-1 lists the register attributes used in this section.

Table 9-1. Register Attribute Descriptions

Type	Description
RO	**Read-Only register** Register bits are read-only and cannot be altered by software.
RW	**Read/Write register** Register bits are read-write and can be either set or reset.
RW1C	**Read-only status — Write 1b to Clear status register** Writing a 0b to RW1C bits has no effect.
ROS	**Read-Only register with Sticky bits** Register bits are read-only and cannot be altered by software. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled.
RWS	**Read/Write register with Sticky bits** Register bits are read-write and can be either set or reset by software to the desired state. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled.
RW1CS	**Read-only status — Write 1b to Clear status register** Register bits indicate status when read, a set bit, indicating a status event, can be cleared by writing a 1b to it. Writing a 0b to RW1C bits has no effect. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled.
HwInit	**Hardware Initialized** Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial NVM. Bits are read-only after initialization and can only be reset (for write-once by firmware) with the PWRGOOD signal.
RsvdP	**Reserved and Preserved** Reserved for future read/write implementations; software must preserve value read for writes to these bits.
RsvdZ	**Reserved and Zero** Reserved for future RW1C implementations; software must use 0b for writes to these bits.

9.2 PCIe Register Map

9.2.1 PCIe Configuration Space Summary

Table 9-2 lists the PCIe configuration registers while their detailed description is given in the sections that follow. PCI configuration fields in the summery table are presented by the following marking:

- Fields that have meaningful default values are indicated in parenthesis — (**value**).
- Dotted fields indicates the same value for both LAN functions
- Light-blue fields indicate read-only fields (loaded from the NVM)
- Magenta fields indicate hard-coded values.
- Other fields contain RW attributes.

Table 9-2. PCI Configuration Registers Map - LAN Functions

Section	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
Mandatory PCI Register	0x0	Device ID		Vendor ID	
	0x4	Status Register		Command Register	
	0x8	Class Code (0x020000/0x010000)			Revision ID
	0xC	Reserved	Header Type (0x0/ 0x80)	Latency Timer	Cache Line Size (0x0)
	0x10	Base Address Register 0			
	0x14	Base Address Register 1			
	0x18	Base Address Register 2			
	0x1C	Base Address Register 3			
	0x20	Base Address Register 4			
	0x24	Base Address Register 5			
	0x28	CardBus CIS pointer (0x0000)			
	0x2C	Subsystem ID		Subsystem Vendor ID	
	0x30	Expansion ROM Base Address			
	0x34	Reserved			Cap Ptr (0x40)
	0x38	Reserved			
	0x3C	Max Latency (0x00)	Min Grant (0x00)	Interrupt Pin (0x01...0x04)	Interrupt Line (0x00)
PCI / PCIe Capabilities	0x40...0x47	Power Management Capability			
	0x50...0x67	MSI Capability			
	0x70...0x7B	MSI-X Capability			
	0xA0...0xDB	PCIe Capability			
	0xE0...0xE7	VPD Capability			

Table 9-2. PCI Configuration Registers Map - LAN Functions [continued]

Section	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
Extended PCIe Configuration	0x100...0x12B	AER Capability			
	0x140...0x14B	Serial ID Capability			
	0x150...0x157	ARI Capability			
	0x160...0x19F	SR-IOV Capability			
	0x1A0...0x1AB	TPH Capability			
	0x1B0...0x1B7	ACS Capability			
	0x1C0...0x1C7	LTR Capability			
	0x1D0...0x1E3	Secondary PCI Express Extended Capability			

Table 9-3. PCIe Configuration Registers Map - Dummy Function

Section	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
Mandatory PCI Register	0x0	Device ID		Vendor ID	
	0x4	Status Register		Command Register	
	0x8	Class Code (0xFF0000)			Revision ID
	0xC	Reserved	Header Type (0x0/0x80)	Latency Timer	Cache Line Size (0x0)
	0x10	Base Address Register 0			
	0x14	Base Address Register 1 (0x0)			
	0x18	Base Address Register 2 (0x0)			
	0x1C	Base Address Register 3 (0x0)			
	0x20	Base Address Register 4 (0x0)			
	0x24	Base Address Register 5 (0x0)			
	0x28	CardBus CIS pointer (0x0000)			
	0x2C	Subsystem Device ID		Subsystem Vendor ID	
	0x30	Expansion ROM Base Address (0x0)			
	0x34	Reserved			Cap Ptr (0x40)
	0x38	Reserved			
	0x3C	Max Latency (0x00)	Min Grant (0x00)	Interrupt Pin (0x00)	Interrupt Line (0x00)
PCI / PCIe Capabilities	0x40...0x47	Power Management Capability			
	0xA0...0xDB	PCIe Capability			
	0xE0...0xE7	VPD Capability			
Extended PCIe Configuration	0x100...0x12B	AER Capability			
	0x140...0x14B	Serial ID Capability			
	0x1B0...0x1B7	ACS Capability			

9.2.1.1 Sharing Among PCI Functions

The X550 supports multiple PCI functions. As each function exposes a PCIe configuration space, each register and each field is either shared among the functions or is replicated per each PCI function. In each section a table summarizes configuration sharing of the registers described in this section. Also, the description of each field describes special considerations regarding configuration sharing.

9.2.2 Mandatory PCI Configuration Registers

Table 9-4. Configuration Sharing of PCI Configuration Space

Field	Sub-field	Shared?	Replicated?	Comments
Vendor ID	Vendor ID	x		
Device ID	Device ID		x	
Command Register	I/O Access Enable		x	Issue UR per PF if disabled.
	Memory Access Enable		x	Issue UR per PF if disabled.
	Bus Master Enable		x	
	Parity Error Response		x	Enables certain error reporting per PF.
	SERR# Enable		x	Controls error reporting per PF.
	Interrupt Disable		x	Selection of interrupt method per PF.
Status Register	Interrupt Status		x	
	Capabilities List	x		Hard-wired to 1b.
	Data Parity Reported / Master Data Parity Error		x	Reports poisoned packets per PF.
	Signaled Target Abort		x	Reports Completer Abort per PF.
	Received Target Abort		x	Reports receiving a Completer Abort per PF.
	Received Master Abort		x	Reports receiving an UR per PF.
	Signaled System Error		x	Reports Fatal / non-fatal message per PF.
	Detected Parity Error		x	Reports receiving a poisoned TLP per PF.
Revision Register		x		
Class Code Register			x	Per function type.
Cache Line Size Register			x	Does not affect device behavior.
Latency Timer		x		Hard-wired to 00h in PCIe.
Header Type Register		x		
BIST		x		Not supported (00h).
BARs	Memory BAR		x	
	I/O BAR		x	
	MSI-X BAR		x	See Section 9.2.3.3, "MSI-X Capability".
I/O BAR mapping	IOADDR, IODATA		x	
Subsystem Vendor ID		x		
Subsystem ID		x		

Table 9-4. Configuration Sharing of PCI Configuration Space [continued]

Field	Sub-field	Shared?	Replicated?	Comments
Expansion ROM			x	
Cap_Ptr Register		x		
Interrupt Line Register			x	Just store the register value.
Interrupt Pin Register			x	Separate interrupt number (A-D) per PF.
Min. Grant		x		
Max Latency		x		

9.2.2.1 Vendor ID Register (0x0; RO)

This is a read-only register that has the same value for all PCI functions. It identifies unique Intel products. The value of this field is loaded from the *PCI_VENDORID* register loaded from NVM. The default value, if not loaded, is 0x8086.

9.2.2.2 Device ID Register (0x2; RO)

This is a read-only register that identifies individual X550 PCI functions. Both ports have the same default value equals to 0x1562, and can be auto-loaded from the NVM during initialization with different values for each port as well as the dummy function (See Section 4.4 for dummy function relevance).

The device ID values available for different the X550 SKUs are:

- 0x1563: X550 LOM & NIC. This is the value used for LOM & Sage Pond NIC.
- 0x1564: X550 VF for Hyper-V. This is the value used for VFs under Hyper-V (windows 8). This value is never exposed as part of the hardware configuration space.
- 0x1565: X550 VF for non-Hyper-V. This is the value used for VFs under “other” virtual environments.

9.2.2.3 Command Register (0x4; RW)

Shaded bits are not used by this implementation and are hard-wired to 0b. Each function has its own Command register. Unless explicitly specified, functionality is the same in both functions.

Bit(s)	Init.	Description
0	0b	**I/O Access Enable** If I/O BAR is not requested, this bit is hard-wired to zero.
1	0b	**Memory Access Enable**
2	0b	**Enable Mastering** Also named Bus Master Enable (BME). • LAN functions RW field. • Dummy function RO as zero field.
3	0b	**Special Cycle Monitoring** Hard-wired to 0b.
4	0b	**MWI Enable** Hard-wired to 0b.
5	0b	**Palette Snoop Enable** Hard-wired to 0b.
6	0b	**Parity Error Response**
7	0b	**Wait Cycle Enable** Hard-wired to 0b.
8	0b	**SERR# Enable**
9	0b	**Fast Back-to-Back Enable** Hard-wired to 0b.
10	0b	**Interrupt Disable** When set, devices are prevented from generating legacy interrupt messages.
15:11	0x0	Reserved.

9.2.2.4 Status Register (0x6; RO)

Shaded bits are not used by this implementation and are hard-wired to 0b. Each function has its own Status register. Unless explicitly specified, functionality is the same in both functions.

Bit(s)	Init.	Type	Description
2:0	000b		Reserved.
3	0b	RO	**Interrupt Status**[1]
4	1b	RO	**New Capabilities** Indicates that a device implements extended capabilities. The X550 sets this bit and implements a capabilities list to indicate that it supports PCI Power Management, Message Signaled Interrupts (MSI), Enhanced Message Signaled Interrupts (MSI-X), VPD and the PCIe extensions.
5	0b		**66 MHz Capable** Hard-wired to 0b.
6	0b		Reserved.
7	0b		**Fast Back-to-Back Capable** Hard-wired to 0b.
8	0b	RW1C	**Data Parity Reported**
10:9	00b		**DEVSEL Timing** Hard-wired to 0b.
11	0b	RW1C	**Signaled Target Abort**
12	0b	RW1C	**Received Target Abort**
13	0b	RW1C	**Received Master Abort**
14	0b	RW1C	**Signaled System Error**
15	0b	RW1C	**Detected Parity Error**

1. The *Interrupt Status* field is a RO field that indicates that an interrupt message is pending internally to the device.

9.2.2.5 Revision Register (0x8; RO)

The default revision ID of this device is 0x00 for X550 A0 and 0x01 for X550 B0. The value of the rev ID is a logic XOR between the default value and the value in the NVM PCIe Device Revision ID (PCI_REVID). Note that LAN 0 and LAN 1 functions have the same revision ID.

9.2.2.6 Class Code Register (0x9; RO)

The class code is a read-only value that identifies the device functionality according to the value of the *Storage Class* bit in the NVM *PCI_CLASS* NVM register.

- Class Code = 0x020000 (Ethernet Adapter) if NVM->*Storage Class* = 0b
- Class Code = 0x010000 (SCSI Storage device) if NVM->*Storage Class* = 1b

In the dummy function the class code equals to 0xFF0000.

9.2.2.7 Cache Line Size Register (0xC; RW)

This field is implemented by PCIe devices as a read/write field for legacy compatibility purposes but has no impact on any PCIe device functionality. The default value is zero.

9.2.2.8 Latency Timer (0xD; RO)

Not used. Hard-wired to 0b.

9.2.2.9 Header Type Register (0xE; RO)

This indicates if a device is single- or multi-function. If a single LAN function is the only active one, this field has a value of 0x00 to indicate a single function device. If other functions are enabled, this field has a value of 0x80 to indicate a multi-function device. Table 9-5 lists the different options to set the header type field.

Table 9-5. Header Type Settings

LAN 0 Enable	LAN 1 Enable	Cross-Mode Enable	Header Type Expected Value
0	0	X	N/A (no function)
1	0	0	0x00
0	1	0	0x80 (dummy exist)
1	1	X	0x80 (dual function)
1	0	1	0x80 (dummy exist)
0	1	1	0x00

9.2.2.10 Subsystem Vendor ID Register (0x2C; RO)

This value can be loaded automatically from the NVM at power up or reset. A value of 0x8086 is the default for this field at power up if the NVM does not respond or is not programmed. All functions are initialized to the same value.

9.2.2.11 Subsystem ID Register (0x2E; RO)

This value can be loaded automatically from the NVM at power up with a default value of 0x0000.

PCI Function	Default Value	NVM Address
LAN Functions	0x0000	0x0B

9.2.2.12 Cap_Ptr Register (0x34; RO)

The *Capabilities Pointer* field (*Cap_Ptr*) is an 8-bit field that provides an offset in the X550's PCI configuration space for the location of the first item in the capabilities linked list. The X550 sets this bit and implements a capabilities list to indicate that it supports PCI power management, MSIs, and PCIe extended capabilities. Its value is 0x40, which is the address of the first entry: PCI power management.

9.2.2.13 Interrupt Line Register (0x3C; RW)

Read/write register programmed by software to indicate which of the system interrupt request lines the X550's interrupt pin is bound to. Refer to the PCI definition for more details. Each PCI function has its own register.

9.2.2.14 Interrupt Pin Register (0x3D; RO)

Read-only register. LAN 0 / LAN 1 — A value of 0x1...0x4 indicates that this function implements a legacy interrupt on INTA#...INTD#, respectively. Loaded from the PCI_CNF NVM word per function. For dummy function the returned value is 0x0 (Function uses no legacy interrupt Message).

Note: If only a single device/function of the X550 component is enabled, this value is ignored and the *Interrupt Pin* field of the enabled device reports INTA# usage.

9.2.2.15 Max_Lat and Min_Gnt (0x3E; RO)

Not used. Hard-wired to 0b.

For Dummy functions this register is RO - zero.

9.2.2.16 Memory and IO Base Address Registers (0x10...0x27; RW)

Base Address Registers (BARs) are used to map the X550 register space of the device functions. The X550 has a memory BAR, I/O BAR and MSI-X BAR described in Table 9-6. The BARs location and sizes are described in the Table 9-6 and Table 9-8. The fields within each BAR are then described in Table 9-8.

Table 9-6. X550 Base Address Registers Description — LAN 0 / LAN 1

Mapping Windows	Mapping Description
Memory BAR	The internal registers memories and external Flash device are accessed as direct memory mapped offsets from the BAR. Software can access a DWord or 64 bits. The Flash space in this BAR is enabled by the *FL_BAR_SIZE* and *CSRSize* fields in the *PCI_LBARCTRL* register. Address 0 in the Flash device is mapped to address 256 K in the memory BAR. When the usable Flash size + CSR space are smaller than the memory BAR, accessing addresses above the top of the Flash wraps back to the beginning of the Flash.
I/O BAR	All internal registers and memories can be accessed using I/O operations. There are two 4-byte registers in the I/O mapping window: Addr Reg and Data Reg accessible as DWord entities. The I/O BAR is supported depending on the *IO_Sup* bit in the NVM at word *PCIe Control 3 – Offset 0x07*.
MSI-X BAR	The MSI-X vectors and Pending Bit Array (PBA) structures are accessed as direct memory mapped offsets from the MSI-X BAR. Software can access DWord entities.

Table 9-7. X550 Base Address Setting in 64-bit BARs Mode

BAR	Addr	31 5	4	3	2	1	0
0	0x10	Memory CSR + FLASH BAR Low see Table 9-8		0/1	1	0	0
1	0x14	Memory CSR + FLASH BAR High (RW)					
2	0x18	IO BAR (RW — 31:5)	0	0	0	0	1
3	0x1C	Reserved (RO — 0)					
4	0x20	MSI-X BAR Low (RW — 31:14; RO 0b — 13:4)		0/1	1	0	0
5	0x24	MSI-X BAR High (RW)					

Table 9-8. Base Address Registers Fields

Field	Bit(s)	Type	Description
Memory and I/O Space Indication	0	RO	**Memory and I/O Space Indication** 0b = Indicates memory space. 1b = Indicates I/O.
Memory Type	2:1	RO	**Memory Type** 00b = Reserved. 10b = 64-bit BAR
Prefetch Memory	3	RO	**Prefetch Memory** 0b = Non-prefetchable space. 1b = Prefetchable space. This bit should be set only in systems that do not generate prefetchable cycles. This bit is loaded from the *PREFBAR* bit in the NVM because it is required for 64-bit memory BARs.
Address Space (low register for 64-bit memory BARs)	31:4	RW	**Address Space** The length of the RW bits and RO 0b bits depend on the mapping window size. Initial value of the RW fields is 0x0. The size of the memory BAR is described in Table 9-9.

Mapping Window	RO bits
Memory CSR + Flash BAR size depends on PCI_LBARCTRL.*FL_BAR_SIZE and* PCI_LBARCTRL.*CSRSize fields.*	17:4 for 256 KB 18:4 for 512 KB and so on...
MSI-X space is 16 KB.	13:4
I/O space size is 32 bytes.	4:0

Table 9-9. Usable FLASH Size and CSR Mapping Window Size

FL_BAR_SIZE	CSRSize	Resulted CSR + FLASH BAR Size	Installed FLASH Device	Usable FLASH Space
000b-100b	X	Reserved		
101b	0	2 MB	2 MB	2 MB minus 256 KB
101b	1	4 MB	2 MB	2 MB
110b	0	4 MB	4 MB	4 MB minus 256 KB
110b	1	8 MB	4 MB	4 MB
111b	0	8 MB	8 MB	8 MB minus 256 KB
111b	1	16 MB	8 MB	8 MB

9.2.2.17 Expansion ROM Base Address Register (0x30; RW)

This register is used to define the address and size information for boot-time access to the Expansion ROM Module in the Flash memory. It is enabled by the PCI_LBARCTRL.*EXROM_DIS* register field. This register returns a zero value for functions without an expansion ROM window and for dummy functions.

Field	Bit(s)	Init.	Type	Description
En	0	0b	RW	**Enable** 0b = Disables expansion ROM access. 1b = Enables expansion ROM access.
Reserved	10:1	0x0	R	Reserved. Always read as 0x0. Writes are ignored.
Address	31:11	0x0	RW	**Address** Read-write bits are hard-wired to 0b and dependent on the memory mapping window size. The LAN Expansion ROM spaces can be either 64 KB or up to 8MBin powers of 2. Mapping window size is set by PCI_LBARCTRL.*EXROM_BAR_SIZE* field.

9.2.3 PCI Capabilities

The first entry of the PCI capabilities link list is pointed to by the Cap_Ptr register. Table 9-10 lists the capabilities supported by the X550.

Table 9-10. PCI Capabilities List for LAN Functions

Address	Item	Next Pointer
0x40:4F	PCI Power Management	0x50 / 0xA0[1]
0x50:6F	MSI	0x70
0x70:8F	MSI-X	0xA0
0xA0:DF	PCIe Capabilities	0xE0 / 0x00
0xE0:0xEF	VPD Capability	0x00

1. In the dummy function, the power management capability points to the PCIe capabilities.

Table 9-11. PCI Capabilities for Dummy Function

Address	Item	Next Pointer
0x40:47	PCI Power Management	0x50
0xA0:DB	PCIe Capabilities	0x00

9.2.3.1 PCI Power Management Capability

All fields are reset at full power up. All fields except *PME_En* and *PME_Status* are reset after exiting from the D3cold state. If AUX power is not supplied, the *PME_En* and *PME_Status* fields also reset after exiting from the D3cold state. Refer to the detailed description for registers loaded from the NVM at initialization.

Table 9-12. PCI Power Management Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x40	Power Management Capabilities		Next Pointer (0x50 / 0xA0)	Capability ID (0x01)
0x44	Data	Bridge Support Extensions	Power Management Control & Status	

Table 9-13 summarizes sharing of the Power Management Capability registers among the different PCI functions.

Table 9-13. Sharing the Power Management Capability Registers

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Power Management Capabilities	PME_Support	x		
	D2_Support	x		
	D1_Support	x		
	AUX Current	x		
	DSI	x		
	PME Clock	x		Hard-wired to 0b.
	Version	x		
Power Management Control / Status	PME_Status		x	
	Data_Scale	x		
	Data_Select		x	
	PME_En		x	
	No_Soft_Reset	x		
	PowerState		x	
Data Register			x	

9.2.3.1.1 Capability ID Register (0x40; RO)

This field equals 0x01 indicating the linked list item as being the PCI Power Management registers.

9.2.3.1.2 Next Pointer Register (0x41; RO)

This field provides an offset to the next capability item in the capability list. This field equals for both LAN ports to 0x50 pointing to the MSI capability. In dummy function, it equals to 0xA0 pointing to the PCIe Capabilities.

9.2.3.1.3 Power Management Capabilities — PMC Register (0x42; RO)

This register describes the device functionality during the power management states as listed in the following table. Note that each device function has its own register.

Bit(s)	Init.	Type	Description
2:0	011b	RO	**Version** The X550 complies with the PCI PM specification revision 1.2.
3	0b	RO	**PME_Clock** Disabled. Hard-wired to 0b.
4	0b	RO	Reserved.
5	1b	RO	**DSI** The X550 requires its device driver to be executed following a transition to the D0 uninitialized state.
8:6	000b	RO	**AUX Current** Required current defined in the Data register.
9	0b	RO	**D1_Support** The X550 does not support the D1 state.
10	0b	RO	**D2_Support** The X550 does not support the D2 state.
15:11	01001b	RO	**PME_Support** This 5-bit field indicates the power states in which the function can assert PME#. Condition Functionality Values: 01001b = No AUX Pwr - PME at D0 and D3hot 11001b = AUX Pwr - PME at D0, D3hot, and D3cold ***Note:*** For dummy function, this field is RO - zero.

9.2.3.1.4 Power Management Control/Status Register — PMCSR (0x44; RW)

This register is used to control and monitor power management events in the device. Note that each device function has its own PMCSR.

Bit(s)	Init.	Type	Description
1:0	00b	RW	**PowerState** This field is used to set and report the power state of a function as follows: 00b = D0 01b = D1 (cycle ignored if written with this value) 10b = D2 (cycle ignored if written with this value) 11b = D3
2	0b	RO	Reserved for PCIe.
3	1b	RO	**No_Soft_Reset** This bit is always set to 1b to indicate that the X550 does not perform an internal reset upon transition from D3hot to D0 via software control of the *PowerState* bits. Configuration context is kept as part of the transition from the D3hot to the D0 state, thus a full re-initialization sequence of the configuration space is not needed to return the X550 to the D0 Initialized state.
7:4	0x0	RO	Reserved.
8	0b at power up	RWS	**PME_En** If power management is enabled in the NVM. Writing a 1b to this register enables Wake-up.
12:9	0x0	RW	**Data_Select** This 4-bit field is used to select which data is to be reported through the Data register and *Data_Scale* field. These bits are writable only when power management is enabled via the NVM.

Bit(s)	Init.	Type	Description
14:13	01b	RO	**Data_Scale** This field indicates the scaling factor that is used when interpreting the value of the Data register. This field equals 01b (indicating 0.1 watt/units) if the *Data_Select* field is set to 0, 3, 4, 7, (or 8 for function 0 in multi-function device). Otherwise, it equals 00b.
15	0b at power up	RW1CS	**PME_Status** This bit is set to 1b when the function detects a wake-up event independent of the state of the *PME_En* bit. Writing a 1b clears this bit.

9.2.3.1.5 PMCSR_BSE Bridge Support Extensions Register (0x46; RO)

This register is not implemented in the X550; values set to 0x00.

9.2.3.1.6 Data Register (0x47; RO)

This optional register is used to report power consumption and heat dissipation. The reported register is controlled by the *Data_Select* field in the PMCSR; the power scale is reported in the *Data_Scale* field in the PMCSR. The data for this field is loaded from the NVM via the PCI_PWRDATA register if power management is enabled in the NVM or with a default value of 0x00. The values for the X550's functions are as follows (the relevant column is selected based on the value of the *Data_Select* field):

Function	D0 (Consume/ Dissipate)	D3 (Consume/ Dissipate)	Common	Data_Scale
Data_Select	(0x0/0x4)	(0x3/0x7)	(0x8)	
0	PCI_PWRDATA.*D0_POWER*	PCI_PWRDATA.*D3_POWER*	Function zero of a Multi-function device: PCI_PWRDATA.*COMM_POWER* Single-function device: 0x00	01b
1	PCI_PWRDATA.*D0_POWER*	PCI_PWRDATA.*D3_POWER*	0x00	01b

Note: For other *Data_Select* values the Data register output is reserved (0b).

9.2.3.2 MSI Capability

Note: This capability is not available for dummy functions.

This structure is required for PCIe devices.

Table 9-14. MSI Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x50	Message Control (0x0080)		Next Pointer (0x70)	Capability ID (0x05)
0x54	Message Address			
0x58	Message Upper Address			
0x5C	Reserved		Message Data	
0x60	Mask Bits			
0x64	Pending Bits			

Table 9-15 summarizes configuration sharing of the MSI Capability registers among the different PCI functions.

Table 9-15. Configuration Sharing of the MSI Capability

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Message Control	MSI Enable		x	
	Multiple Messages Capable	x		
	Multiple Message Enable		x	
	64-bit Capable	x		
	MSI per-vector masking	x		
Message Address Low			x	
Message Address High			x	
Message Data			x	
Mask Bits			x	
Pending Bits			x	

9.2.3.2.1 Capability ID Register (0x50; RO)

This field equals 0x05 indicating that the linked list item as being the MSI registers.

9.2.3.2.2 Next Pointer Register (0x51; RO)

This field provides an offset to the next capability item in the capability list. Its value of 0x70 and points to MSI-X capability.

9.2.3.2.3 Message Control Register (0x52; RW)

Note: There is a dedicated register (per PCI function) to separately enable its MSI.

Bit(s)	Init.	Type	Description
0	0b	RW	**MSI Enable** 1b = Message Signaled Interrupts — The X550 generates an MSI for interrupt assertion instead of INTx signaling.
3:1	000b	RO	**Multiple Messages Capable** The X550 indicates a single requested message per function.
6:4	000b	RW	**Multiple Message Enable** Since the X550 requests a single vector in the *Multiple Message Capable* field, software is expected to write 000b to this field.
7	1b	RO	**64-bit Capable** A value of 1b indicates that the X550 is capable of generating 64-bit message addresses.
8	1b[1]	RO	**MSI per-vector masking** A value of 0b indicates that the X550 is not capable of per-vector masking. A value of 1b indicates that the X550 is capable of per-vector masking.
15:9	0x0	RO	Reserved. Reads as 0x0.

1. The value is loaded from the MSI Mask bit in the NVM.

9.2.3.2.4 Message Address Low Register (0x54; RW)

Written by the system to indicate the lower 32 bits of the address to use for the MSI memory write transaction. The lower two bits always return 0b regardless of the write operation.

9.2.3.2.5 Message Address High Register (0x58; RW)

Written by the system to indicate the upper 32 bits of the address to use for the MSI memory write transaction.

9.2.3.2.6 Message Data Register (0x5C; RW)

Written by the system to indicate the lower 16 bits of the data written in the MSI memory write DWord transaction. The upper 16 bits of the transaction are written as 0b.

9.2.3.2.7 Mask Bits Register (0x60; RW)

The Mask Bits and Pending Bits registers enable software to disable or defer message sending on a per-vector basis. As the X550 supports only one message, only bit 0 of these registers are implemented.

Bit(s)	Init.	Type	Description
0	0b	RW	**MSI Vector 0 Mask** If set, the X550 is prohibited from sending MSI messages.
31:1	0x0	RO	Reserved.

9.2.3.2.8 Pending Bits Register (0x64; RW)

Bit(s)	Init.	Type	Description
0	0b	RO	**MSI Message** If set, the X550 has a pending MSI message.
31:1	0x0	RO	Reserved.

9.2.3.3 MSI-X Capability

Note: This capability is not available for dummy functions.

More than one MSI-X capability structure per function is prohibited while a function is permitted to have both an MSI and an MSI-X capability structure.

In contrast to the MSI capability structure, which directly contains all of the control/status information for the function's vectors, the MSI-X capability structure instead points to an MSI-X table structure and an MSI-X Pending Bit Array (PBA) structure, each residing in memory space.

Each structure is mapped by a BAR belonging to the function that begins at 0x10 in the configuration space. A BAR Indicator Register (BIR) indicates which BAR and a QWord-aligned offset indicates where the structure begins relative to the base address associated with the BAR. The BAR is 64-bit, but must map to the memory space. A function is permitted to map both structures with the same BAR or map each structure with a different BAR.

The MSI-X table structure (Section 9.2.3.4) typically contains multiple entries, each consisting of several fields: *Message Address*, *Message Upper Address*, *Message Data*, and *Vector Control*. Each entry is capable of specifying a unique vector.

The PBA structure [MSI-X Table Offset Register (0x74; RW)] contains the function's pending bits, one per table entry, organized as a packed array of bits within QWords. The last QWord is not necessarily fully populated.

To request service using a given MSI-X table entry, a function performs a DWord memory write transaction using:

- The contents of the *Message Data* field entry for data.
- The contents of the *Message Upper Address* field for the upper 32 bits of the address.
- The contents of the *Message Address* field entry for the lower 32 bits of the address.

A memory read transaction from the address targeted by the MSI-X message produces undefined results.

The MSI-X table and MSI-X PBA are permitted to co-reside within a naturally aligned 4 KB address range, though they must not overlap with each other.

MSI-X table entries and *Pending* bits are each numbered 0 through N-1, where N-1 is indicated by the *Table Size* field in the MSI-X Message Control register. For a given arbitrary MSI-X table entry K, its starting address can be calculated with the formula:

Entry starting address = Table base + K*16

For the associated *Pending* bit *K*, its address for QWord access and bit number within that QWord can be calculated with the formulas:

QWord address = PBA base + (K div 64)*8

QWord bit# = K mod 64

Software that chooses to read *Pending* bit *K* with DWord accesses can use these formulas:

DWord address = PBA base + (K div 32)*4

DWord bit# = K mod 32

Table 9-16. MSI-X Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x70	Message Control (0x00090)		Next Pointer (0xA0)	Capability ID (0x11)
0x74	Table Offset			
0x78	PBA Offset			

Table 9-17 summarizes configuration sharing of the MSI-X Capability registers among the different PCI functions.

Table 9-17. 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	

Table 9-17. Configuration Sharing of the MSI-X Capability [continued]

Field	Sub-field	Shared?	Replicated?	Comments
MSI-X Enable			x	
MSI-X Table Offset	Table BIR		x	
	Table Offset		x	
MSI-X Pending Bit Array	PBA BIR		x	
	PBA Offset		x	
MSI-X Table			x	
MSI-X PBA Structure			x	

9.2.3.3.1 Capability ID Register (0x70; RO)

This field equals 0x11 indicating that the linked list item as being the MSI-X registers.

9.2.3.3.2 Next Pointer Register (0x71; RO)

This field provides an offset to the next capability item in the capability list. Its value of 0xA0 points to PCIe capability.

9.2.3.3.3 Message Control Register (0x72; RW)

Note: There is a dedicated register (per PCI function).

Bit(s)	Init.	Type	Description
10:0	0x3F	RO	**Table Size** System software reads this field to determine the MSI-X Table Size N, which is encoded as N-1. The X550 supports up to 64 different interrupt vectors per function. This field is loaded from the PCI_CNF2.*MSI_X_PF_N* register field.
13:11	000b	RO	Reserved. Always returns 000b on a read. A write operation has no effect.
14	0b	RW	**Function Mask** 0b = Each vector's *Mask* bit determines whether the vector is masked or not. 1b = All of the vectors associated with the function are masked, regardless of their per-vector *Mask* bit states. Setting or clearing the MSI-X *Function Mask* bit has no effect on the state of the per-vector *Mask* bits.
15	0b	RW	**MSI-X Enable** 0b = The function is prohibited from using MSI-X to request service. 1b = If 1b and the MSI *Enable* bit in the MSI Message Control register is 0b, the function is permitted to use MSI-X to request service and is prohibited from using its INTx# pin. System configuration software sets this bit to enable MSI-X. A device driver is prohibited from writing this bit to mask a function's service request.

9.2.3.3.4 MSI-X Table Offset Register (0x74; RW)

Bit(s)	Init.	Type	Description
2:0	0x3/0x4	RO	**Table BIR** Indicates which one of a function's BARs, beginning at 0x10 in the configuration space, is used to map the function's MSI-X table into the memory space. while BIR values: 0...5 correspond to BARs 0x10...0x 24, respectively.
31:3	0x0	RO	**Table Offset** Used as an offset from the address contained in one of the function's BARs to point to the base of the MSI-X table. The lower three *Table BIR* bits are masked off (set to 0b) by software to form a 32-bit QWord-aligned offset. This field is read only.

9.2.3.3.5 MSI-X Pending Bit Array — PBA Offset (0x78; RW)

Bit(s)	Init.	Type	Description
2:0	0x4	RO	**PBA BIR** Indicates which one of a function's BARs, beginning at 0x10 in the configuration space, is used to map the function's MSI-X PBA into the memory space. while BIR values: 0...5 correspond to BARs 0x10...0x 24, respectively.
31:3	0x0400	RO	**PBA Offset** Used as an offset from the address contained in one of the functions BARs to point to the base of the MSI-X PBA. The lower three *PBA BIR* bits are masked off (set to 0b) by software to form a 32-bit QWord-aligned offset. This field is read only.

9.2.3.4 MSI-X Table Structure

DWord3 — MSIXTVCTRL	DWord2 — MSIXTMSG	DWord1 — MSIXTUADD	DWord0 — MSIXTADD	Entry Number	BAR 3 — Offset
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	0	Base (0x0000)
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	1	Base + 1*16
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	2	Base + 2*16
...	...	...	...	...	
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	63	Base + 63*16
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	64	Base + 64*16
...	...	...	...	...	
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	255	Base + 255*16

Note: All MSI-X vectors > MSI-X 63, are usable only by the Virtual Functions (VFs) in IOV mode. These vectors are not exposed to the operating system by the *Table Size* field in the *MSI-X Message Control* word.

See Section 8.2.2.7 for details of the MSI-X registers in BAR 3 of the PF.

9.2.3.5 VPD Registers

Note: This capability is not available for dummy functions.

The X550 supports access to a VPD structure stored in the NVM using the following set of registers.

Initial values of the configuration registers are marked in parenthesis.

Note: The VPD structure is available through both ports functions. As the interface is common to the two functions, accessing the VPD structure of one function while an access to the NVM is in process on the other function can yield to unexpected results.

Table 9-18. VPD Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xE0	VPD Address		Next Pointer (0x00)	Capability ID (0x03)
0xE4	VPD Data			

Table 9-19 summarizes configuration sharing of the VPD Capability registers among the different PCI functions.

Table 9-19. Configuration Sharing of the VPD Capability

Field	Shared?	Replicated?	Comments
Capability ID	X		
Next Pointer	X		
VPD Address/F	X		
VPD Data	X		

9.2.3.5.1 Capability ID Register (0xE0; RO)

This field equals 0x3 indicating the linked list item as being the VPD registers.

9.2.3.5.2 Next Pointer Register (0xE1; RO)

Offset to the next capability item in the capability list. A 0x00 value indicates that it is the last item in the capability-linked list.

9.2.3.5.3 VPD Address Register (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.

Bit(s)	Init.	Type	Description
14:0	X	RW	**Address** DWord-aligned byte address of the VPD area in the NVM to be accessed. The register is read/write, and the initial value at power-up is indeterminate. The two LSBs are RO as zero.
15	0b	RW	**F** A flag used to indicate when the transfer of data between the VPD Data register and the storage component completes. The Flag register is written when the VPD Address register is written. 0b = Read — Set by hardware when data is valid. 1b = Write — Cleared by hardware when data is written to the NVM. The VPD address and data should not be modified before the action is done.

9.2.3.5.4 VPD Data Register (0xE4; RW)

VPD read/write data.

Bit(s)	Init.	Type	Description
31:0	X	RW	**VPD Data** VPD data can be read or written through this register. The LS byte of this register (at offset 4 in this capability structure) corresponds to the byte of VPD at the address specified by the VPD Address register. The data read from or written to this register uses the normal PCI byte transfer capabilities. Four bytes are always transferred between this register and the VPD storage component. Reading or writing data outside of the VPD space in the storage component is not allowed. In a write access, the data should be set before the address and the flag is set.

9.2.3.6 PCIe Capability

The X550 implements the PCIe capability structure linked to the legacy PCI capability list for endpoint devices as follows:

Table 9-20. PCIe Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xA0	PCI Express Capability Register (0x0002)		Next Pointer (0xE0/0x00)	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			
0xD0	Link Status 2		Link Control 2	
0xD4	Reserved			
0xD8	Reserved		Reserved	

Table 9-21 summarizes configuration sharing of the PCIe Capability registers among the different PCI functions.

Table 9-21. 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	
	Auxiliary Power PM Enable		x	Same policy for all PFs (Logical OR of the PF's bits)
	Enable No Snoop		x	
	Max Read Request Size		x	Use minimum of all configured values.
	Initiate Function Level Reset		x	
Device Status	Correctable Detected		x	
	Non-Fatal Error Detected		x	
	Fatal Error Detected		x	
	Unsupported Request Detected	x		
	Aux Power Detected	x		
	Transactions Pending		x	
Link Capabilities	Supported Link Speeds	x		
	Max Link Width	x		
	Active State Link PM Support	x		
	L0s Exit Latency	x		
	L1 Exit Latency	x		
	Clock Power Management	x		
	Port Number	x		

Table 9-21. Configuration Sharing of the PCIe Capability [continued]

Field	Sub-field	Shared?	Replicated?	Comments
Link Control	Active State Link PM Control		x	Same policy for all PFs (Logical AND of the PF's bits). In ARI mode, use value in Function 0.
	Read Completion Boundary (RCB)		x	
	Common Clock Configuration		x	Same policy for all PFs (Logical AND of the PF's bits) In ARI mode, use value in Function 0.
	Extended Sync		x	Same policy for all PFs (Logical OR of the PF's bits).
Link Status	Current Link Speed	x		
	Negotiated Link Width	x		
	Slot Clock Configuration	x		
Device Capabilities 2	Completion Timeout Ranges Supported	x		
	Completion Timeout Disable Supported	x		
	LTR Mechanism Supported	x		
	TPH Completer Supported	x		
	Extended Fmt Field Supported	x		
	OBFF Supported	x		
Device Control 2	Completion Timeout Value		x	Completion timeout decision per PF or use the largest configured value among PFs.
	Completion Timeout Disable		x	Completion timeout mechanism enabled per PF.
	IDO Request Enable		x	
	IDO Completion Enable		x	
	LTR Mechanism Enable		x	PF0 only. RsvdP on other functions.
	OBFF Enable		x	PF0 only. RsvdP on other functions.
Link Capabilities 2		x		
Link Control 2		x		PF0 only. RsvdP on other functions.
Link Status 2		x		

9.2.3.6.1 Capability ID Register (0xA0; RO)

This field equals 0x10 indicating that the linked list item as being the PCIe Capabilities registers.

9.2.3.6.2 Next Pointer Register (0xA1; RO)

Offset to the next capability item in the capability list. Its value of 0xE0 points to the VPD structure. If VPD is disabled or for a dummy function, a value of 0x00 value indicates that it is the last item in the capability-linked list.

9.2.3.6.3 PCIe Capabilities Register (0xA2; RO)

The PCIe Capabilities register identifies PCIe device type and associated capabilities. This is a read-only register identical to all functions.

Bit(s)	Init.	Type	Description
3:0	0x2	RO	**Capability Version** Indicates the PCIe capability structure version. The X550 supports PCIe version 2 (also loaded from the PCI_CAPSUP.*PCIE_VER* bit in the NVM).
7:4	0x0	RO	**Device/Port Type** Indicates the type of PCIe functions. All functions are native PCI functions with a value of 0x0.
8	0b	RO	**Slot Implemented** The X550 does not implement slot options. Therefore, this field is hard-wired to 0b.
13:9	0x0	RO	**Interrupt Message Number** The X550 does not implement multiple MSI per function. As a result, this field is hard-wired to 0x0.
15:14	00b	RO	Reserved.

9.2.3.6.4 Device Capabilities Register (0xA4; RO)

This register identifies the PCIe device specific capabilities. It is a read-only register with the same value for the two LAN functions and for all other functions.

Bit(s)	Init.	Type	Description
2:0	010b	RO	**Max Payload Size Supported** This field indicates the maximum payload that The X550 can support for TLPs. It is loaded from the NVM with a default value of 512 bytes.
4:3	00b	RO	**Phantom Function Supported** Not supported by the X550.
5	0b	RO	**Extended Tag Field Supported** Maximum supported size of the *Tag* field. The X550 supports a 5-bit *Tag* field for all functions.
8:6	011b	RO	**Endpoint L0s Acceptable Latency** This field indicates the acceptable latency that the X550 can withstand due to the transition from L0s state to the L0 state. All functions share the same value loaded from the NVM PCIe Init Configuration 1 bits [8:6]. A value of 011b equals 512 ns.
11:9	110b	RO	**Endpoint L1 Acceptable Latency** This field indicates the acceptable latency that the X550 can withstand due to the transition from L1 state to the L0 state. A value of 110b equals 32 µs to 64 µs. All functions share the same value loaded from the NVM.
12	0b	RO	**Attention Button Present** Hard-wired in the X550 to 0b for all functions.
13	0b	RO	**Attention Indicator Present** Hard-wired in the X550 to 0b for all functions.
14	0b	RO	**Power Indicator Present** Hard-wired in the X550 to 0b for all functions.
15	1b	RO	**Role Based Error Reporting** Hard-wired in the X550 to 1b for all functions.
17:16	000b	RO	Reserved.
25:18	0x0	RO	**Slot Power Limit Value** Used in upstream ports only. Hard-wired in the X550 to 0x0 for all functions.
27:26	00b	RO	**Slot Power Limit Scale** Used in upstream ports only. Hard-wired in the X550 to 0b for all functions.

Bit(s)	Init.	Type	Description
28	1b	RO	**Function Level Reset Capability** A value of 1b indicates the Function supports the optional Function Level Reset (FLR) mechanism.
31:29	000b	RO	Reserved.

9.2.3.6.5 Device Control Register (0xA8; RW)

This register controls the PCIe specific parameters. Note that there is a dedicated register per each function.

Bit(s)	Init.	Type	Description
0	0b	RW	**Correctable Error Reporting Enable** Enable error report.
1	0b	RW	**Non-Fatal Error Reporting Enable** Enable error report.
2	0b	RW	**Fatal Error Reporting Enable** Enable error report.
3	0b	RW	**Unsupported Request Reporting Enable** Enable error report.
4	1b	RW	**Enable Relaxed Ordering** If this bit is set, the X550 is permitted to set the *Relaxed Ordering* bit in the *Attribute* field of write transactions that do not need strong ordering. Refer to the CTRL_EXT register bit *RO_DIS* for more details.
7:5	000b (128 bytes)	RW	**Max Payload Size** This field sets the maximum TLP payload size for the X550 functions. As a receiver, the X550 must handle TLPs as large as the set value. As a transmitter, the X550 must not generate TLPs exceeding the set value. The *Max Payload Size* field supported in the Device Capabilities register indicates permissible values that can be programmed. 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	0b	RW	**Extended Tag field Enable** Not implemented in the X550.
9	0b	RW	**Phantom Functions Enable** Not implemented in the X550.
10	0b	RWS	**Auxiliary Power PM Enable** When set, enables the X550 to draw AUX power independent of PME AUX power. The X550 is a multi-function device, therefore allowed to draw AUX power if at least one of the functions has this bit set.
11	0b	RW	**Enable No Snoop** No-snoop is not used by the X550.
14:12	010b	RW	**Max Read Request Size** This field sets maximum read request size for the X550 as a requester. 000b = 128 bytes 001b = 256 bytes 010b = 512 bytes 011b = 1024 bytes 100b = 2048 bytes 101b = 4096 bytes (Not supported by the X550) 110b = Reserved 111b = Reserved
15	0b	RW	**Initiate FLR** A write of 1b initiates FLR to the function. The value read by software from this bit is always 0b.

9.2.3.6.6 Device Status Register (0xAA; RW1C)

This register provides information about PCIe device specific parameters. Note that there is a dedicated register per each function.

Bit(s)	Init.	Type	Description
0	0b	RW1C	**Correctable Detected** Indicates status of correctable error detection.
1	0b	RW1C	**Non-Fatal Error Detected** Indicates status of non-fatal error detection.
2	0b	RW1C	**Fatal Error Detected** Indicates status of fatal error detection.
3	0b	RW1C	**Unsupported Request Detected** Indicates that the X550 received an unsupported request. This field is identical in all functions. The X550 cannot distinguish which function causes the error.
4	0b	RO	**Aux Power Detected** If Aux Power is detected, this field is set to 1b. It is a strapping signal from the periphery and is identical for all functions. Resets on LAN Power Good and PE_RST_N only.
5	0b	RO	**Transaction Pending** Indicates whether the X550 has ANY transactions pending. (transactions include completions for any outstanding non-posted request for all used traffic classes).
15:6	0x0	RO	Reserved.

9.2.3.6.7 Link Capabilities Register (0xAC; RO)

This register identifies PCIe link-specific capabilities. This is a read-only register identical to all functions.

Bit(s)	Init.	Type	Description
3:0	0x1	RO	**Supported Max Link Speed** This field indicates the supported Link speed(s) of the associated link port. Defined encodings are: 0001b = 2.5 GT/s link speed supported. 0010b = 5 GT/s and 2.5 GT/s link speeds supported. 0011b = 8 GT/s and 5 GT/s and 2.5 GT/s link speeds supported
9:4	0x04 - X550-AT2 0x08 - X550-BT2	RO	**Max Link Width** Indicates the maximum link width. The X550 supports a x1, x4 and x8-link width. This field is loaded from the PCIe Analog Configuration NVM module by interpreting the masked lanes, with a default value of eight lanes for the X550-BT2 and four for the X550-AT2. Defined encoding: 000000b = Reserved 000001b = x1 000010b = Reserved 000100b = x4 001000b = x8
11:10	10b	RO	**Active State Link PM Support** Indicates the level of the active state of power management supported in the X550. Defined encodings are: 00b = No ASPM support. 01b = L0s Entry supported. 10b = L1 supported. 11b = L0s and L1 supported. All functions share the same value loaded from the NVM.

Bit(s)	Init.	Type	Description
14:12	101b (1 µs – 2 µs)	RO	**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.
17:15	100b (8 µs — 16 µs)	RO	**L1 Exit Latency** Indicates the exit latency from L1 to L0 state. 000b = Less than 1 µs 001b = 1 µs — 2 µs 010b = 2 µs — 4 µs 011b = 4 µs — 8 µs 100b = 8 µs — 16 µs 101b = 16 µs — 32 µs 110b = 32 µs — 64 µs 111b = L1 transition not supported. All functions share the same value loaded from the NVM.
18	0	RO	**Clock Power Management**
19	0	RO	**Surprise Down Error Reporting Capable** Hard-wired to 0b.
20	0	RO	**Data Link Layer Link Active Reporting Capable**
21	0	RO	**Link Bandwidth Notification Capability** Hard-wired to 0b.
22	1b	RO	**ASPM Optional Compliance** This bit must be set to 1b. Components that were implemented according to an earlier PCIe specification version has this bit set to 0b. Software is permitted to use the value of this bit to help determine whether to enable ASPM or run ASPM compliance tests.
23	0b	RO	Reserved.
31:24	0x0	HwInit	**Port Number** The PCIe port number for the given PCIe link. This field is set in the link training phase.

9.2.3.6.8 Link Control Register (0xB0; RO)

This register controls PCIe link specific parameters. There is a dedicated register per each function.

Bit(s)	Init.	Type	Description
1:0	00b	RW	**Active State Link PM Control** This field controls the active state PM supported on the link. Link PM functionality is determined by the lowest common denominator of all functions. For non-ARI mode, only capabilities enabled in all functions are enabled for the component as a whole. When ARI support is exposed, ASPM control is determined solely by the setting in Function 0 (even when it is a dummy function), regardless of Function 0's D-state. The settings in the other functions always return whatever value software programmed for each, but otherwise are ignored by the X550. 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 while it is meaningless in the other function(s).
2	0b	RO	Reserved.
3	0b	RO	**Read Completion Boundary**
4	0b	RO	**Link Disable** Reserved for endpoint devices. Hard-wired to 0b.
5	0b	RO	**Retrain Clock** Not applicable for endpoint devices. Hard-wired to 0b.
6	0b	RW	**Common Clock Configuration** When set, indicates that the X550 and the component at the other end of the link are operating with a common reference clock. A value of 0b indicates that they are operating with an asynchronous clock. In ARI mode, the common clock configuration is determined solely by the field in function 0 (even when it is a dummy function) while it is meaningless in the other function(s).
7	0b	RW	**Extended Sync** When set, this bit forces an extended Tx of the FTS ordered set in FTS and an extra TS1 at the exit from L0s prior to entering L0.
8	0b	RO	**Enable Clock Power Management** Not supported in the X550. Hard-wired to 0.
9	0b	RW	Reserved. Returns the value that was written.
10	0b	RO	**Link Bandwidth Management Interrupt Enable** Not supported in the X550. Hard-wired to 0.
11	0b	RO	**Link Autonomous Bandwidth Interrupt Enable** Not supported in the X550. Hard-wired to 0.
15:12	0x0	RO	Reserved.

9.2.3.6.9 Link Status Register (0xB2; RO)

This register provides information about PCIe Link specific parameters. This is a read only register identical to all functions.

Bit(s)	Init.	Type	Description
3:0	X	RO	**Current Link Speed** This field indicates the negotiated link speed of the given PCIe link. Defined encodings are: 0001b = 2.5 GT/s PCIe link. 0010b = 5 GT/s PCIe link. 0011b = 8 GT/s PCIe link. All other encodings are reserved. The value in this field is undefined when the Link is not up.
9:4	X	RO	**Negotiated Link Width** Indicates the negotiated width of the link. Relevant encodings for the X550 are: 000001b = x1 000010b = Reserved 000100b = x4 001000b = x8 The value in this field is undefined when the Link is not up.
10	0b	RO	Undefined.
11	0b	RO	**Link Training** Indicates that link training is in progress. This field is not applicable and is reserved for endpoint devices, and is hard-wired to 0b.
12	1b	HwInit	**Slot Clock Configuration** When set, indicates that the X550 uses the physical reference clock that the platform provides at the connector. This bit must be cleared if the X550 uses an independent clock. The *Slot Clock Configuration* bit is loaded from the *Slot_Clock_Cfg* NVM bit.
13	0b	RO	**Data Link Layer Link Active** Not supported in the X550. Hard-wired to 0b.
14	0b	RO	**Link Bandwidth Management Status** Not supported in the X550. Hard-wire to 0b.
15	0b	RO	**Link Autonomous Bandwidth Status** This bit is not applicable and is reserved for endpoints.

The following registers are supported only if the capability version is two and above.

9.2.3.6.10 Device Capability 2 Register (0xC4; RO)

This register identifies the PCIe device-specific capabilities. It is a read-only register with the same value for both LAN functions.

Bit(s)	Init.	Type	Description
3:0	0xF	RO	**Completion Timeout Ranges Supported** This field indicates the X550'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 X550 supports. 0000b = Completion timeout programming not supported. The X550 must implement a timeout value in the range of 50 µs to 50 ms. 0001b = Range A. 0010b = Range B. 0011b = Ranges A and B. 0110b = Ranges B and C. 0111b = Ranges A, B and C. 1110b = Ranges B, C and D. 1111b = Ranges A, B, C and D. All other values are reserved.
4	1b	RO	**Completion Timeout Disable Supported** A value of 1b indicates support for the completion timeout disable mechanism. ***Note:*** For dummy functionality, a completion timeout is not relevant as a dummy function because it never sends non-posted requests.
5	0b	RO	**ARI Forwarding Supported** Applicable only to Switch Downstream. Ports and Root Ports; must be 0b for other function types.
10:6	0x0	RO	**Not supported** Hard-wired to 0x0.
11	1b	RO	**LTR Mechanism Supported** A value of 1b indicates support for the optional Latency Tolerance Reporting (LTR) mechanism. For a multi-Function device associated with an Upstream Port, each Function must report the same value for this bit. ***Note:*** Value loaded from *LTR_EN* bit in the NVM
13:12	00b	RO	**TPH Completer Supported** Value indicates Completer support for TPH or Extended TPH. This capability is not supported.
17:14	0x0	RO	Reserved.
19:18	00b	RO	**OBFF Supported** 00b = OBFF not supported. 01b = OBFF supported using Message signaling only.. 10b = OBFF supported using WAKE# signaling only 11b = OBFF supported using WAKE# and Message signaling.
20	0b	RO	**Extended Fmt Field Supported** 0b = The Function supports a 2-bit definition of the *Fmt* field. 1b = The Function supports the 3-bit definition of the *Fmt* field. Not supported by this device
21	0b	RO	**End-End TLP Prefix Supported** Indicates whether End-End TLP Prefix support is offered by a Function. Not supported by this device.

Bit(s)	Init.	Type	Description
23:22	00b	RsvdP	**Max End-End TLP Prefixes** Indicates the maximum number of End-End TLP Prefixes supported by this Function. Reserved for this device.
31:24	0x0	RO	Reserved.

9.2.3.6.11 Device Control 2 Register (0xC8; RW)

This register controls the PCIe specific parameters. Note that there is a dedicated register per each function.

Bit(s)	Init.	Type	Description
3:0	0x0	RW	**Completion Timeout Value** For devices that support completion timeout programmability, this field enables system software to modify the completion timeout value. Defined encodings: 0000b = Default range: 16 ms to 32 ms. ***Note:*** It is strongly recommended that the completion timeout mechanism not expire in less than 10 ms. Values available if Range A (50 µs to 10 ms) programmability range is supported: 0001b = 50 µs to 100 µs. 0010b = 1 ms to 2 ms. Values available if Range B (10 ms to 250 ms) programmability range is supported: 0101b = 16 ms to 32 ms. 0110b = 65 ms to 130 ms. Values available if Range C (250 ms to 4 s) programmability range is supported: 1001b = 260 ms to 520 ms. 1010b = 1 s to 2 s. Values available if the Range D (4 s to 64 s) programmability range is supported: 1101b = 4 s to 8 s. 1110b = 17 s to 34 s. Values not defined are reserved. Software is permitted to change the value of this field at any time. For requests already pending when the completion timeout value is changed, hardware is permitted to use either the new or the old value for the outstanding requests and is permitted to base the start time for each request either on when this value was changed or on when each request was issued. ***Note:*** For dummy function, this field is RO - zero.
4	0b	RW	**Completion Timeout Disable** When set to 1b, this bit disables the completion timeout mechanism. Software is permitted to set or clear this bit at any time. When set, the completion timeout detection mechanism is disabled. If there are outstanding requests when the bit is cleared, it is permitted but not required for hardware to apply the completion timeout mechanism to the outstanding requests. If this is done, it is permitted to base the start time for each request on either the time this bit was cleared or the time each request was issued. ***Note:*** For dummy function, this field is RO - zero.
5	0b	RO	**ARI Forwarding Enable** Applicable only to switch devices.
7:6	00b	RO	Not supported. Hard-wired to 00b.
8	0b	RW	**IDO Request Enable** If this bit is Set, the Function is permitted to set the ID-Based Ordering (*IDO*) bit (Attribute[2]) of Requests it initiates. Default value of this bit is 0b.
9	0b	RW	**IDO Completion Enable** If this bit is Set, the Function is permitted to set the ID-Based Ordering (*IDO*) bit (Attribute[2]) of Completions it returns Default value of this bit is 0b.

Bit(s)	Init.	Type	Description
10	0b	RW / RsvdP	**LTR Mechanism Enable** When Set to 1b, this bit enables Upstream Ports to send LTR messages. For a multi-function device, the bit in Function 0 is RW, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is RsvdP. If the *LTR_EN* bit in the NVM is 0b, this bit is RO with a value of 0b. Default value of this bit is 0b.
12:11	00b	RO	Reserved
14:13	00b	RsvdP	**OBFF Enable** 00b = Disabled 01b = Enabled using Message signaling [Variation A] 10b = Enabled using Message signaling [Variation B] 11b = Enabled using WAKE# signaling
15	0b	RsvdP	**End-End TLP Prefix Blocking** Not applicable to endpoints.

9.2.3.6.12 Link Capabilities 2 Register (0xCC)

Bit(s)	Init.	Type	Description
0	0b	RO	Reserved.
7:1	0x01	RO	**Supported Link Speeds Vector** This field indicates the supported Link speed(s) of the associated Port. For each bit, a value of 1b indicates that the corresponding Link speed is supported; otherwise, the Link speed is not supported. Bit definitions are: Bit 1 = 2.5 GT/s Bit 2 = 5.0 GT/s Bit 3 = 8.0 GT/s 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 is reflected in the PCI_LINKCAP register.
8	0b	RO	**Crosslink Supported** When set to 1b, this bit indicates that the associated Port supports cross links. It is recommended that this bit be Set in any Port that supports cross-links even though doing so is only required for Ports that also support operating at 8.0 GT/s or higher Link speeds.
31:9	0x0	RO	Reserved.

9.2.3.6.13 Link Control 2 Register (0xD0; RWS)

All RW fields in this register affect the device behavior only through function 0. In function 1, these fields are reserved read as zeros.

Bit(s)	Init.	Type	Description
3:0	0x1 (func 0) 0x0 (else)	RWS (func 0) RsvdP (else)	**Target Link Speed** This field is used to set the target compliance mode speed when software is using the *Enter Compliance* bit to force a link into compliance mode. The encoding is the binary value of the bit in the Supported Link Speeds Vector (in the Link Capabilities 2 register) that corresponds to the desired target Link speed. All other encodings are Reserved. For example, 5.0 GT/s corresponds to bit 2 in the Supported Link Speeds Vector, so the encoding for a 5.0 GT/s target Link speed in this field is 0010b If a value is written to this field that does not correspond to a speed included in the *Supported Link Speeds* field, the result is undefined. The default value of this field is the highest link speed supported by the X550 (as reported in the *Supported Link Speeds* field of the Link Capabilities register).

Bit(s)	Init.	Type	Description
4	0b	RWS (func 0) RsvdP (else)	**Enter Compliance** Software is permitted to force a link to enter compliance mode at the speed indicated in the *Target Link Speed* field by setting this bit to 1b in both components on a link and then initiating a hot reset on the link. The default value of this field following a fundamental reset is 0b.
5	0b	RWS (func 0) RsvdP (else)	**Hardware Autonomous Speed Disable** When set to 1b, this bit disables hardware from changing the link speed for reasons other than attempting to correct unreliable link operation by reducing link speed.
6	0b	RO	**Selectable De-Emphasis** This bit is not applicable and reserved for endpoints.
9:7	000b	RWS (func 0) RsvdP (else)	**Transmit Margin** This field controls the value of the non de emphasized voltage level at the Transmitter pins. Encodings: 000b = Normal operating range. 001b = 800-1200 mV for full swing and 400-700 mV for half-swing. 010b = (n-1) — Values must be monotonic with a non-zero slope. The value of n must be greater than 3 and less than 7. At least two of these must be below the normal operating range of n: 200-400 mV for full-swing and 100-200 mV for half-swing. 111b = (n) Reserved.
10	0b	RWS (func 0) RsvdP (else)	**Enter Modified Compliance** When this bit is set to 1b, the device transmits modified compliance pattern if the LTSSM enters Polling.Compliance state. The default value of this bit is 0b.
11	0b	RWS (func 0) RsvdP (else)	**Compliance SOS** When set to 1b, the LTSSM is required to send SOS periodically in between the (modified) compliance patterns. This bit is applicable when the Link is operating at 2.5 GT/s or 5 GT/s data rates only. The default value of this bit is 0b.
15:12	0x0	RWS (func 0) RsvdP (else)	**Compliance Preset/De-emphasis** For 8.0 GT/s Data Rate: This field sets the Transmitter Preset in Polling.Compliance state if the entry occurred due to the *Enter Compliance* bit being 1b. For 5.0 GT/s Data Rate: This field sets the de-emphasis level in Polling.Compliance state if the entry occurred due to the *Enter Compliance* bit being 1b. When the Link is operating at 2.5 GT/s, the setting of this bit field has no effect. Defined Encodings are: 0001b -3.5 dB 0000b -6 dB For a Multi-Function device associated with an Upstream Port, the bit field in Function 0 is of type RWS, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit field is of type RsvdP. This bit field is intended for debug, and compliance testing purposes. The default value of this field is 0000b

9.2.3.7 Link Status 2 Register (0xD2; RW)

Bit(s)	Init.	Type	Description
0	0b	RO	**Current De-emphasis Level** When the link is operating at 5 GT/s speed, this bit reflects the level of de-emphasis. it is undefined when the Link is operating at 2.5 GT/s speed Encodings: 0b = -6 dB 1b = -3.5 dB
1	0b	ROS (Func 0) RsvdZ (Func 1)	**Equalization Complete** When set to 1b, this bit indicates that the Transmitter Equalization procedure has completed
2	0b	ROS (Func 0) RsvdZ (Func 1)	**Equalization Phase 1 Successful** When set to 1b, this bit indicates that Phase 1 of the Transmitter Equalization procedure has successfully completed.
3	0b	ROS (Func 0) RsvdZ (Func 1)	**Equalization Phase 2 Successful** When set to 1b, this bit indicates that Phase 2 of the Transmitter Equalization procedure has successfully completed.
4	0b	ROS (Func 0) RsvdZ (Func 1)	**Equalization Phase 3 Successful** When set to 1b, this bit indicates that Phase 3 of the Transmitter Equalization procedure has successfully completed.
5	0b	RW1C (Func 0) RsvdZ (Func 1)	**Link Equalization Request** This bit is Set by hardware to request the Link equalization process to be performed on the Link. This bit is available only in function zero.
15:6	0x0	RsvdZ	Reserved.

9.2.4 PCIe Extended Configuration Space

PCIe configuration space is located in a flat memory-mapped address space. PCIe extends the configuration space beyond the 256 bytes available for PCI to 4096 bytes. The X550 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 X550 supports the following PCIe extended capabilities:

Table 9-22. Extended Capabilities List

Capability	Offset	Next Header	Section Number
Advanced Error Reporting	0x100	Any of the below / 0x000	9.2.4.1
Serial Number	0x140	Any of the below / 0x000[1]	9.2.4.2
Alternative RID Interpretation (ARI)	0x150	Any of the below / 0x000[1]	9.2.4.3
IOV Support[2]	0x160	Any of the below / 0x000	9.2.4.4
TPH Requester	0x1A0	Any of the below / 0x000	9.2.4.5
Access Control Services (ACS)[3]	0x1B0	Any of the below / 0x000	9.2.4.6
Latency Tolerance Reporting (LTR)	0x1C0	Any of the below / 0x000	9.2.4.7
Secondary PCI Express	0x1D0	0x000	9.2.4.8

1. Depends on NVM settings enabling the ARI/IOV structures.
2. In a dummy function, the IOV and TPH structures are not exposed.
3. When in Single function mode (function 1 is disabled), the ACS capability is not exposed.

9.2.4.1 Advanced Error Reporting Capability (AER)

The PCIe advanced error reporting capability is an optional extended capability to support advanced error reporting. The tables that follow list the PCIe advanced error reporting extended capability structure for PCIe devices.

Table 9-23. AER Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x100	Next Capability offset	Version (0x1)	AER Capability ID (0x0001)	
0x104	Uncorrectable Error Status			
0x108	Uncorrectable Error Mask			
0x10C	Uncorrectable Error Severity			
0x110	Correctable Error Status			
0x114	Correctable Error Mask			
0x118	Advanced Error Capabilities and Control Register			
0x11C... 0x128	Header Log			

Table 9-24 summarizes configuration sharing of the AER Capability registers among the different PCI functions.

Table 9-24. 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 3.1.2.7.
Header Log			x	

9.2.4.1.1 Advanced Error Reporting Enhanced Capability Header Register (0x100; RO)

Bit(s)	Init.	Type	Description
15:0	0x1	RO	**Extended Capability ID** PCIe extended capability ID indicating advanced error reporting capability.
19:16	0x2	RO	**Version Number** PCIe advanced error reporting extended capability version number.
31:20	See description	RO	**Next Capability Offset** Next PCIe extended capability offset. See Table 9-22 and Table 9-11 for possible values of the next capability offset.

9.2.4.1.2 Uncorrectable Error Status Register (0x104; RW1CS)

The Uncorrectable Error Status register reports error status of individual uncorrectable error sources on a PCIe device. An individual error status bit that is set to 1b indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by LAN_PWR_GOOD.

Bit(s)	Init.	Type	Description
0	0b	RO	Reserved.
3:1	000b	RsvdP	Reserved.
4	0b	RW1CS	**Data Link Protocol Error Status**
5	0b	RO	Reserved.
11:6	0x0	RsvdP	Reserved.
12	0b	RW1CS	**Poisoned TLP Status**
13	0b	RW1CS	**Flow Control Protocol Error Status**

Bit(s)	Init.	Type	Description
14	0b	RW1CS	**Completion Timeout Status**
15	0b	RW1CS	**Completer Abort Status**
16	0b	RW1CS	**Unexpected Completion Status**
17	0b	RW1CS	**Receiver Overflow Status**
18	0b	RW1CS	**Malformed TLP Status.**
19	0b	RW1CS	**ECRC Error Status**
20	0b	RW1CS	**Unsupported Request Error Status**
21	0b	RO	**ACS Violation Status** Not supported. Hard-wired to 0b.
25:22	0x0	RO	Not Supported.
31:26	0x0	RsvdP	Reserved.

9.2.4.1.3 Uncorrectable Error Mask Register (0x108; RWS)

The Uncorrectable Error Mask register controls reporting of individual uncorrectable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. Note that there is a mask bit per bit of the Uncorrectable Error Status register.

Bit(s)	Init.	Type	Description
0	0b	RO	Reserved.
3:1	000b	RsvdP	Reserved.
4	0b	RWS	**Data Link Protocol Error Mask**
5	0b	RO	Reserved.
11:6	0x0	RsvdP	Reserved.
12	0b	RWS	**Poisoned TLP Mask**
13	0b	RWS	**Flow Control Protocol Error Mask**
14	0b	RWS	**Completion Timeout Mask**
15	0b	RWS	**Completer Abort Mask**
16	0b	RWS	**Unexpected Completion Mask**
17	0b	RWS	**Receiver Overflow Mask**
18	0b	RWS	**Malformed TLP Mask**
19	0b	RWS	**ECRC Error Mask**
20	0b	RWS	**Unsupported Request Error Mask**
21	0b	RO	**ACS Violation Mask**
25:22	0x0	RO	Not Supported.
31:26	0x0	RsvdP	Reserved.

9.2.4.1.4 Uncorrectable Error Severity Register (0x10C; RWS)

The Uncorrectable Error Severity register controls whether an individual uncorrectable error is reported as a fatal error. An uncorrectable error is reported as fatal when the corresponding error bit in the severity register is set. If the bit is cleared, the corresponding error is considered non-fatal.

Bit(s)	Init.	Type	Description
0	0b	RO	Reserved.
3:1	000b	RsvdP	Reserved.
4	1b	RWS	**Data Link Protocol Error Severity**
5	0b	RO	Reserved.
11:6	0x0	RsvdP	Reserved.
12	0b	RWS	**Poisoned TLP Severity**
13	1b	RWS	**Flow Control Protocol Error Severity**
14	0b	RWS	**Completion Timeout Severity**
15	0b	RWS	**Completer Abort Severity**
16	0b	RWS	**Unexpected Completion Severity**
17	1b	RWS	**Receiver Overflow Severity**
18	1b	RWS	**Malformed TLP Severity**
19	0b	RWS	**ECRC Error Severity**
20	0b	RWS	**Unsupported Request Error Severity**
21	0b	RO	**ACS Violation Severity**
25:22	0x0	RO	Not Supported.
31:26	0x0	RsvdP	Reserved.

9.2.4.1.5 Correctable Error Status Register (0x110; RW1CS)

The Correctable Error Status register reports error status of individual correctable error sources on a PCIe device. When an individual error status bit is set to 1b it indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by LAN_PWR_GOOD.

Bit(s)	Init.	Type	Description
0	0b	RW1CS	**Receiver Error Status**
5:1	0x0	RsvdZ	Reserved.
6	0b	RW1CS	**Bad TLP Status**
7	0b	RW1CS	**Bad DLLP Status**
8	0b	RW1CS	**REPLAY_NUM Rollover Status**
11:9	000b	RsvdZ	Reserved.
12	0b	RW1CS	**Replay Timer Timeout Status**
13	0b	RW1CS	**Advisory Non-Fatal Error Status**
15:14	00b	RO	Reserved.

9.2.4.1.6 Correctable Error Mask Register (0x114; RWS)

The Correctable Error Mask register controls reporting of individual correctable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. There is a mask bit per bit in the Correctable Error Status register.

Bit(s)	Init.	Type	Description
0	0b	RWS	**Receiver Error Mask**
5:1	0x0	RsvdP	Reserved.
6	0b	RWS	**Bad TLP Mask**
7	0b	RWS	**Bad DLLP Mask**
8	0b	RWS	**REPLAY_NUM Rollover Mask**
11:9	000b	RsvdP	Reserved.
12	0b	RWS	**Replay Timer Timeout Mask**
13	1b	RWS	**Advisory Non-Fatal Error Mask** This bit is set by default to enable compatibility with software that does not comprehend Role-Based Error Reporting.
15:14	00b	RO	Reserved.

9.2.4.1.7 Advanced Error Capabilities and Control Register (0x118; RO)

Bit(s)	Init.	Type	Description
4:0	0x0	ROS	**Vector** Vector pointing to the first recorded error in the Uncorrectable Error Status register. This is a read-only field that identifies the bit position of the first uncorrectable error reported in the Uncorrectable Error Status register.
5	1b	RO	**ECRC Generation Capable** If set, this bit indicates that the function is capable of generating ECRC. This bit is loaded from NVM. It is reflected in the PCI_CAPSUP register.
6	0b	RWS	**ECRC Generation Enable** When set, ECRC generation is enabled.
7	1b	RO	**ECRC Check Capable** If set, this bit indicates that the function is capable of checking ECRC. This bit is loaded from NVM. It is reflected in the PCI_CAPSUP register.
8	0b	RWS	**ECRC Check Enable** When set Set, ECRC checking is enabled.
9	0b	RO	**Multiple Header Recording Capable** Not Supported. Hard-wired to 0b
10	0b	RO	**Multiple Header Recording Enable** Not Supported. Hard-wired to 0b
11	0b	RsvdP	**TLP Prefix Log Present** Not supported. Hard-wired to 0b
15:12	0x0	RsvdP	Reserved.

9.2.4.1.8 Header Log Register (0x11C:128; RO)

The header log register captures the header for the transaction that generated an error. This register is 16 bytes.

Bit(s)	Init.	Type	Description
127:0	0x0	ROS	**Header Log Register** Header of the packet in error (TLP or DLLP).

9.2.4.2 Serial Number Capability

The PCIe device serial number capability is an optional extended capability that can be implemented by any PCIe device. The device serial number is a read-only 64-bit value that is unique for a given PCIe device.

The X550 implements this capability on all the functions and returns the same value in both.

Table 9-25. Serial Number Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x140	Next Capability Offset	Version (0x1)	Serial ID Capability ID (0x0003)	
0x144	Serial Number Register (Lower DWord)			
0x148	Serial Number Register (Upper DWord)			

Table 9-26 summarizes configuration sharing of the Serial ID Capability registers among the different PCI functions.

Table 9-26. Configuration Sharing of the Serial Number Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
Serial Number Registers		x		

9.2.4.2.1 Device Serial Number Enhanced Capability Header Register (0x140; RO)

Bit(s)	Init.	Type	Description
15:0	0x3	RO	**PCIe Extended Capability ID** This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. The extended capability ID for the device serial number capability is 0x0003.
19:16	0x1	RO	**Capability Version** This field is a PCI-SIG defined version number that indicates the version of the capability structure present. ***Note:*** Must be set to 0x1 for this version of the specification.
31:20	See description	RO	**Next Capability Offset** This field contains the offset to the next PCIe capability structure or 0x000 if no other items exist in the linked list of capabilities. See Table 9-22 and Table 9-11 for possible values of the next capability offset.

9.2.4.2.2 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.

Bit(s)	Type	Description
63:0	RO	**PCIe Device Serial Number** This field contains the IEEE defined 64-bit EUI-64*. This identifier includes a 24-bit company ID value assigned by IEEE registration authority and a 40-bit extension identifier assigned by the manufacturer.

The serial number uses the Ethernet MAC Address according to the following definition:

Field	Company ID			Extension Identifier				
Order	Addr+0	Addr+1	Addr+2	Addr+3	Addr+4	Addr+5	Addr+6	Addr+7
Most Significant Byte						Least Significant Byte		
Most Significant Bit						Least Significant Bit		

The serial number can be constructed from the 48-bit Ethernet MAC Address in the following form:

Field	Company ID			MAC Label		Extension identifier		
Order	Addr+0	Addr+1	Addr+2	Addr+3	Addr+4	Addr+5	Addr+6	Addr+7
Most Significant Bytes						Least Significant Byte		
Most Significant Bit						Least Significant Bit		

In this case, the MAC label is 0xFFFF.

For example, assume that the company ID is (Intel) 00-A0-C9 and the extension identifier is 23-45-67. In this case, the 64-bit serial number is:

Field	Company ID			MAC Label		Extension Identifier		
Order	Addr+0	Addr+1	Addr+2	Addr+3	Addr+4	Addr+5	Addr+6	Addr+7
	00	A0	C9	FF	FF	23	45	67
Most Significant Byte						Least Significant Byte		
Most Significant Bit						Least Significant Bit		

The Ethernet MAC Address for the serial number capability is loaded from the NVM (not the same field that is loaded from NVM into the *RAL* and *RAH* registers). It is reflected in the *PCI_SERL* and *PCI_SERH* registers. The default value in case of no NVM is 0x0.

Note: The official document that defines EUI-64* is:
http://standards.ieee.org/regauth/oui/tutorials/EUI64.html

9.2.4.3 Alternate Routing ID Interpretation (ARI) Capability

To allow more than eight functions per endpoint without requesting an internal switch, as is usually needed in virtualization scenarios, the PCI-SIG defines a new capability that allows a different interpretation of the *Bus*, *Device*, and *Function* fields. The capability is exposed when the PCI_CAPSUP.*ARI_EN* bit is set from NVM.

The ARI capability structure is as follows:

Table 9-27. ARI Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x150	Next Capability Offset	Version (0x1)	ARI Capability ID (0x000E)	
0x154	ARI Control Register		ARI Capabilities	

Table 9-28 summarizes configuration sharing of the ARI Capability registers among the different PCI functions.

Table 9-28. Configuration Sharing of the ARI Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
ARI capability Register	Next Function Pointer		x	

9.2.4.3.1 PCIe ARI Header Register (0x150; RO)

Field	Bit(s)	Init.	Type	Description
ID	15:0	0xE	RO	**PCIe Extended Capability ID** PCIe extended capability ID for the alternative RID interpretation.
Version	19:16	0x1	RO	**Capability Version** This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 0x1 for this version of the specification.
Next Capability Offset	31:20	See description	RO	**Next Capability Offset** This field contains the offset to the next PCIe extended capability structure. See Table 9-22. and Table 9-11 for possible values of the next capability offset.

9.2.4.3.2 PCIe ARI Capabilities and Control Register (0x154; RO)

Field	Bit(s)	Init.	Type	Description
Reserved	7:0	0x2	RO	Reserved.
NFP	15:8	0x1 (func 0) 0x0 (func 1)[1]	RO	**Next Function Pointer** This field contains the pointer to the next physical function configuration space or 0x0000 if no other items exist in the linked list of functions. Function 0 is the start of the link list of functions.
Reserved	31:16	0x0	RO	Reserved.

1. Even if port 0 and port 1 are switched or function zero is a dummy function, this register should keep its attributes according to the function number. If LAN1 is disabled, the value of this field in function zero should be zero.

9.2.4.4 IOV Capability

Note: This capability structure is not exposed in a dummy function.

This is a structure used to support the SR-IOV capabilities reporting and control. The capability is exposed when the PCI_CAPSUP.*IOV_EN* bit is set from NVM and the PCI_CNF2.*NUM_VFS* is non zero.

Table 9-29. IOV Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x160	Next Capability Offset (0x0)	Version (0x1)	IOV Capability ID (0x0010)	
0x164	SR IOV Capabilities			
0x168	SR IOV Status		SR IOV Control	
0x16C	Total VFs (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)			

Table 9-29. IOV Capability Structure [continued]

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x18C	VF BAR2 (RO)			
0x190	VF BAR3- Low (RW) (64 bit)			
0x194	VF BAR3 High (RW) (64 bit)			
0x198	VF BAR5 (RO)			
0x19C	VF Migration State Array Offset (RO)			

Table 9-30 summarizes configuration sharing of the SR-IOV Capability registers among the different PCI functions.

Table 9-30. Configuration Sharing of the SR-IOV Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
SR-IOV Capabilities	VF Migration Capable	x		Not supported.
	ARI Capable Hierarchy Preserved		x	PF0 only. RO zero in all other functions.
	VF Migration Interrupt Message Number			Not supported.
SR-IOV Control	VF Enable		x	
	Memory Space Enable		x	
	ARI Capable Hierarchy	x		PF0 only. RO zero in all other functions.
InitialVFs			x	
TotalVFs			x	
NumVFs			x	
Function Dependency Link			x	Each PF indicates its PF number here.
First VF Offset			x	
VF Stride			x	
VF Device ID			x	
Supported Page Size		x		
System Page Size			x	
VF BARs			x	

9.2.4.4.1 PCIe SR-IOV Header Register (0x160; RO)

Field	Bit(s)	Init.	Type	Description
ID	15:0	0x10	RO	**PCIe Extended Capability ID** PCIe extended capability ID for the SR-IOV capability.
Version	19:16	0x1	RO	**Capability Version** This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 0x1 for this version of the specification.
Next offset	31:20	See description	RO	**Next Capability Offset** This field contains the offset to the next PCIe extended capability structure or 0x000 if no other items exist in the linked list of capabilities. See Table 9-22. and Table 9-11 for possible values of the next capability offset.

9.2.4.4.2 PCIe SR-IOV Capabilities Register (0x164; RO)

Field	Bit(s)	Init.	Type	Description
Reserved	0	0b	RO	Reserved.
ARICHP	1	1b/0b[1]	RO	**ARI Capable Hierarchy Preserved** If set, the ARI Capable Hierarchy bit is preserved across certain power state transitions.
Reserved	31:2	0x0	RO	Reserved.

1. Set on first function where SR-IOV is enabled (see PCI_CAPSUP.*IOV_EN* bit) and Read Only Zero in the other PF.

9.2.4.4.3 PCIe SR-IOV Control/Status Register (0x168; RW)

Field	Bit(s)	Init.	Type	Description
VFE	0	0b	RW	**VF Enable/Disable** VF Enable manages the assignment of VFs to the associated PF. If *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.
Reserved	2:1	00b	RO	Reserved.
VF MSE	3	0b	RW	**Memory Space Enable for Virtual Functions** VF MSE controls memory space enable for all VFs associated with this PF as with the *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 must follow the same error reporting rules as defined in the base specification if an attempt is made to access a virtual functions memory space when VF Enable is 1 and VF MSE is zero. Implementation Note: Virtual functions memory space cannot be accessed when VF Enable is zero. Thus, VF MSE is “don't care” when VF Enable is zero, however, software may choose to set VF MSE after programming the VF BARn registers, prior to setting VF Enable to 1.

Field	Bit(s)	Init.	Type	Description
ARI Capable Hierarchy	4	0b	RW (first function where SR-IOV is enabled) ROS (other function)[1]	**ARI Capable Hierarchy** The X550 is permitted to locate VFs in function numbers 8 to 255 of the captured bus number. This field is R/W in the lowest numbered PF. Other functions use the PF0 value as sticky. ***Note:*** If either ARI Capable Hierarchy Preserved is set (see Section 9.2.4.4.2) or *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	31:5	0x0	RO	Reserved.

1. Even if port 0 and port 1 are switched or function zero is a dummy function, this field should keep its attributes according to the function number.

9.2.4.4.4 PCIe SR-IOV Max/Total VFs Register (0x16C; RO)

Field	Bit(s)	Init.	Type	Description
InitialVFs	15:0	64	RO	**InitialVFs** Indicates the number of VFs that are initially associated with the PF. If *VF Migration Capable* is cleared, this field must contain the same value as TotalVFs. In the X550 this parameter is equal to the TotalVFs in this register.
TotalVFs	31:16	64	RO	**TotalVFs** Defines the maximum number of VFs that can be associated with the PF. This field is loaded from the Max VFs field in the NVM. Reflected in PCI_CNF2.*TOTAL_VFS*.

9.2.4.4.5 PCIe SR-IOV Num VFs Register (0x170; RW)

Field	Bit(s)	Init.	Type	Description
NumVFs	15:0	0x0	RW	**Num VFs** Defines the number of VFs software has assigned to the PF. Software sets NumVFs to any value between one and the TotalVFs as part of the process of creating VFs. NumVFs VFs must be visible in the PCIe fabric after both NumVFs is set to a valid value *VF Enable* is set to 1b.
FDL	23:16	0x0 (func 0) 0x1 (func 1)[1]	RO	**Function Dependency Link** Defines dependencies between physical functions allocation. In the X550 there are no constraints.
Reserved	31:24	0x0	RO	Reserved.

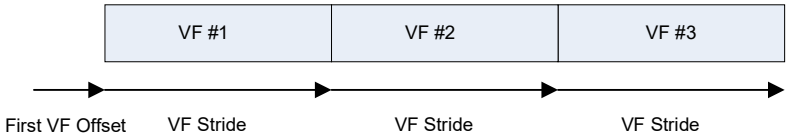
1. Even if port 0 and port 1 are switched or function zero is a dummy function, this register should keep it's attributes according to the function number.

9.2.4.4.6 PCIe SR-IOV VF RID Mapping Register (0x174; RO)

Field	Bit(s)	Init.	Type	Description
FVO	15:0	0x180	RO	**First VF offset** Defines the requestor ID (RID) offset of the first VF that is associated with the PF that contains this capability structure. The first VFs 16-bit RID is calculated by adding the contents of this field to the RID of the PF containing this field. The content of this field is valid only when *VF Enable* is set. If *VF Enable* is 0b, the contents are undefined. If the *ARI Enable* bit is set, this field changes to 0x80.

Field	Bit(s)	Init.	Type	Description
VFS	31:16	0x2[1]	RO	**VF stride** Defines the requestor ID (RID) offset from one VF to the next one for all VFs associated with the PF that contains this capability structure. The next VFs 16-bit RID is calculated by adding the contents of this field to the RID of the current VF. The contents of this field is valid only when *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 7.7.9.6.1.



9.2.4.4.7 PCIe SR-IOV VF Device ID Register (0x178; RO)

Field	Bit(s)	Init.	Type	Description
DEVID	31:16	0x1565	RO	**VF Device ID** This field contains the device ID that should be presented for every VF to the Virtual Machine (VM). The value of this field can be read from the *IOV Control Word 2* in the NVM.
Reserved	15:0	0x0	RO	Reserved.

9.2.4.4.8 PCIe SR-IOV Supported Page Size Register (0x17C; RO)

Field	Bit(s)	Init.	Type	Description
Supported Page Size	31:0	0x553	RO	**Supported Page Size** For PFs that supports the stride-based BAR mechanism, this field defines the supported page sizes. This PF supports a page size of 2^(n+12) if bit 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.

9.2.4.4.9 PCIe SR-IOV System Page Size Register (0x180; RW)

Field	Bit(s)	Init.	Type	Description
Page Size	31:0	0x1	RW	**Page Size** This field defines the page size the system uses to map the PF's and associated VFs' memory addresses. Software must set the value of the *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 PF and its associated VFs are required to support a page size of 2^(n+12). For example, if bit 1 is set, the system is using an 8 KB page size. The results are undefined if more than one bit is set in *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 PF and associated 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.

9.2.4.4.10 PCIe SR-IOV BAR 0 — Low Register (0x184; RW)

Field	Bit(s)	Init.	Type	Description
Mem	0	0b	RO	**Memory** 0b = Indicates memory space.
Mem Type	2:1	10b	RO	**Memory Type** Indicates the address space size. 10b = 64-bit. This bit is loaded from the NVM. It is reflected in the PCI_VFSUP register.
Prefetch Mem	3	0b*	RO	**Prefetch Memory** 0b = Non-prefetchable space. 1b = Prefetchable space. This bit is loaded from the NVM. It is reflected in the PCI_VFSUP register.
Memory Address Space	31:4	0x0	RW	**Memory Address Space** Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size. The size is a maximum between 16 KB and the page size.

9.2.4.4.11 PCIe SR-IOV BAR 0 — High Register (0x188; RW)

Field	Bit(s)	Init.	Type	Description
BAR0 — MSB	31:0	0x0	RW	MSB part of BAR0.

9.2.4.4.12 PCIe SR-IOV BAR 2 Register (0x18C; RO)

Field	Bit(s)	Init.	Type	Description
BAR2	31:0	0x0	RO	This BAR is not used.

9.2.4.4.13 PCIe SR-IOV BAR 3 — Low Register (0x190; RW)

Field	Bit(s)	Init.	Type	Description
Mem	0	0b	RO	**Memory** 0b = Indicates memory space.
Mem Type	2:1	10b	RO	**Memory Type** Indicates the address space size. 10b = 64-bit. This bit is loaded from the NVM. It is reflected in the PCI_VFSUP register.
Prefetch Mem	3	0b*	RO	**Prefetch Memory** 0b = Non-prefetchable space. 1b = Prefetchable space. This bit is loaded from the NVM. It is reflected in the PCI_VFSUP register.
Memory Address Space	31:4	0x0	RW	**Memory Address Space** Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size. The size is a maximum between 16 KB and the page size.

9.2.4.4.14 PCIe SR-IOV BAR 3 — High Register (0x194; RW)

Field	Bit(s)	Init.	Type	Description
BAR4 — MSB	31:0	0x0	RW	MSB part of BAR3.

9.2.4.4.15 PCIe SR-IOV BAR 5 Register (0x198; RO)

Field	Bit(s)	Init.	Type	Description
BAR5	31:0	0x0	RO	This BAR is not used

9.2.4.4.16 PCIe SR-IOV VF Migration State Array Offset Register (0x19C; RO)

Field	Bit(s)	Init.	Type	Description
Reserved	31:0	0x0	RO	Reserved.

9.2.4.5 TPH Requester Capability

The TPH Requester capability is an optional extended capability to support TLP Processing Hints. Table 9-31 lists the TPH extended capability structure for PCIe devices.

Table 9-31. TPH Requester Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1A0	Next Capability Offset	Version (0x1)	TPH Capability ID (0x0017)	
0x1A4	TPH Requester Capability Register			
0x1A8	TPH Requester Control Register			

Table 9.2.4.5.3 summarizes configuration sharing of the TPH Requester Capability registers among the different PCI functions.

Table 9-32. Configuration Sharing of the TPH Requester Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
TPH Requester Capability		x		
TPH Requester Control			x	
TPH ST Table			x	The Steering Table Upper fields are not supported

9.2.4.5.1 TPH Requester Extended Capability Header (0x1A0; RO)

Bit(s)	Init.	Type	Description
15:0	0x17	RO	**Extended Capability ID** PCIe extended capability ID indicating TPH capability.
19:16	0x1	RO	**Capability Version** PCIe TPH extended capability version number.
31:20	See description	RO	**Next Capability Offset** This field contains the offset to the next PCIe capability structure. See Table 9-22. and Table 9-11 for possible values of the next capability offset.

9.2.4.5.2 TPH Requester Capability Register (0x1A4; RO)

Bit(s)	Init.	Type	Description
0	1b	RO	**No ST Mode Supported** If set indicates that the Function supports the No ST Mode of operation
1	0b	RO	**Interrupt Vector Mode Supported** Cleared to indicates that the X550 does not support Interrupt Vector Mode of operation
2	1b	RO	**Device Specific Mode** Set to indicate that the X550 supports Device Specific Mode of operation
7:3	0x0	RO	Reserved.
8	0b	RO	**Extended TPH Requester Supported** Cleared to indicate that the function is not capable of generating requests with Extended TPH TLP Prefix
10:9	00b	RO	**ST Table Location** Value indicates if and where the ST Table is located. Defined Encodings are: 00b = ST Table is not present. 01b = ST Table is located in the TPH Requester Capability structure. 10b = ST Table is located in the MSI-X Table structure. 11b = Reserved ST Table is not supported.
15:11	0x0	RO	Reserved.
26:16	0x0	RO	**ST_Table Size** System software reads this field to determine the ST_Table_Size N, which is encoded as N-1. For example, a returned value of 00000000011b indicates a table size of 4. The value in this field is undefined since the X550 does not support an ST Table
31:27	0x0	RO	Reserved.

9.2.4.5.3 TPH Requester Control Register (0x1A8; R/W)

Bit(s)	Init.	Type	Description
2:0	000b	RW	**ST Mode Select** Indicates the ST mode of operation selected. Defined encodings are: 000b = No Table Mode 001b = Interrupt Vector Mode (not supported by the X550) 010b = Device Specific Mode All other values are reserved. The default value of 000 indicates No Table mode of operation. Functions that support only the No ST Mode of operation must hard-wire this field to 000b.
7:3	0x0	RO	Reserved.
9:8	00b	RW	**TPH Requester Enable** Controls the ability to issue Request TLPs using either TPH or Extended TPH. Defined Encodings are: 00b = The X550 is not permitted to issue transactions with TPH or Extended TPH as Requester 01b = The X550 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 X550 is permitted to issue transactions with TPH and Extended TPH as Requester (The X550 does not issue transactions with Extended TPH). The default value of this field is 00b.
31:10	0x0	RO	Reserved.

9.2.4.6 Access Control Services (ACS) Capability

The PCIe ACS defines a set of control points within a PCIe topology to determine whether a TLP should be routed normally, blocked, or redirected. ACS is applicable to RCs, switches, and multifunction devices and is not exposed in Single function mode (when function 1 is disabled). The capability is exposed when the PCI_CAPSUP.*ACS_EN* bit is set from NVM.

The ACS Capability structure is shared and exposed to all PFs.

Table 9-33 lists the PCIe ACS extended capability structure for PCIe devices.

Table 9-33. ACS Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1B0	Next Capability Offset	Version (0x1)	ACS Capability ID (0x0D)	
0x1B4	ACS Control Register (0x0)		ACS Capability Register (0x0)	

9.2.4.6.1 ACS CAP ID (0x1B0; RO)

Bit(s)	Init.	Type	Description
15:0	0x0D	RO	**ACS Capability ID** PCIe extended capability ID indicating ACS capability.
19:16	0x1	RO	**Version Number** PCIe ACS extended capability version number.
31:20	See description	RO	**Next Capability Pointer** This is the last capability, so the next pointer is 0x000. See Table 9-22. and Table 9-11 for possible values of the next capability offset.

9.2.4.6.2 ACS Control and Capabilities (0x1B4; RO)

Bit(s)	Init.	Type	Description
0	0b	RO	**ACS Source Validation (V)** Hard-wired to zero, not supported in the X550.
1	0b	RO	**ACS Translation Blocking (B)** Hard-wired to zero, not supported in the X550.
2	0b	RO	**ACS P2P Request Redirect (R)** Hard-wired to zero, not supported in the X550.
3	0b	RO	**ACS P2P Completion Redirect (C)** Hard-wired to zero, not supported in the X550.
4	0b	RO	**ACS Upstream Forwarding (U)** Hard-wired to zero, not supported in the X550.
5	0b	RO	**ACS P2P Egress Control (E)** Hard-wired to zero, not supported in the X550.
6	0b	RO	**ACS Direct Translated P2P (T)** Hard-wired to zero, not supported in the X550.
7	0b	Rsrv	Reserved.
15:8	0x0	RO	**Egress Control Vector Size** Hard-wired to zero, not supported in the X550.
16	0b	RO	**ACS Source Validation Enable (V)** Hard-wired to zero, not supported in the X550.
17	0b	RO	**ACS Translation Blocking Enable (B)** Hard-wired to zero, not supported in the X550.
18	0b	RO	**ACS P2P Request Redirect Enable (R)** Hard-wired to zero, not supported in the X550.
19	0b	RO	**ACS P2P Completion Redirect Enable (C)** Hard-wired to zero, not supported in the X550.
20	0b	RO	**ACS Upstream Forwarding Enable (U)** Hard-wired to zero, not supported in the X550.
21	0b	RO	**ACS P2P Egress Control Enable (E)** Hard-wired to zero, not supported in the X550.
22	0b	RO	**ACS Direct Translated P2P Enable (T)** Hard-wired to zero, not supported in the X550.
31:23	0x0	Rsrv	Reserved.

9.2.4.7 Latency Tolerance Reporting (LTR) Capability

The Latency Tolerance Reporting Capability is an optional Extended Capability that allows software to provide platform latency information to devices with upstream ports (Endpoints and Switches). This capability structure is required if the device supports Latency Tolerance Reporting (LTR). The capability is exposed when the PCI_CAPSUP.*LTR_EN* bit is set from NVM.

The LTR Capability structure is implemented only in Function 0 even when Function 0 is a dummy function, and controls the component's Link behavior on behalf of all the Functions of the device

Table 9-34 lists the PCIe LTR extended capability structure for PCIe devices.

Table 9-34. LTR Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1C0	PCI Express Extended Capability Header			
0x1C4	Max No-Snoop Latency Register		Max Snoop Latency Register	

9.2.4.7.1 LTR Extended Capability Header (0x1C0; RO)

Bit(s)	Init.	Type	Description
15:0	0x18	RO	**PCI Express Extended Capability ID** PCIe extended capability ID indicating LTR capability.
19:16	0x1	RO	**Capability Version** PCIe LTR extended capability version number.
31:20	See description	RO	**Next Capability Offset** See Table 9-22 and Table 9-11 for possible values of the next capability offset.

9.2.4.7.2 Max Snoop Latency Register (0x1C4; RW)

Bit(s)	Init.	Type	Description
9:0	0x0	RW	**Maximum Snoop Latency Value** Along with the *Max Snoop Latency Scale field*, this register specifies the maximum snoop latency that a device is permitted to request. Software should set this to the platform's maximum supported latency or less. Field is also an indicator of the platform maximum latency, should an endpoint send up LTR Latency Values with the *Requirement* bit not set.
12:10	000b	RW	**Max Snoop Latency Scale** This field provides a scale for the value contained within the *Maximum Snoop Latency Value* field. Encoding: 000b = Value times 1 ns 001b = Value times 32 ns 010b = Value times 1,024 ns 011b = Value times 32,768 ns 100b = Value times 1,048,576 ns 101b = Value times 33,554,432 ns 110b = Not permitted 111b = Not permitted
15:13	0x0	RW	Reserved.

9.2.4.7.3 Max No-Snoop Latency Register (0x1C6; RW)

Bit(s)	Init.	Type	Description
9:0	0x0	RW	**Maximum No-Snoop Latency Value** Along with the *Max No-Snoop Latency Scale* field, this register specifies the maximum no-snoop latency that a device is permitted to request. Software should set this to the platform's maximum supported latency or less. Field is also an indicator of the platforms maximum latency, should an endpoint send up LTR Latency Values with the *Requirement* bit not set.
12:10	000b	RW	**Max No-Snoop Latency Scale** This field provides a scale for the value contained within the *Maximum No-Snoop Latency Value* field. Encoding: 000b = Value times 1 ns 001b = Value times 32 ns 010b = Value times 1,024 ns 011b = Value times 32,768 ns 100b = Value times 1,048,576 ns 110b = Not permitted 111b = Not permitted
15:13	0x0	RW	Reserved.

9.2.4.8 Secondary PCI Express Extended Capability

The Secondary PCI Express Extended Capability structure is required for all Ports and RCRBs that support a Link speed of 8.0 GT/s or higher. For Multi-Function Upstream Ports, this capability must be implemented in Function 0 and must not be implemented in other Functions. The capability is exposed when the PCI_CAPSUP.*SEC_EN* bit is set from NVM.

Table 9-35 lists the Secondary PCI Express extended capability structure for PCIe devices.

Table 9-35. Secondary PCI Express Extended Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1D0	PCI Express Extended Capability Header			
0x1D4	Link Control 3 Register			
0x1D8	Lane Error Status Register			
0x1DC	Equalization Control Register (Sized by Maximum Link Width)			
0x1E0	Equalization Control Register (Sized by Maximum Link Width)			

9.2.4.8.1 Secondary PCIe Extended Capability Header (0x1D0)

Bit(s)	Init.	Type	Description
15:0	0x19	RO	**PCI Express Extended Capability ID** This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Secondary PCI Express Extended Capability is 0019h.
19:16	0x1	RO	**Capability Version** This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.

Bit(s)	Init.	Type	Description
31:20	See description	RO	**Next Capability Offset** See Table 9-22. and Table 9-11 for possible values of the next capability offset.

9.2.4.8.2 Link Control 3 Register (0x1D4)

Bit(s)	Init.	Type	Description
0	0b	RW/RsvdP	**Perform Equalization** When this bit is 1b and a 1b is written to the Link Retrain register with Target Link Speed set to 8.0 GT/s, the Downstream Port must perform Transmitter Equalization. This bit is RW for Upstream Ports when *Crosslink Supported* is 1b. This bit is not applicable and is RsvdP for Upstream Ports when the *Crosslink Supported* bit is 0b. The default value is 0b.
1	0b	RW/RsvdP	**Link Equalization Request Interrupt Enable** When Set, this bit enables the generation of interrupt to indicate that the *Link Equalization Request* bit has been set. This bit is RW for Upstream Ports when *Crosslink Supported* is 1b. This bit is not applicable and is RsvdP for Upstream Ports when the *Crosslink Supported bit* is 0b. The default value for this bit is 0b.
31:2	0x0	RsvdZ	Reserved.

9.2.4.8.3 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(s)	Init.	Type	Description
7:0	0x0	RW1CS	**Lane Error Status Bits** Each bit indicates if the corresponding Lane detected a Lane-based error. A value of 1b indicates that a Lane based-error was detected on the corresponding Lane Number. The default value of this field is 0b. This field is intended for debug purposes only.
31:8	0x0	RsvdZ	Reserved.

9.2.4.8.4 Lane Equalization Control Register (0x1DC: 0x1E3)

The Equalization Control register consists of control fields required for per Lane equalization and number of entries in this register are sized by Maximum Link Width.

15	0
Lane (0) Equalization Control Register	
Lane (1) Equalization Control Register	
...	
Lane (Maximum Link Width - 1) Equalization Control Register	

Lane ((Maximum Link Width – 1):0) Equalization Control Register:

Bit(s)	Init.	Type	Description
3:0	0x0	RsvdP	**Downstream Port Transmitter Preset** For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP.
6:4	000b	RsvdP	**Downstream Port Receiver Preset Hint** For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP.
11:8	0xF	RO	**Upstream Port Transmitter Preset** Field contains the Transmit Preset value sent or received during Link Equalization. Since crosslink is not supported, the field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization. Field is RO. The default value is 1111b.
14:12	111b	HwInit/RO	**Upstream Port Receiver Preset Hint** Field contains the Receiver Preset Hint value sent or received during Link Equalization. Field usage varies as follows: Since crosslink is not supported, the field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization. Field is RO. The default value is 111b.
15	0b	Rsvd	Reserved.

9.2.5 Driver Forward Compatibility Register (0x94; RO)

This register is an advisory register that returns a fixed value indicating the type of software device driver class needed for this device.

Bit(s)	Init.	Type	Description
15:0	0xFFFF	RO	**Pointer to Admin Queues** A value of 0xFFFF indicates that this device does not support admin. queues.
19:16	0x0	RO	**Status** Not implemented in the X550.
31:20	0xA0A	RO	**Signature** Signature indicating a driver class register.

9.2.6 CSR Access Via Configuration Address Space

The registers described in the section are not part of the PCI Express standard configuration and can be used to access the CSR space before memory BARs are allocated. When this mechanism is used, there is no need to expose an I/O BAR for pre boot operation.

Note: These registers are not available for dummy functions.

9.2.6.1 IOADDR Register (0x98; R/W)

This is a read/write register. Each function has its own IOADDR register. Functionality is the same in all functions. Register is cleared at power-up (LAN_PWR_GOOD) or PCIe reset.

Note: When functioning in a D3 state, software should not attempt to access CSRs via the IOADDR and IODATA registers.

Bit(s)	Init.	Type	Description
30:0	0x0	R/W[1]	**Internal Register or Internal Memory location Address** 0x00000-0x1FFFF = Internal registers and memories 0x20000-0x7FFFFFFF = Undefined
31	0b	R/W	**Configuration IO Access Enable** 0b = CSR configuration read or write disabled. 1b = CSR configuration read or write enabled. When this bit is set, accesses to the IODATA register actually generate transactions to the X550. Otherwise, accesses to the IODATA register are don't-cares (writes are discarded silently, reads return arbitrary results).

1. In the event that the PCI_CAPSUP.*CSR_CONF_EN* bit is cleared, accesses to the IOADDR register via the configuration address space is ignored and has no effect on the register and the CSRs referenced by the IOADDR register.

9.2.6.2 IODATA Register (0x9C; R/W)

This is a read/write register. Each function has its own IODATA register. Functionality is the same in all functions. Register is cleared at power-up (LAN_PWR_GOOD) or PCIe reset.

Bit(s)	Init.	Type	Description
31:0	0x0	R/W[1]	Data field for reads or writes to the Internal register or internal memory location as identified by the current value in IOADDR. All 32 bits of this register can be read from or written to.

1. In the event that the *IO_by_cfg* bit in the *PCIe Init Configuration 2* EEPROM word is cleared, access to the IODATA register via the configuration address space is ignored and has no effect on the register and the CSRs referenced by the IOADDR register.

9.3 Virtual Functions Configuration Space

The configuration space reflected to each of the VF is a sparse version of the physical function configuration space. Table 9-36 describes the behavior of each register in the VF configuration space.

Table 9-36. VF PCIe Configuration Space

Section	Offset	Name	VF behavior	Notes
PCI Mandatory Registers	0	Vendor ID	RO — 0xFFFF	
	2	Device ID	RO — 0xFFFF	
	4	Command	Per VF	See Section 9.3.1.1.
	6	Status	Per VF	See Section 9.3.1.2.
	8	RevisionID	RO as PF	
	9	Class Code	RO as PF	
	C	Cache Line Size	RO — 0x0	
	D	Latency Timer	RO — 0x0	
	E	Header Type	RO — 0x0	
	F	Reserved	RO — 0x0	
	10 — 27	BARs	RO — 0x0	Emulated by VMM.
	28	CardBus CIS	RO — 0x0	Not used.
	2C	Sub Vendor ID	RO as PF	
	2E	Sub System	RO as PF	
	30	Expansion ROM	RO — 0x0	Emulated by VMM.
	34	Cap Pointer	RO — 0x70	Next = MSI-X capability.
	3C	Int Line	RO — 0x0	
	3D	Int Pin	RO — 0x0	
	3E	Max Lat/Min Gnt	RO — 0x0	
MSI-X Capability	70	MSI-X Header	RO — 0xA011	Next = PCIe capability.
	72	MSI-x Message Control	Per VF	See Section 9.3.2.1.1.
	74	MSI-X table Address	RO — as PF	See Section 9.3.2.1.2
	78	MSI-X PBA Address	RO	See Section 9.3.2.1.3

Table 9-36. VF PCIe Configuration Space [continued]

Section	Offset	Name	VF behavior	Notes
PCIe Capability	A0	PCIe Header	RO — 0x0010	Next = Last capability.
	A2	PCIe Capabilities	RO — as PF	
	A4	PCIe Dev Cap	RO — as PF	
	A8	PCIe Dev Ctrl	RW	As PF apart from FLR — See Section 9.3.2.2.1.
	AA	PCIe Dev Status	Per VF	See Section 9.3.2.2.2.
	AC	PCIe Link Cap	RO — as PF	
	B0	PCIe Link Ctrl	RO — 0x0	
	B2	PCIe Link Status	RO — 0x0	
	C4	PCIe Dev Cap 2	RO — as PF	
	C8	PCIe Dev Ctrl 2	RO — 0x0	
	D0	PCIe Link Ctrl 2	RO — 0x0	
	D2	PCIe Link Status 2	RO — 0x0	
AER Capability	100	AER — Header	RO — 0x15010001	Next = ARI structure.
	104	AER — Uncorr Status	Per VF	See Section 9.3.3.1.1.
	108	AER — Uncorr Mask	RO — 0x0	
	10C	AER — Uncorr Severity	RO — 0x0	
	110	AER — Corr Status	Per VF	See Section 9.3.3.1.2.
	114	AER — Corr Mask	RO — 0x0	
	118	AER — Cap/Ctrl	RO as PF	
	11C — 128	AER — Error Log	Shared two logs for all VFs	Same structure as in PF. In case of overflow, the header log is filled with ones.
ARI Capability	150	ARI — Header	0x1A01000E	Next = TPH.
	154	ARI — Cap/Ctrl	RO — 0X0	
TPH Capability	1A0	TPH - Header	RO - 0x1B010017	Next = ACS.
	1A4	TPH - Capability	RO - 0x00000005	No ST Table.
	1A8	TPH - Control	Per VF	Same structure as PF.
ACS capability	1B0	ACS - Header	RO - 0x0001000D	Last extended capability.
	1B4	ACS - Capability	RO - 0x00000000	

9.3.1 Mandatory Configuration Space

9.3.1.1 VF Command Register (0x4; RW)

Bit(s)	Init.	Type	Description
0	0b	RO	**IOAE** I/O Access Enable. RO as zero field.
1	0b	RO	**MAE** Memory Access Enable. RO as zero field.
2	0b	RW	**BME** Bus Master Enable. Disabling this bit prevents the associated VF from issuing any memory or I/O requests. Note that as MSI/MSI-X interrupt messages are in-band memory writes, disabling the bus master enable bit disables MSI/MSI-X interrupt messages as well. Requests other than memory or I/O requests are not controlled by this bit. ***Note:*** The state of active transactions is not specified when this bit is disabled after being enabled. The device can choose how it behaves when this condition occurs. Software cannot count on the device retaining state and resuming without loss of data when the bit is re-enabled. Transactions for a VF that has its *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	0x0	RO	Reserved.

9.3.1.2 VF Status Register (0x6; RW)

Bit(s)	Init.	Type	Description
2:0	0x0	RO	Reserved.
3	0b	RO	**IS.** Interrupt Status. Hard-wired to 0b.
4	1b	RO	**NC** New Capabilities. Indicates that the X550 VFs implement extended capabilities. X550 VFs implement a capabilities list, to indicate that it supports MSI-X and PCIe extensions.
5	0b	RO	**66E** 66 MHz Capable. Hard-wired to 0b.
6	0b	RO	Reserved.

Bit(s)	Init.	Type	Description
7	0b	RO	**FB2BC** Fast Back-to-Back Capable. Hard-wired to 0b.
8	0b	RW1C	**MPERR** Data Parity Reported.
10:9	00b	RO	**DEVSEL** DEVSEL Timing. Hard-wired to 0b.
11	0b	RW1C	**STA** Signaled Target Abort.
12	0b	RW1C	**RTA** Received Target Abort.
13	0b	RW1C	**RMA** Received Master Abort.
14	0b	RW1C	**SSERR** Signaled System Error.
15	0b	RW1C	**DSERR** Detected Parity Error.

9.3.2 PCI Capabilities

9.3.2.1 MSI-X Capability

The only registers with a different layout than the PF for MSI-X, is the control register.

Note: The message address and data registers in enhanced mode use the first MSI-X entry of each VF in the regular MSI-X table.

See Section 8.3.4.1 for details of the MSI-X registers in BAR 3 of the VF.

9.3.2.1.1 VF MSI-X Control Register (0x72; RW)

Bit(s)	Init.	Type	Description
10:0	0x2[1]	RO	**TS** Table Size.
13:11	000b	RO	Reserved.
14	0b	RW	**Mask** Function Mask.
15	0b	RW	**En** MSI-X Enable.

1. Default value is read from the NVM. This field is loaded from the PCI_CNF2.*MSI_X_VF_N* register field.

9.3.2.1.2 MSI-X Table Offset Register (0x74; RW)

Bit(s)	Init.	Type	Description
2:0	011b	RO	**Table BIR** Indicates which one of a function's BARs, beginning at 0x10 in the configuration space, is used to map the function's MSI-X table into the memory space. BIR values: 0...5 correspond to BARs 0x10...0x24, respectively. BIR equals 3 to indicate BAR 3 and 4 are used for MSI-X vectors.
31:3	0x0	RO	**Table Offset** Used as an offset from the address contained in one of the function's BARs to point to the base of the MSI-X table. The lower three *Table BIR* bits are masked off (set to 0b) by software to form a 32-bit QWord-aligned offset. This field is read only.

9.3.2.1.3 MSI-X PBA Register (0x78; RO)

Bit(s)	Init.	Type	Description
2:0	011b	RO	**PBA BIR** Indicates which one of a function's BARs, located beginning at 0x10 in configuration space, is used to map the function's MSI-X PBA into memory space. BIR values: 0...5 correspond to BARs 0x10...0x24, respectively. BIR equals 3 to indicate BAR 3 and 4 are used for MSI-X PBA.
31:3	0x400	RO	**PBA Offset** Used as an offset from the address contained by one of the function's BARs to point to the base of the MSI-X PBA. The lower three *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 value programmed to the IOV System Page Size register.

9.3.2.2 PCIe Capability Registers

The Device Control and Device Status registers have some fields which are specific per VF.

9.3.2.2.1 VF Device Control Register (0xA8; RW)

Bit(s)	Init.	Type	Description
0	0b	RO	**Correctable Error Reporting Enable** Zero for VFs.
1	0b	RO	**Non-Fatal Error Reporting Enable** Zero for VFs.
2	0b	RO	**Fatal Error Reporting Enable** Zero for VFs.
3	0b	RO	**Unsupported Request Reporting Enable** Zero for VFs.
4	0b	RO	**Enable Relaxed Ordering** Zero for VFs.
7:5	000b	RO	**Max Payload Size** Zero for VFs.
9:8	00b	RO	Reserved.
10	0b	RO	**Auxiliary Power PM Enable** Zero for VFs.
11	0b	RO	**Enable No Snoop** Zero for VFs.

Bit(s)	Init.	Type	Description
14:12	000b	RO	**Max Read Request Size** Zero for VFs.
15	0b	RW	**Initiate Function Level Reset** Specific to each VF.

9.3.2.2.2 VF Device Status Register (0xAA; RW1C)

Bit(s)	Init.	Type	Description
0	0b	RW1C	**Correctable Detected** Indicates status of correctable error detection. Zero for VF.
1	0b	RW1C	**Non-Fatal Error Detected** Indicates status of non-fatal error detection. Zero for VF.
2	0b	RW1C	**Fatal Error Detected** Indicates status of fatal error detection. Zero for VF.
3	0b	RW1C	**Unsupported Request Detected** Indicates that the X550 received an unsupported request. This field is identical in all functions. The X550 cannot distinguish which function caused an error. Zero for VF.
4	0b	RO	**Aux Power Detected** Zero for VFs.
5	0b	RO	**Transaction Pending** Specific per VF. When set, indicates that a particular function (PF or VF) has issued non-posted requests that have not been completed. A function reports this bit cleared only when all completions for any outstanding non-posted requests have been received.
15:6	0x0	RO	Reserved.

9.3.3 PCIe Extended Capabilities

9.3.3.1 AER Registers

The following registers in the AER capability have a different behavior in a VF function.

Unlike the PF AER registers, these registers are not sticky since the VF is reset on FLR and on in-band reset.

9.3.3.1.1 Uncorrectable Error Status Register (0x104; RW1C)

Bit(s)	Init.	Type	Description
3:0	0x0	RO	Reserved.
4	0b	RO	**Data Link Protocol Error Status**
5	0b	RO	**Surprise Down Error Status (Optional)**
11:6	0x0	RO	Reserved.
12	0b	RW1C	**Poisoned TLP Status**
13	0b	RO	**Flow Control Protocol Error Status**
14	0b	RW1C	**Completion Timeout Status**
15	0b	RW1C	**Completer Abort Status**
16	0b	RW1C	**Unexpected Completion Status**
17	0b	RO	**Receiver Overflow Status**
18	0b	RO	**Malformed TLP Status**
19	0b	RO	**ECRC Error Status**
20	0b	RW1C	**Unsupported Request Error Status** When caused by a function that claims a TLP.
21	0b	RO	**ACS Violation Status**
31:22	0x0	RO	Reserved.

9.3.3.1.2 Correctable Error Status Register (0x110; RW1C)

The Correctable Error Status register reports error status of individual correctable error sources on a PCIe device. When an individual error status bit is set to 1b it indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit.

Bit(s)	Init.	Type	Description
0	0b	RW1C	**Receiver Error Status**
5:1	0x0	RO	Reserved.
6	0b	RW1C	**Bad TLP Status**
7	0b	RW1C	**Bad DLLP Status**
8	0b	RW1C	**REPLAY_NUM Rollover Status**
11:9	000b	RO	Reserved.
12	0b	RW1C	**Replay Timer Timeout Status**
13	0b	RW1C	**Advisory Non-Fatal Error Status**
31:14	0x0	RO	Reserved.

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

Chapter 10 PHY Registers

10.1 Introduction

This chapter describes the PHY registers in the X550, which are accessible via the internal MDIO interface. Refer to Section 3.7.2.

The PHY is internally divided into a series of MDIO Manageable Devices (MMDs), each of which performs a logical function as per the 10GBASE-T standard:

- MMD #1 contains the PMA, which is basically the analog front-end of the chip (Section 10.2).
- MMD #3 contains the PCS, which handles the 10GBASE-T transmission frame coding and decoding, including the 128-DSQ and LDPC work (Section 10.3).
- MMD #7 contains the auto-negotiation function (Section 10.4).
- MMD #29 contains the controls for the GbE and 100 Mb/s PCS machinery (Section 10.5).
- MMD# 30 contains the global control functionality for the PHY, including the embedded micro controller (Section 10.6).

10.1.1 PHY Register Structure

A map of these regions is shown in Table 10-1.

Table 10-1. MMD Device Addresses

5-bit Device Address (Hex)	MMD Name
0	Reserved
1	PMA/PMD (128 DSQ)
2	Reserved
3	PCS (64/65B coder/decoder)
4	Reserved
5 → 6	Reserved
7	Auto-negotiation
8 → 1C	Reserved
1D	Clause 22 Extension
1E	Global
1F	Reserved

Any attempt to read from the reserved MMD addresses returns a value of 0x00, and any writes to these addresses have no effect.

10.1.2 Format and Nomenclature

PHY registers within the device are referenced in the format:

Region.Register.Bit

where ***Region*** corresponds to the MMD region being addressed, ***Register*** corresponds to the register within the MMD region, and ***Bit*** is the bit within the register. All registers within the MDIO register space are 16 bits. The address of the register is the 16 bit MDIO address.

All read and write operation are word based, which means that the entire 16 bit register is read or written (versus individual bits). Within the MDIO register space, there are several different bit types. A list of these bit types are listed in Table 10-2.

Table 10-2. Field Types within the MDIO Register Space

Type	Description
LL	**Latching Low** If the condition the bit is monitoring goes low, this bit latches low, generates a maskable interrupt, and stays low until read. Reading this bit resets it to one. This bit is read-only.
LH	**Latching High** If the condition the bit is monitoring goes high, this bit latches high, generates a maskable interrupt, and stays high until read. Reading this bit resets it to zero. This bit is read-only.
LRF	**Latch Rising or Falling** Set high on either a rising or falling edge. If a transition occurs, this bit latches high, generates a maskable interrupt, and stays high until read. Reading this bit resets it to zero. This bit is read-only.
PD	**Provisionable Defaults** Indicates that the default value associated with this field is provisionable.
RO	**Read-Only** Read-only field. Writes are ignored.
ROS	**Read Only Static** Read-only static field. The same value is always returned. Writes are ignored.
RSV	**Reserved** Reserved for future use.
RW	**Read/Write** Field can be both read from and written to.
SC	**Self-Clearing** A read/write register which resets itself upon completion of an action.
SCT	**Saturating Counter** A read-only counter that saturates at the limit, and is cleared on read.
SCTL	**Saturating Counter LSW** The Least Significant Word of a Saturating Counter. This register clears the pair to zero on read and snapshots the mate MSW to shadow memory, awaiting read.
SCTM	**Saturating Counter MSW** The Most Significant Word of a Saturating Counter. Reading this completes the read process of the register pair.

10.1.3 Structure

The following structure is used for registers:

1. All Clause 45 registers (registers defined in Clause 45) are placed in their respective areas within the MMDs as specified.
2. Intel specific registers associated with each of the Clause 45 MMDs are placed in the Intel specific area beginning at 0xC000, according to the register map listed in Table 10-3.

Table 10-3. Register Layout

Base Offset (Hex)	Description
C000	Tx & Overall MMD Control
C400	Tx & Overall MMD Provisioning
C800	Tx & Overall MMD State
CC00	Tx & Overall MMD Alarms
D000	Standard Interrupt Mask
D400	Tx & Overall MMD Interrupt Mask
D800	Tx & Overall MMD Debug
DC00	Reserved
E000	Rx Control
E400	Rx Provisioning
E800	Rx State
EC00	Rx Alarms
F000	Standard Interrupt Mask
F400	Rx Interrupt Mask
F800	Rx Debug
FC00	Global Interrupt Flags

The table is split into a transmit portion, and a receive portion, with the transmit portion also containing any overall Intel specific registers for the MMD. In this table, the following definitions apply:

Table 10-4. Terms Used within the Register Layout

Term	Definition
Control	Action bits that affect the operation of the MMD, such as reset.
Provisioning	Static provisioning bits that control the behavior of the MMD.
State	Bits that reflect the state of the MMD.
Alarm	Bits that can generate maskable interrupts.
Standard Interrupt Mask	Interrupt masks for alarm bits defined in the Clause 45 register set.
Intel Specific Interrupt Mask	Interrupt masks for Intel Specific alarms.

3. Interrupts are handled in an hierarchical fashion, with the PHY top level interrupt indication being reported in the EICR register. Below this are two maskable interrupt trees: one composed of standard interrupts, and one composed of Intel defined interrupts. The top level summary register for these trees resides at the end of the register space in MMD #30 - the Global PHY Standard Interrupt Flags (1E.FC00). Feeding this are interrupt registers from each of the individual MMDs.
   a. The standard interrupt tree is designed so that the source of any interrupt can be determined in a maximum of two MDIO reads.
   b. The Intel defined interrupt tree requires at most three MDIO reads to determine the source of an interrupt.
   c. All interrupts are maskable, whether they are from the Standard interrupt tree, or from the Intel Specific interrupt tree.

10.1.4 PHY Registers and Documentation

The PHY registers for X550 are provided in the following tables, listed by the numerical order of their MMD address.

10.2 PMA Registers

10.2.1 PMA Standard Control 1: Address 1.0

Bit	Name	Type	Default	Description
0	Loopback	RW PD	0b	**Loopback** This enables the PMA analog system loopback. 0b = Normal operation. 1b = Enable loopback mode.
1	Reserved	RSV		Reserved. Do not modify.
5:2	10 GbE Speed Selection [3:0]	ROS	0x0	**10 GbE Speed Selection** 0000b = 10 GbE xxx1b = 10PASS-TS / 2BASE-TL xx1xb = Reserved x1xxb = Reserved 1xxxb = Reserved
6	Speed Selection MSB	RW PD	1b	**Speed Selection MSB** {6,D}: 00b = Not supported 01b = 100 Mb/s 10b = 1000 Mb/s 11b = Speed set by Bits [5:2]
A:7	Reserved	RSV		Reserved. Do not modify.
B	Low Power	RW PD	0b	**Low Power** A 1b written to this register causes the PMA to enter low-power mode. If a global chip low-power state is desired, use Bit [B] in the Global Standard Control 1: Address 1E.0 register. 0b = Normal operation. 1b = Low-power mode.
C	Reserved	RSV		Reserved. Do not modify.
D	Speed Selection LSB	RW PD	1b	**Speed Selection LSB** {6,D}: 00b = Not supported 01b = 100 Mb/s 10b = 1000 Mb/s 11b = Speed set by Bits [5:2]
F:E	Reserved	RSV		Reserved. Do not modify.

10.2.2 PMA Standard Status 1: Address 1.1

Bit	Name	Type	Default	Description
0	Reserved	RSV		Reserved. Do not modify.
1	Low Power Ability	ROS	1b	**Low Power Ability** Indicates whether the PHY supports a low-power mode. 0b = No low-power mode supported. 1b = PMA supports low-power mode.

Bit	Name	Type	Default	Description
2	PMA Receive Link Status	LL		**PMA Receive Link Status** This indicates the status of the PMA receive link. This is the latched version of 1.E800.0 (see Section 10.2.46). 0b = Link lost since last read. 1b = Link up. ***Note:*** This is latching low, so it can only be used to detect link drops, and not the current status of the link without performing back-to-back reads.
6:3	Reserved	RSV		Reserved. Do not modify.
7	Fault	RO		**Fault** This is the top-level fault indicator flag for the PMA. This bit is set if either of the two bits 1.8.B or 1.8.A are set (see Section 10.2.9). 0b = No fault detected. 1b = Fault condition detected.
F:8	Reserved	RSV		Reserved. Do not modify.

10.2.3 PMA Standard Device Identifier 1: Address 1.2

Bit	Name	Type	Default	Description
F:0	Device ID MSW [1F:10]	RO		**Device ID MSW** Bits [31:16] of the Device ID.

10.2.4 PMA Standard Device Identifier 2: Address 1.3

Bit	Name	Type	Default	Description
F:0	Device ID LSW [F:0]	RO		**Device ID LSW** Bits [15-0] of the Device ID.

10.2.5 PMA Standard Speed Ability: Address 1.4

Bit	Name	Type	Default	Description
0	PMA 10 GbE Capable	ROS	1b	**PMA 10 GbE Capable** 0b = PMA is not 10 GbE capable. 1b = PMA is 10 GbE capable. This is always set to 1b in the X550.
1	PMA 2BASE-TL Capable	ROS	0b	**PMA 2BASE-TL Capable** 0b = PMA is not 2BASE-TL capable. 1b = PMA is 2BASE-TL capable. This is always set to 0b in the X550.
2	PMA 10PASS-TS Capable	ROS	0b	**PMA 10PASS-TS Capable** 0b = PMA is not 10PASS-TS capable. 1b = PMA is 10PASS-TS capable. This is always set to 0b in the X550.
3	Reserved	RSV		Reserved. Do not modify.

Bit	Name	Type	Default	Description
4	PMA 1 GbE Capable	ROS	1b	**PMA 1 GbE Capable** 0b = PMA is not 1 GbE capable. 1b = PMA is 1 GbE capable. This is always set to 1b in the X550.
5	PMA 100 Mb/s Capable	ROS	1b	**PMA 100 Mb/s Capable** 0b = PMA is not 100 Mb/s capable. 1b = PMA is 100 Mb/s capable. This is always set to 1b in the X550.
6	PMA 10 Mb/s Capable	ROS	0b	**PMA 10 Mb/s Capable** 0b = PMA is not 10 Mb/s capable. 1b = PMA is 10 Mb/s capable. This is always set to 0b in the X550.
F:7	Reserved	RSV		Reserved. Do not modify.

10.2.6 PMA Standard Devices in Package 1: Address 1.5

Bit	Name	Type	Default	Description
0	Clause 22 Registers Present	ROS	0b	**Clause 22 Registers Present** 0b = Clause 22 registers are not present in the package. 1b = Clause 22 registers are present in the package. This is always set to 0b, as there are no Clause 22 registers in the X550.
1	PMA Present	ROS	1b	**PMA Present** 0b = PMA is not present. 1b = PMA is present in the package. This is always set to 1b, as there is PMA functionality in the X550.
2	WIS Present	ROS	0b	**WIS Present** 0b = WIS is not present in the package. 1b = WIS is present in the package. This is always set to 0b, as there is no WIS functionality in the X550.
3	PCS Present	ROS	1b	**PCS Present** 0b = PCS is not present in the package. 1b = PCS is present in the package. This is always set to 1b, as there is PCS functionality in the X550.
4	PHY XS Present	ROS	0b	**PHY XS Present** 0b = PHY XS is not present in the package. 1b = PHY XS is present in the package. This is always set to 0b as there is no PHY XS interface in the X550.,
5	DTE XS Present	ROS	0b	**DTE XS Present** 0b = DTE XS is not present in the package. 1b = DTE XS is present in the package. This is always set to 0b, as there is no DTE XAUI interface in the X550.
6	TC Present	ROS	0b	**TC Present** 0b = TC is not present in the package. 1b = TC is present in the package. This is always set to 0b, as there is no TC functionality in the X550.
7	Auto-Negotiation Present	ROS	1b	**Auto-Negotiation Present** 0b = Auto-negotiation is not present in the package. 1b = Auto-negotiation is present in the package. This is always set to 1b, as there is auto-negotiation in the X550.
F:8	Reserved	RSV		Reserved. Do not modify.

10.2.7 PMA Standard Devices in Package 2: Address 1.6

Bit	Name	Type	Default	Description
C:0	Reserved	RSV		Reserved. Do not modify.
D	Clause 22 Extension Present	ROS	1b	**Clause 22 Extension Present** 0b = Clause 22 extension is not present in the package. 1b = Clause 22 extension is present in the package. This is always set to 1b, as the X550 uses this device for the GbE registers.
E	Vendor Specific Device #1 Present	ROS	1b	**Vendor-Specific Device #1 Present** 0b = Device #1 is not present in the package. 1b = Device #1 is present in the package. This is always set to 1b, as the X550 uses this device for the Global registers.
F	Vendor Specific Device #2 Present	ROS	1b	**Vendor-Specific Device #2 Present** 0b = Device #2 is not present in the package. 1b = Device #2 is present in the package. This is always set to 1b, as the X550 uses this device for the DSP PMA registers.

10.2.8 PMA Standard Control 2: Address 1.7

Bit	Name	Type	Default	Description
3:0	PMA Device Type [3:0]	ROS	0x9	**PMA Device Type** 0x9 = 10GBASE-T PMA type. All other values are reserved. This is always set to 0x9, as the X550 is a 10GBASE-T device.
F:4	Reserved	RSV		Reserved. Do not modify.

10.2.9 PMA Standard Status 2: Address 1.8

Bit	Name	Type	Default	Description
0	PMA Loopback Ability	ROS	1b	**PMA Loopback Ability** 0b = PMA does not support loopback. 1b = PMA supports loopback. This is always set to 1b, as the PMA in the X550 supports loopback.
1	PMA 10GBASE-EW Capable	ROS	0b	**PMA 10GBASE-EW Capable** 0b = PMA does not support 10GBASE-EW. 1b = PMA supports 10GBASE-EW. This field is always set to 0b, as the PMA in the X550 supports only 10GBASE-T.
2	PMA 10GBASE-LW Capable	ROS	0b	**PMA 10GBASE-LW Capable** 0b = PMA does not support 10GBASE-LW. 1b = PMA supports 10GBASE-LW. This field is always set to 0b, as the PMA in the X550 supports only 10GBASE-T.
3	PMA 10GBASE-SW Capable	ROS	0b	**PMA 10GBASE-SW Capable** 0b = PMA does not support 10GBASE-SW. 1b = PMA supports 10GBASE-SW. This field is always set to 0b, as the PMA in the X550 supports only 10GBASE-T.

Bit	Name	Type	Default	Description
4	PMA 10GBASE-LX4 Capable	ROS	0b	**PMA 10GBASE-LX4 Capable** 0b = PMA does not support 10GBASE-LX4. 1b = PMA supports 10GBASE-LX4. This field is always set to 0b, as the PMA in the X550 supports only 10GBASE-T.
5	PMA 10GBASE-ER Capable	ROS	0b	**PMA 10GBASE-ER Capable** 0b = PMA does not support 10GBASE-ER. 1b = PMA supports 10GBASE-ER. This field is always set to 0b, as the PMA in the X550 supports only 10GBASE-T.
6	PMA 10GBASE-LR Capable	ROS	0b	**PMA 10GBASE-LR Capable** 0b = PMA does not support 10GBASE-LR. 1b = PMA supports 10GBASE-LR. This field is always set to 0b, as the PMA in the X550 supports only 10GBASE-T.
7	PMA 10GBASE-SR Capable	ROS	0b	**PMA 10GBASE-SR Capable** 0b = PMA does not support 10GBASE-SR. 1b = PMA supports 10GBASE-SR. This field is always set to 0b, as the PMA in the X550 supports only 10GBASE-T.
8	PMD Transmit Disable Ability	ROS	1b	**PMD Transmit Disable Ability** This bit indicates whether the PMD has the capability of disabling its transmitter. 0b = PMD does not have the capability of disabling the transmitter. 1b = PMD has the capability of disabling the transmitter. This field is always set to 1b, as the PMD in the X550 has this ability.
9	Extended Abilities	ROS	1b	**Extended Abilities** 0b = PMA does not have extended abilities. 1b = PMA has extended abilities listed in register 1.11. This field is set to 1b, as the PMA in the X550 has extended abilities.
A	Receive Fault	LH		**Receive Fault** This bit indicates whether there is a fault somewhere along the receive path. This is a hardware fault and should never occur during normal operation. 0b = No fault condition on receive path. 1b = Fault condition on receive path.
B	Transmit Fault	LH		**Transmit Fault** This bit indicates whether there is a fault somewhere along the transmit path. This is a hardware fault and should never occur during normal operation. 0b = No fault condition on transmit path. 1b = Fault condition on transmit path.
C	Receive Fault Location Ability	ROS	1b	**Receive Fault Location Ability** This bit indicates whether the PMA has the ability to locate faults along the receive path. 0b = PMA does not have the capability to detect a fault condition on the receive path. 1b = PMA has the capability to detect a fault condition on the receive path.
D	Transmit Fault Location Ability	ROS	1b	**Transmit Fault Location Ability** This bit indicates whether the PMA has the ability to locate faults along the transmit path. 0b = PMA does not have the capability to detect a fault condition on the transmit path. 1b = PMA has the capability to detect a fault condition on the transmit path.
F:E	Device Present [1:0]	ROS	10b	**Device Present** 00b = No device at this address. 01b = No device at this address. 10b = Device present at this address. 11b = No device at this address. This field is always set to 10b, as the PMA is present in the X550.

10.2.10 PMD Standard Transmit Disable Control: Address 1.9

Bit	Name	Type	Default	Description
0	PMD Global Transmit Disable	RW PD	0b	**PMD Global Transmit Disable** When set, this bit disables (and overrides) all four channels, and sets the average launch power on all pairs to less than -53 dBm. 0b = Normal operation. 1b = Disable output on all channels.
1	PMD Channel 0 Transmit Disable	RW PD	0b	**PMD Channel 0 Transmit Disable** When disabled, the average launch power on a pair is set to less than -53 dBm. 0b = Normal operation. 1b = Disable output on transmit channel 0.
2	PMD Channel 1 Transmit Disable	RW PD	0b	**PMD Channel 1 Transmit Disable** When disabled, the average launch power on a pair is set to less than -53 dBm. 0b = Normal operation. 1b = Disable output on transmit channel 1.
3	PMD Channel 2 Transmit Disable	RW PD	0b	**PMD Channel 2 Transmit Disable** When disabled, the average launch power on a pair is set to less than -53 dBm. 0b = Normal operation. 1b = Disable output on transmit channel 2.
4	PMD Channel 3 Transmit Disable	RW PD	0b	**PMD Channel 3 Transmit Disable** When disabled, the average launch power on a pair is set to less than -53 dBm. 0b = Normal operation. 1b = Disable output on transmit channel 3.
F:5	Reserved	RSV		Reserved. Do not modify.

10.2.11 PMD Standard Signal Detect: Address 1.A

Bit	Name	Type	Default	Description
0	PMD Global Signal Detect	RO		**PMD Global Signal Detect** This bit is marked when all required, valid Ethernet signals to create a connection are present on the line. 0b = No signal detected. 1b = Signals detected on all required channels.
1	PMD Channel 0 Signal Detect	RO		**PMD Channel 0 Signal Detect** These bits are used to indicate the presence of signals on a given pair. A signal is defined as an auto-negotiation pulse or Ethernet signals. 0b = No signal detected. 1b = Signal detected on receive channel 0.

Bit	Name	Type	Default	Description
2	PMD Channel 1 Signal Detect	RO		**PMD Channel 1 Signal Detect** These bits are used to indicate the presence of signals on a given pair. A signal is defined as an auto-negotiation pulse or Ethernet signals. 0b = No signal detected. 1b = Signal detected on receive channel 1.
3	PMD Channel 2 Signal Detect	RO		**PMD Channel 2 Signal Detect** These bits are used to indicate the presence of signals on a given pair. A signal is defined as an auto-negotiation pulse or Ethernet signals. 0b = No signal detected. 1b = Signal detected on receive channel 2.
4	PMD Channel 3 Signal Detect	RO		**PMD Channel 3 Signal Detect** These bits are used to indicate the presence of signals on a given pair. A signal is defined as an auto-negotiation pulse or Ethernet signals. 0b = No signal detected. 1b = Signal detected on receive channel 3.
F:5	Reserved	RSV		Reserved. Do not modify.

10.2.12 PMD Standard 10G Extended Ability Register: Address 1.B

Bit	Name	Type	Default	Description
1:0	Reserved	RSV		Reserved. Do not modify.
2	PMA 10GBASE-T Capable	ROS	1b	**PMA 10GBASE-T Capable** 0b = PMA incapable of 10GBASE-T. 1b = PMA capable of 10GBASE-T. This field is set to 1b, as the PMA in the X550 supports 10GBASE-T.
F:3	Reserved	RSV		Reserved. Do not modify.

10.2.13 PMA Standard Package Identifier 1: Address 1.E

Bit	Name	Type	Default	Description
F:0	Package ID MSW [1F:10]	RO		**Package ID MSW** Bits [31:16] of the Package ID.

10.2.14 PMA Standard Package Identifier 2: Address 1.F

Bit	Name	Type	Default	Description
F:0	Package ID LSW [F:0]	RO		**Package ID LSW** Bits [15:0] of the Package ID.

10.2.15 PMA 10GBASE-T Status: Address 1.81

Bit	Name	Type	Default	Description
0	Link Partner Information Valid	RO		**Link Partner Information Valid** When set, this bit indicates that the start-up protocol has completed. 0b = 10GBASE-T link partner information is not valid. 1b = 10GBASE-T link partner information is valid.
F:1	Reserved	RSV		Reserved. Do not modify.

10.2.16 PMA 10GBASE-T Pair Swap and Polarity Status: Address 1.82

Bit	Name	Type	Default	Description
1:0	MDI / MD-X Connection State [1:0]	RO		**MDI/MD-X Connection State** These two bits indicate the current status of pair swaps at the MDI / MD-X 00b = Pair A/B and C/D crossover. 01b = Pair C/D crossover. 10b = Pair A/B crossover. 11b = No crossover.
7:2	Reserved	RSV		Reserved. Do not modify.
B:8	Pair Polarity [3:0]	RO		**Pair Polarity** When set, this bit indicates that the wires on the respective pair are reversed. 0b = Polarity of pair is normal. 1b = Polarity of pair is reversed. where: [0] = Pair A polarity. [1] = Pair B polarity. [2] = Pair C polarity. [3] = Pair D polarity.
F:C	Reserved	RSV		Reserved. Do not modify.

10.2.17 PMA 10GBASE-T Tx Power Backoff Setting: Address 1.83

Bit	Name	Type	Default	Description
9:0	Reserved	RSV		Reserved. Do not modify.
C:A	Tx Power Backoff [2:0]	RO		**Transmit Power Backoff** The power backoff of the PMA. 000b = 0 dB 001b = 2 dB 010b = 4 dB 011b = 6 dB 100b = 8 dB 101b = 10 dB 110b = 12 dB 111b = 14 dB

Bit	Name	Type	Default	Description
F:D	Link Partner Tx Power Backoff [2:0]	RO		**Link Partner Transmit Power Backoff** The power backoff of the link partner. 000b = 0 dB 001b = 2 dB 010b = 4 dB 011b = 6 dB 100b = 8 dB 101b = 10 dB 110b = 12 dB 111b = 14 dB

10.2.18 PMA 10GBASE-T Test Modes: Address 1.84

Bit	Name	Type	Default	Description
9:0	Reserved	RSV		Reserved. Do not modify.
C:A	Transmitter Test Frequencies [2:0]	RW PD	0x0	**Transmitter Test Frequencies** The test frequencies associated with test mode #4 in [F:D]. 000b = Reserved 001b = Dual tone #1 010b = Dual tone #2 011b = Reserved 100b = Dual tone #3 101b = Dual tone #4 110b = Dual tone #5 111b = Reserved
F:D	Test Mode Control [2:0]	RW PD	0x0	**Test Mode Control** Test mode control for the PMA as defined in Section 55.5.2 of 802.3an. 000b = Normal operation. 001b = Master source for slave mode jitter test. 010b = Master mode jitter test. 011b = Slave mode jitter test. 100b = Transmitter distortion test. 101b = PSD and power level test. 110b = Transmitter droop test. 111b = Pseudo random test mode for BER Monitor.

10.2.19 PMA 10GBASE-T SNR Operating Margin Channel A: Address 1.85

Bit	Name	Type	Default	Description
F:0	Channel A Operating Margin [F:0]	RO		**Channel A Operating Margin** Operating margin (dB) of Channel A. The excess SNR that is used by the channel, over and above the minimum SNR required to operate at a BER of 10^{-12}. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -12.7 dB to 12.7 dB. The number is in offset binary, with 0.0 dB represented by 0x8000.

10.2.20 PMA 10GBASE-T SNR Operating Margin Channel B: Address 1.86

Bit	Name	Type	Default	Description
F:0	Channel B Operating Margin [F:0]	RO		**Channel B Operating Margin** Operating margin (dB) of Channel B. The excess SNR that is used by the channel, over and above the minimum SNR required to operate at a BER of 10^{-12}. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -12.7 dB to 12.7 dB. The number is in offset binary, with 0.0 dB represented by 0x8000.

10.2.21 PMA 10GBASE-T SNR Operating Margin Channel C: Address 1.87

Bit	Name	Type	Default	Description
F:0	Channel C Operating Margin [F:0]	RO		**Channel C Operating Margin** Operating margin (dB) of Channel C. The excess SNR that is used by the channel, over and above the minimum SNR required to operate at a BER of 10^{-12}. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -12.7 dB to 12.7 dB. The number is in offset binary, with 0.0 dB represented by 0x8000.

10.2.22 PMA 10GBASE-T SNR Operating Margin Channel D: Address 1.88

Bit	Name	Type	Default	Description
F:0	Channel D Operating Margin [F:0]	RO		**Channel D Operating Margin** Operating margin (dB) of Channel D. The excess SNR that is used by the channel, over and above the minimum SNR required to operate at a BER of 10^{-12}. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -12.7 dB to 12.7 dB. The number is in offset binary, with 0.0 dB represented by 0x8000.

10.2.23 PMA 10GBASE-T SNR Minimum Operating Margin Channel A: Address 1.89

Bit	Name	Type	Default	Description
F:0	Channel A Minimum Operating Margin [F:0]	RO		**Channel A Minimum Operating Margin** Minimum operating margin (dB) of Channel A since the last link up. The excess SNR that is used by the channel, over and above the minimum SNR required to operate at a BER of 10^{-12}. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -12.7 dB to 12.7 dB. The number is in offset binary, with 0.0 dB represented by 0x8000.

10.2.24 PMA 10GBASE-T SNR Minimum Operating Margin Channel B: Address 1.8A

Bit	Name	Type	Default	Description
F:0	Channel B Minimum Operating Margin [F:0]	RO		**Channel B Minimum Operating Margin** Minimum operating margin (dB) of Channel B since the last link up. The excess SNR that is used by the channel, over and above the minimum SNR required to operate at a BER of 10^{-12}. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -12.7 dB to 12.7 dB. The number is in offset binary, with 0.0 dB represented by 0x8000.

10.2.25 PMA 10GBASE-T SNR Minimum Operating Margin Channel C: Address 1.8B

Bit	Name	Type	Default	Description
F:0	Channel C Minimum Operating Margin [F:0]	RO		**Channel C Minimum Operating Margin** Minimum operating margin (dB) of Channel C since the last link up. The excess SNR that is used by the channel, over and above the minimum SNR required to operate at a BER of 10^{-12}. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -12.7 dB to 12.7 dB. The number is in offset binary, with 0.0 dB represented by 0x8000.

10.2.26 PMA 10GBASE-T SNR Minimum Operating Margin Channel D: Address 1.8C

Bit	Name	Type	Default	Description
F:0	Channel D Minimum Operating Margin [F:0]	RO		**Channel D Minimum Operating Margin** Minimum operating margin (dB) of Channel D since the last link up. The excess SNR that is used by the channel, over and above the minimum SNR required to operate at a BER of 10^{-12}. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -12.7 dB to 12.7 dB. The number is in offset binary, with 0.0 dB represented by 0x8000.

10.2.27 PMA 10GBASE-T Receive Signal Power Channel A: Address 1.8D

Bit	Name	Type	Default	Description
F:0	Channel A Received Signal Power [F:0]	RO		**Channel A Received Signal Power** Received signal power (dBm) for Channel A. The received signal power on the channel. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -20.0 dB to +5.5 dB. The number is in offset two's complement notation, with 0.0 dB represented by 0x8000.

10.2.28 PMA 10GBASE-T Receive Signal Power Channel B: Address 1.8E

Bit	Name	Type	Default	Description
F:0	Channel B Received Signal Power [F:0]	RO		**Channel B Received Signal Power** Received signal power (dBm) for Channel B. The received signal power on the channel. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -20.0 dB to +5.5 dB. The number is in offset two's complement notation, with 0.0 dB represented by 0x8000.

10.2.29 PMA 10GBASE-T Receive Signal Power Channel C: Address 1.8F

Bit	Name	Type	Default	Description
F:0	Channel C Received Signal Power [F:0]	RO		**Channel C Received Signal Power** Received signal power (dBm) for Channel C. The received signal power on the channel. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -20.0 dB to +5.5 dB. The number is in offset two's complement notation, with 0.0 dB represented by 0x8000.

10.2.30 PMA 10GBASE-T Receive Signal Power Channel D: Address 1.90

Bit	Name	Type	Default	Description
F:0	Channel D Received Signal Power [F:0]	RO		**Channel D Received Signal Power** Received signal power (dBm) for Channel D. The received signal power on the channel. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -20.0 dB to +5.5 dB. The number is in offset two's complement notation, with 0.0 dB represented by 0x8000.

10.2.31 PMA 10GBASE-T Skew Delay 1: Address 1.91

Bit	Name	Type	Default	Description
7:0	Reserved	RSV		Reserved. Do not modify.
E:8	Skew Delay B [6:0]	RO		**Skew Delay for Pair B** The skew delay reports the current skew delay on each of the pair with respect to physical pair A. It is reported with 1.25 ns resolution to an accuracy of 2.5 ns. The number is in 2's complement notation with positive values representing delay and negative values representing advance with respect to physical pair A. If the delay exceeds the maximum amount that can be represented by the range (-80 ns to +78.75 ns), the field displays the maximum respective value.
F	Reserved	RSV		Reserved. Do not modify.

10.2.32 PMA 10GBASE-T Skew Delay 2: Address 1.92

Bit	Name	Type	Default	Description
6:0	Skew Delay C [6:0]	RO		**Skew Delay for Pair C** The skew delay reports the current skew delay on each of the pair with respect to physical pair A. It is reported with 1.25 ns resolution to an accuracy of 2.5 ns. The number is in two's complement notation with positive values representing delay and negative values representing advance with respect to physical pair A. If the delay exceeds the maximum amount that can be represented by the range (-80 ns to +78.75 ns), the field displays the maximum respective value.
7	Reserved	RSV		Reserved. Do not modify.
E:8	Skew Delay D [6:0]	RO		**Skew Delay for Pair D** The skew delay reports the current skew delay on each of the pair with respect to physical pair A. It is reported with 1.25 ns resolution to an accuracy of 2.5 ns. The number is in two's complement notation with positive values representing delay and negative values representing advance with respect to physical pair A. If the delay exceeds the maximum amount that can be represented by the range (-80 ns to +78.75 ns), the field displays the maximum respective value.
F	Reserved	RSV		Reserved. Do not modify.

10.2.33 PMA 10GBASE-T Fast Retrain Status and Control: Address 1.93

Bit	Name	Type	Default	Description
0	Fast Retrain Enable	RW	0b	**Fast Retrain Enable** 0b = Fast retrain capability is disabled. 1b = Fast retrain capability is enabled.
2:1	Fast Retrain Signal Type [1:0]	RW	00b	**Fast Retrain Signal Type** 00b = PHY signals IDLE during fast retrain. 01b = PHY signals Local Fault during fast retrain. 10b = PHY signals Link Interruption during fast retrain. 11b = Reserved.
3	Fast Retrain Negotiated	RO		**Fast Retrain Negotiated** 0b = Fast retrain capability was not negotiated. 1b = Fast retrain capability was negotiated.
4	Fast Retrain Ability	RO		**Fast Retrain Ability** 0b = Fast retrain capability is not supported. 1b = Fast retrain capability is supported.
5	Reserved	RS		Reserved. Do not modify.
A:6	LD Fast Retrain Count [4:0]	SCT	0x0	**Local Device Fast Retrain Count** Counts the number of fast retrains requested by the local device. Saturating clear on read counter.
F:B	LP Fast Retrain Count [4:0]	SCT	0x0	**Link Partner Fast Retrain Count** Counts the number of fast retrains requested by the link partner. Saturating clear on read counter.

10.2.34 PMA TimeSync Capability: Address 1.1800

Bit	Name	Type	Default	Description
0	TimeSync Receive Path Data Delay	RO	0b	**TimeSync Receive Path Data Delay** 0b = PMA does not provide information on receive path data delay. 1b = PMA provides information on receive path data delay in registers 1.1805 through 1.1808.
1	TimeSync Transmit Path Data Delay	RO	0b	**TimeSync Transmit Path Data Delay** 0b = PMA does not provide information on transmit path data delay. 1b = PMA provides information on transmit path data delay in registers 1.1801 through 1.1804.
F:2	Reserved	RSV	0x0	Reserved. Do not modify.

10.2.35 PMA TimeSync Transmit Path Data Delay 1: Address 1.1801

Bit	Name	Type	Default	Description
F:0	Maximum PMA Transmit Path Data Delay LSW [F:0]	RO		**Maximum PMA Transmit Path Data Delay LSW** LSW of maximum PMA transmit delay in nanoseconds.

10.2.36 PMA TimeSync Transmit Path Data Delay 2: Address 1.1802

Bit	Name	Type	Default	Description
F:0	Maximum PMA Transmit Path Data Delay MSW [F:0]	RO		**Maximum PMA Transmit Path Data Delay MSW** MSW of maximum PMA transmit delay in nanoseconds.

10.2.37 PMA TimeSync Transmit Path Data Delay 3: Address 1.1803

Bit	Name	Type	Default	Description
F:0	Minimum PMA Transmit Path Data Delay LSW [F:0]	RO		**Minimum PMA Transmit Path Data Delay LSW** LSW of minimum PMA transmit delay in nanoseconds.

10.2.38 PMA TimeSync Transmit Path Data Delay 4: Address 1.1804

Bit	Name	Type	Default	Description
F:0	Minimum PMA Transmit Path Data Delay MSW [F:0]	RO		**Minimum PMA Transmit Path Data Delay MSW** MSW of minimum PMA transmit delay in nanoseconds.

10.2.39 PMA TimeSync Receive Path Data Delay 1: Address 1.1805

Bit	Name	Type	Default	Description
F:0	Maximum PMA Receive Path Data Delay LSW [F:0]	RO		**Maximum PMA Receive Path Data Delay LSW** LSW of maximum PMA Receive delay in nanoseconds.

10.2.40 PMA TimeSync Receive Path Data Delay 2: Address 1.1806

Bit	Name	Type	Default	Description
F:0	Maximum PMA Receive Path Data Delay MSW [F:0]	RO		**Maximum PMA Receive Path Data Delay MSW** MSW of maximum PMA Receive delay in nanoseconds.

10.2.41 PMA TimeSync Receive Path Data Delay 3: Address 1.1807

Bit	Name	Type	Default	Description
F:0	Minimum PMA Receive Path Data Delay LSW [F:0]	RO		**Minimum PMA Receive Path Data Delay LSW** LSW of minimum PMA Receive delay in nanoseconds.

10.2.42 PMA TimeSync Receive Path Data Delay 4: Address 1.1808

Bit	Name	Type	Default	Description
F:0	Minimum PMA Receive Path Data Delay MSW [F:0]	RO		**Minimum PMA Receive Path Data Delay MSW** MSW of minimum PMA Receive delay in nanoseconds.

10.2.43 PMA Transmit Standard Interrupt Mask 1: Address 1.D000

Bit	Name	Type	Default	Description
1:0	Reserved	RSV		Reserved. Do not modify.
2	PMA Receive Link Status Mask	RW PD	0b	**PMA Receive Link Status Mask** Mask for Bit 1.1.2 (see Section 10.2.2). 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F:3	Reserved	RSV		Reserved. Do not modify.

10.2.44 PMA Transmit Standard Interrupt Mask 2: Address 1.D001

Bit	Name	Type	Default	Description
9:0	Reserved	RSV		Reserved. Do not modify.
A	Receive Fault Mask	RW PD	0b	**Receive Fault Mask** Mask for Bit 1.8.A (see Section 10.2.9). 0b = Disable interrupt generation. 1b = Enable interrupt generation.
B	Transmit Fault Mask	RW PD	0b	**Transmit Fault Mask** Mask for Bit 1.8.B (see Section 10.2.9). 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F:C	Reserved	RSV		Reserved. Do not modify.

10.2.45 PMA Receive Reserved Vendor Provisioning 1: Address 1.E400

Bit	Name	Type	Default	Description
0	MDI Configuration	RW PD	0b	**MDI Configuration** Setting this bit determines whether MDI is reversed or not. ***Note:*** The reversal does not change pair polarity. For example, A+ maps to D+, etc. The value of this bit is sampled during auto-negotiation. If this bit is changed manually after auto-negotiation completes, auto-negotiation must be restarted to achieve the desired MDI configuration. 0b = MDI Normal (ABCD -> ABCD) 1b = MDI Reversed (ABCD -> DCBA)
1	Force MDI Configuration	RW PD	0b	**Force MDI Configuration** MDI_CFG pin is not brought out. Set this bit to 1b and use the MDI Configuration bit to control the MDI pairs' arrangement. 0b = Set MDI Configuration based on state of MDI_CFG. 1b = Ignore state of MDI_CFG pin.
2	Enable X550 Cisco Fast Retrain	RW PD	0b	**Enable X550 Cisco Fast Retrain** If the link is a X550 PHY and also has Cisco Fast Retrain enabled, use a special retrain sequence to bring the link back up without going back through the auto-negotiation sequence. 0b = Disable PMA fast link retrain. 1b = Enable PMA fast link retrain.
E:3	Reserved Receive Provisioning 1 [B:0]	RW PD	0x0	**Reserved Receive Provisioning 1** Reserved for future use.
F	External PHY Loopback	RW PD	0b	**External PHY Loopback** External PHY loopback expects a loopback connector such that Pair A is connected to Pair B, and Pair C is connected to Pair D. 0b = Normal operation. 1b = Enable external PHY loopback.

10.2.46 PMA Receive Vendor State 1: Address 1.E800

Bit	Name	Type	Default	Description
0	PMA Receive Link Current Status	RO		**PMA Receive Link Current Status** This is the current state of Bit 1.1.2 (see Section 10.2.2). 1b = Rx link good.
F:1	Reserved	RSV		Reserved. Do not modify.

10.2.47 PMA Receive Vendor State 2: Address 1.E811

Bit	Name	Type	Default	Description
7:0	Total Number Of RFI Training Link Recovery Events Since Last AutoNeg [7:0]	RO		**Total Number Of RFI Training Link Recovery Events Since Last Auto-Negotiation** The count of the cumulative number of RFI training link recovery events since last auto-negotiation. This register is automatically reset to zero during auto-negotiation. The result is reported modulo 256 (wrap-around).
F:8	Total Number Of Link Recovery Events Since Last AutoNeg [7:0]	RO		**Total Number Of Link Recovery Events Since Last Auto-Negotiation** The count of the cumulative number of Link Recovery Events since last auto-negotiation. This register is automatically reset to zero during auto-negotiation. It increments once for each series of back-to-back fast retrain events. The result is reported modulo 256 (wrap-around).

10.2.48 PMA Vendor Global Interrupt Flags 1: Address 1.FC00

Bit	Name	Type	Default	Description
9:0	Reserved	RSV		Reserved. Do not modify.
A	Standard Alarm 2 Interrupt	RO		**Standard Alarm 2 Interrupt** An interrupt was generated from either Bit 1.8.B or 1.8.A. An interrupt was generated from status register PMA Standard Status 2: Address 1.8 and the corresponding mask register PMA Transmit Standard Interrupt Mask 2: Address 1.D001. 1b = Interrupt
B	Standard Alarm 1 Interrupt	RO		**Standard Alarm 1 Interrupt** An interrupt was generated from Bit 1.1.2. An interrupt was generated from status register PMA Standard Status 1: Address 1.1 and the corresponding mask register PMA Transmit Standard Interrupt Mask 1: Address 1.D000. 1b = Interrupt
F:C	Reserved	RSV		Reserved. Do not modify.

10.3 PCS Registers

10.3.1 PCS Standard Control 1: Address 3.0

Bit	Name	Type	Default	Description
1:0	Reserved	RSV		Reserved. Do not modify.
5:2	10 GbE Speed Selection [3:0]	RW PD	0x0	**10 GbE Speed Selection** 0000b = 10 GbE xxx1b = 10PASS-TS / 2BASE-TL xx1xb = Reserved x1xxb = Reserved 1xxxb = Reserved
6	Speed Selection MSB	RW PD	1b	**Speed Selection MSB** {6,D}: 00b = Not supported. 01b = 100 Mb/s 10b = 1000 Mb/s 11b = Speed set by Bits [5:2]
9:7	Reserved	RSV		Reserved. Do not modify.
A	Clock Stop Enable	RW PD	0b	**Clock Stop Enable** 0b = Clock not stoppable. 1b = The PHY may stop the clock during LPI.
B	Low Power	RW PD	0b	**Low Power** A 1b written to this register causes the PCS to enter low-power mode. If a global chip low-power state is desired, Bit [B] in the Global Standard Control 1: Address 1E.0 register should be set (Section 10.6.1). 0b = Normal operation. 1b = Low-power mode.
C	Reserved	RSV		Reserved. Do not modify.
D	Speed Selection LSB	RW PD	1b	**Speed Selection LSB** {6,D}: 00b = Not supported 01b = 100 Mb/s 10b = 1000 Mb/s 11b = Speed set by Bits [5:2]
E	Loopback	RW PD	0b	**Loopback** This enables the PCS DSQ system loopback. 0b = Normal operation. 1b = Enable loopback mode.
F	Reserved	RSV		Reserved. Do not modify.

10.3.2 PCS Standard Status 1: Address 3.1

Bit	Name	Type	Default	Description
0	Reserved	RSV		Reserved. Do not modify.
1	Low Power Ability	ROS	1b	**Low Power Ability** Indicates whether the XAUI interface supports a low-power mode. 0b = No low-power mode supported. 1b = PCS supports low-power mode.
2	PCS Receive Link Status	LL		**PCS Receive Link Status** This indicates the status of the PCS receive link. This is a latching low version of Bit 3.20.C (Section 10.3.12). Status of the PCS receive link. 0b = Link lost since last read. 1b = Link up.
5:3	Reserved	RSV		Reserved. Do not modify.
6	Clock Stop Capable	ROS	0	**Clock Stop Capable** 0b = Clock not stoppable. 1b = The MAC may stop the clock during LPI.
7	Fault	RO		**Fault** This is the top-level fault indicator flag for the PCS block. This bit is set if either of the two Bits 3.8.B or 3.8.A are set (Section 10.3.9). 0b = No fault detected. 1b = Fault condition detected.
8	Rx LPI Indication	RO		**Receive LPI Indication** The source of the LPI signal is configured in 1E.C4A1.3:0. 0b = Rx PCS is not currently receiving LPI. 1b = Rx PCS is currently receiving LPI.
9	Tx LPI Indication	RO		**Transmit LPI Indication** The source of the LPI signal is configured in 1E.C4A1.3:0. 0b = Tx PCS is not currently receiving LPI. 1b = Tx PCS is currently receiving LPI.
A	Rx LPI Received	LH		**Receive LPI Received** The source of the LPI signal is configured in 1E.C4A1.3:0. 0b = LPI not received. 1b = Rx PCS has received LPI.
B	Tx LPI Received	LH		**Transmit LPI Received** The source of the LPI signal is configured in 1E.C4A1.3:0. 0b = LPI not received. 1b = Tx PCS has received LPI.
F:C	Reserved	RSV		Reserved. Do not modify.

10.3.3 PCS Standard Device Identifier 1: Address 3.2

Bit	Name	Type	Default	Description
F:0	Device ID MSW [1F:10]	RO		**Device ID MSW** Bits [31:16] of the Device ID.

10.3.4 PCS Standard Device Identifier 2: Address 3.3

Bit	Name	Type	Default	Description
F:0	Device ID LSW [F:0]	RO		**Device ID LSW** Bits [15:0] of the Device ID.

10.3.5 PCS Standard Speed Ability: Address 3.4

Bit	Name	Type	Default	Description
0	10 GbE Capable	ROS	1b	**10 GbE Capable** 0b = PCS is not 10 GbE capable. 1b = PCS is 10 GbE capable. This is always set to 1b in the X550.
1	PCS 10PASS-TS / 2BASE-TL Capable	ROS	1b	**PCS 10PASS-T /2BASE-TL Capable** 0b = PCS is not 10PASS-TS / 2BASE-TL capable. 1b = PCS is 10PASS-TS / 2BASE-TL capable. This is always set to 0b in the X550.
F:2	Reserved	RSV		Reserved. Do not modify.

10.3.6 PCS Standard Devices in Package 1: Address 3.5

Bit	Name	Type	Default	Description
0	Clause 22 Registers Present	ROS	0b	**Clause 22 Registers Present** 0b = Clause 22 registers are not present in the package. 1b = Clause 22 registers are present in the package. This is always set to 0b, as there are no Clause 22 registers in the X550.
1	PMA Present	ROS	1b	**PMA Present** 0b = PMA is not present. 1b = PMA is present in the package. This is always set to 1b, as there is PMA functionality in the X550.
2	WIS Present	ROS	0b	**WIS Present** 0b = WIS is not present in the package. 1b = WIS is present in the package. This is always set to 0b, as there is no WIS functionality in the X550.
3	PCS Present	ROS	1b	**PCS Present** 0b = PCS is not present in the package. 1b = PCS is present in the package. This is always set to 1b, as there is PCS functionality in the X550.
4	PHY XS Present	ROS	0b	**PHY XS Present** 0b = PHY XS is not present in the package. 1b = PHY XS is present in the package. This is always set to 0b, as there is no PHY XS interface in the X550.
5	DTE XS Present	ROS	0b	**DTE XS Present** 0b = DTE XS is not present in package. 1b = DTE XS is present in the package. This is always set to 0b, as there is no DTE XAUI interface in the X550.

Bit	Name	Type	Default	Description
6	TC Present	ROS	0b	**TC Present** 0b = TC is not present in the package. 1b = TC is present in the package. This is always set to 0b, as there is no TC functionality in the X550.
7	Auto-Negotiation Present	ROS	1b	**Auto-Negotiation Present** 0b = Auto-negotiation is not present in the package. 1b = Auto-negotiation is present in the package. This is always set to 1b, as there is auto-negotiation in the X550.
F:8	Reserved	RSV		Reserved. Do not modify.

10.3.7 PCS Standard Devices in Package 2: Address 3.6

Bit	Name	Type	Default	Description
C:0	Reserved	RSV		Reserved. Do not modify.
D	Clause 22 Extension Present	ROS	1b	**Clause 22 Extension Present** 0b = Clause 22 Extension is not present in the package. 1b = Clause 22 Extension is present in the package. This is always set to 1b, as the X550 uses this device for the GbE registers.
E	Vendor Specific Device #1 Present	ROS	1b	**Vendor Specific Device #1 Present** 0b = Device #1 is not present in the package. 1b = Device #1 is present in the package. This is always set to 1b, as the X550 uses this device for the global control registers.
F	Vendor Specific Device #2 Present	ROS	1b	**Vendor Specific Device #2 Present** 0b = Device #2 is not present in the package. 1b = Device #2 is present in the package. This is always set to 1b, as the X550 uses this device for the DSP PMA registers.

10.3.8 PCS Standard Control 2: Address 3.7

Bit	Name	Type	Default	Description
1:0	PCS Device Type [1:0]	RW PD	11b	**PCS Device Type** 00b = 10GBASE-R 01b = 10GBASE-X 10b = 10GBASE-W 11b = 10GBASE-T The default is 10GBASE-T. This is always set to 11b, as the X550 supports only 10GBASE-T operation.
F:2	Reserved	RSV		Reserved. Do not modify.

10.3.9 PCS Standard Status 2: Address 3.8

Bit	Name	Type	Default	Description
0	10GBASE-R Capable	ROS	0b	**10GBASE-R Capable** 0b = PCS does not support 10GBASE-R. 1b = PCS supports 10GBASE-R PCS type. This field is always set to 0b, as the PCS in the X550 supports only 10GBASE-T.
1	10GBASE-X Capable	ROS	0b	**10GBASE-X Capable** 0b = PCS does not support 10GBASE-X. 1b = PCS supports 10GBASE-X PCS type. This field is always set to 0b, as the PCS in the X550 supports only 10GBASE-T.
2	10GBASE-W Capable	ROS	0b	**10GBASE-W Capable** 0b = PCS does not support 10GBASE-W. 1b = PCS supports 10GBASE-W PCS type. This field is always set to 0b, as the PCS in the X550 supports only 10GBASE-T.
3	10GBASE-T Capable	ROS	1b	**10GBASE-T Capable** 0b = PCS does not support 10GBASE-T. 1b = PCS supports 10GBASE-T PCS type. This field is always set to 1b, as the PCS in the X550 supports only 10GBASE-T.
9:4	Reserved	RSV		Reserved. Do not modify.
A	Receive Fault	LH		**Receive Fault** This bit indicates whether there is a fault somewhere along the receive path. This bit is duplicated at 3.EC04.2. 0b = No fault condition on receive path. 1b = Fault condition on receive path.
B	Transmit Fault	LH		**Transmit Fault** This bit indicates whether there is a fault somewhere along the transmit path. This bit is duplicated at 3.CC01.0. 0b = No fault condition on transmit path. 1b = Fault condition on transmit path.
D:C	Reserved	RSV		Reserved. Do not modify.
F:E	Device Present [1:0]	ROS	10b	**Device Present** 00b = No device at this address. 01b = No device at this address. 10b = Device present at this address. 11b = No device at this address. This field is always set to 10b, as the PCS registers reside here in the X550.

10.3.10 PCS Standard Package Identifier 1: Address 3.E

Bit	Name	Type	Default	Description
F:0	Package ID MSW [1F:10]	RO		**Package ID MSW** Bits [31:16] of the Package ID.

10.3.11 PCS Standard Package Identifier 2: Address 3.F

Bit	Name	Type	Default	Description
F:0	Package ID LSW [F:0]	RO		**Package ID LSW** Bits [15:0] of the Package ID.

10.3.12 PCS 10GBASE-T Status 1: Address 3.20

Bit	Name	Type	Default	Description
0	10GBASE-T PCS Block Lock	RO		**10GBASE-T PCS Block Lock** When set, this bit indicates that 10GBASE-T PCS framer has acquired frame synchronization and is locked. The interrupt for this bit is at 3.21.F (see Section 10.3.13). 0b = 10GBASE-T PCS framer is not locked. 1b = 10GBASE-T PCS framer is locked.
1	10 GbE High BER	RO		**10 GbE High Bit Error Rate** When set, this bit indicates a high BER is being seen at the PCS. The interrupt for this bit is at 3.21.E. The status bit for medium BER is found in Global Alarms 2: Address 1E.CC01 (Section 10.6.44). 0b = PCS is reporting a BER $< 10^{-4}$. 1b = PCS is reporting a BER $\geq 10^{-4}$.
B:2	Reserved	RSV		Reserved. Do not modify.
C	10 GbE Receive Link Status	RO		**10 GbE Receive Link Status** When set, this bit indicates that the 10 GbE receive link is functioning properly. This is a non-latching version of Bit 3.1.2 (see Section 10.3.2). The receive link is up when *Block Lock* status is asserted and *High BER* is de-asserted.) 0b = 10 GbE receive link down. 1b = 10 GbE receive link up.
F:D	Reserved	RSV		Reserved. Do not modify.

10.3.13 PCS 10GBASE-T Status 2: Address 3.21

Bit	Name	Type	Default	Description
7:0	Errored Block Counter [7:0]	SCT	0x0	**Errored Block Counter** Clear on read. In 10GBASE-T mode, this is taken from the state machine in Figure 55.16 in the 10GBASE-T specification.
D:8	Errored Frame Counter [5:0]	SCT	0x0	**Errored Frame Counter** Clear on read. In 10GBASE-T mode, this is taken from the state machine in Figure 55.14 in the 10GBASE-T specification. In 10GBASE-R mode, this is taken from 3.E
E	High BER Latched	LH		**High Bit Error Rate Latched** When set, this bit indicates a high BER is being seen at the PCS. This is the interrupt for Bit 3.20.1 (see Section 10.3.12). 0b = PCS is reporting a BER $< 10^{-4}$. 1b = PCS is reporting a BER $\geq 10^{-4}$.
F	PCS Block Lock Latched	LL		**PCS Block Lock Latched** When set, this bit indicates that 10GBASE-T PCS framer has acquired frame synchronization and is locked. This is the interrupt for Bit 3.20.0 (see Section 10.3.12). 0b = 10GBASE-T PCS framer is not locked. 1b = 10GBASE-T PCS framer is locked.

10.3.14 PCS TimeSync Capability: Address 3.1800

Bit	Name	Type	Default	Description
0	TimeSync Receive Path Data Delay	RO	0b	**TimeSync Receive Path Data Delay** 0b = PCS does not provide information on receive path data delay. 1b = PCS provides information on receive path data delay in registers 3.1805 through 3.1808.
1	TimeSync Transmit Path Data Delay	RO	0b	**TimeSync Transmit Path Data Delay** 0b = PCS does not provide information on transmit path data delay. 1b = PCS provides information on transmit path data delay in registers 3.1801 through 3.1804.
F:2	Reserved	RSV	0x0	Reserved. Do not modify.

10.3.15 PCS TimeSync Transmit Path Data Delay 1: Address 3.1801

Bit	Name	Type	Default	Description
F:0	Maximum PCS Transmit Path Data Delay LSW [F:0]	RO		**Maximum PCS Transmit Path Data Delay LSW** LSW of maximum PCS transmit delay in nanoseconds.

10.3.16 PCS TimeSync Transmit Path Data Delay 2: Address 3.1802

Bit	Name	Type	Default	Description
F:0	Maximum PCS Transmit Path Data Delay MSW [F:0]	RO		**Maximum PCS Transmit Path Data Delay MSW** MSW of maximum PCS transmit delay in nanoseconds

10.3.17 PCS TimeSync Transmit Path Data Delay 3: Address 3.1803

Bit	Name	Type	Default	Description
F:0	Minimum PCS Transmit Path Data Delay LSW [F:0]	RO		**Minimum PCS Transmit Path Data Delay LSW** LSW of minimum PCS transmit delay in nanoseconds.

10.3.18 PCS TimeSync Transmit Path Data Delay 4: Address 3.1804

Bit	Name	Type	Default	Description
F:0	Minimum PCS Transmit Path Data Delay MSW [F:0]	RO		**Minimum PCS Transmit Path Data Delay MSW** MSW of minimum PCS transmit delay in nanoseconds.

10.3.19 PCS TimeSync Receive Path Data Delay 1: Address 3.1805

Bit	Name	Type	Default	Description
F:0	Maximum PCS Receive Path Data Delay LSW [F:0]	RO		**Maximum PCS Receive Path Data Delay LSW** LSW of maximum PCS Receive delay in nanoseconds.

10.3.20 PCS TimeSync Receive Path Data Delay 2: Address 3.1806

Bit	Name	Type	Default	Description
F:0	Maximum PCS Receive Path Data Delay MSW [F:0]	RO		**Maximum PCS Receive Path Data Delay MSW** MSW of maximum PCS Receive delay in nanoseconds.

10.3.21 PCS TimeSync Receive Path Data Delay 3: Address 3.1807

Bit	Name	Type	Default	Description
F:0	Minimum PCS Receive Path Data Delay LSW [F:0]	RO		**Minimum PCS Receive Path Data Delay LSW** LSW of minimum PCS Receive delay in nanoseconds.

10.3.22 PCS TimeSync Receive Path Data Delay 4: Address 3.1808

Bit	Name	Type	Default	Description
F:0	Minimum PCS Receive Path Data Delay MSW [F:0]	RO		**Minimum PCS Receive Path Data Delay MSW** MSW of minimum PCS Receive delay in nanoseconds.

10.3.23 PCS Transmit Vendor Provisioning 1: Address 3.C400

Bit	Name	Type	Default	Description
0	PCS Tx Auxiliary Bit Value	RW PD	0b	**PCS Transmit Auxiliary Bit Value** The value that is set in the *Auxiliary* bit of the PCS transmission frame. ***Note:*** This bit is currently undefined in the 802.3an standard.
F:1	Reserved	RSV		Reserved. Do not modify.

10.3.24 PCS Transmit Reserved Vendor Provisioning 1: Address 3.C410

Bit	Name	Type	Default	Description
0	PCS IEEE Loopback Pass-through Disable	RW PD	0b	**PCS IEEE Loopback Pass-Through Disable** When set, this bit disables the output of the PHY when IEEE loopback is set. 1b = Disable data pass-through on IEEE loopback.
F:1	Reserved Transmit Provisioning 1 [F:1]	RW PD	0x0	**Reserved Transmit Provisioning 1** Reserved for future use.

10.3.25 PCS Standard Interrupt Mask 1: Address 3.D000

Bit	Name	Type	Default	Description
1:0	Reserved	RSV		Reserved. Do not modify.
2	PCS Receive Link Status Mask	RW PD	0b	**PCS Receive Link Status Mask** Mask for Bit 3.1.2 (see Section 10.3.2). 0b = Disable interrupt generation. 1b = Enable interrupt generation. ***Note:*** This bit also shows up as Bit 3.20.C, but only as a status bit.
9:3	Reserved	RSV		Reserved. Do not modify.
A	Rx LPI Received Mask	RW PD	0b	**Receive LPI Received Mask** Mask for Bit 3.1.A (see Section 10.3.2). 0b = Disable interrupt generation. 1b = Enable interrupt generation.
B	Tx LPI Received Mask	RW PD	0b	**Transmit LPI Received Mask** Mask for Bit 3.1.B (see Section 10.3.2). 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F:C	Reserved	RSV		Reserved. Do not modify.

10.3.26 PCS Standard Interrupt Mask 2: Address 3.D001

Bit	Name	Type	Default	Description
9:0	Reserved	RSV		Reserved. Do not modify.
A	Receive Fault Mask	RW PD	0b	**Receive Fault Mask** Mask for Bit 3.8.A (see Section 10.3.9). 0b = Disable interrupt generation. 1b = Enable interrupt generation.
B	Transmit Fault Mask	RW PD	0b	**Transmit Fault Mask** Mask for Bit 3.8.B (see Section 10.3.9). 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F:C	Reserved	RSV		Reserved. Do not modify.

10.3.27 PCS Standard Interrupt Mask 3: Address 3.D002

Bit	Name	Type	Default	Description
D:0	Reserved	RSV		Reserved. Do not modify.
E	10GBASE-T High BER Latched Mask	RW PD	0b	**10GBASE-T High Bit Error Rate Latched Mask** When set, this bit indicates a high BER is being seen at the PCS. This is the interrupt for Bit 3.21.E 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F	10GBASE-T PCS Block Lock Latched Mask	RW PD	0b	**10GBASE-T PCS Block Lock Latched Mask** When set, this bit indicates that 10GBASE-T PCS Framer has acquired frame synchronization and is locked. This is the interrupt for Bit 3.21.F 0b = Disable interrupt generation. 1b = Enable interrupt generation.

10.3.28 PCS Receive Vendor State 1: Address 3.E800

Bit	Name	Type	Default	Description
0	PCS Rx Current Value of Auxiliary Bit	RO		**PCS Rx Current Value of Auxiliary Bit** The current value of the PCS Rx auxiliary bit. This value has a maskable interrupt associated with it in 3.EC00.0.
F:1	Reserved	RSV		Reserved. Do not modify.

10.3.29 PCS Receive Vendor Alarms 1: Address 3.EC00

Bit	Name	Type	Default	Description
0	Change in Auxiliary Bit	LRF		**Change in Auxiliary Bit** This bit is set when a change is detected in the auxiliary bit. 1b = Indicates a change in the value of the auxiliary bit.
1	Reserved	RSV		Reserved. Do not modify.
2	EEE Rx LPI Received Latched High	LH		**EEE Rx LPI Received Latched High** Indicate LPI ordered-set is detected. 1b = Rx LPI has been detected
3	EEE Rx LPI Received Latched Low	LL		**EEE Rx LPI Received Latched Low** Indicate LPI ordered-set is detected. 1b = Rx LPI has been detected
4	EEE Rx LPI Alert	LH		**EEE Rx LPI Alert** 1b = Rx PCS received alert indication.
5	LDPC Consecutive Errored Frame Exceeded	LH		**LDPC Consecutive Errored Frame Exceeded** Indicates the consecutive LDPC errored frame has exceeded the threshold. 1b = Rx PCS LDPC consecutive errored frame threshold exceeded
6	EEE Rx LPI Active On	LH		**EEE Rx LPI Active On** 1b = EEE Rx LPI Active On
7	EEE Rx LPI Active Off	LL		**EEE Rx LPI Active Off** 1b = EEE Rx LPI Active Off

Bit	Name	Type	Default	Description
8	Invalid 65B Block	LH		**Invalid 65-Byte Block** This bit is set when an invalid 65-byte block (but without LDPC frame parity error) has been detected on the received LDPC frame. 1b = Invalid Rx 65-byte block received in PCS transmission frame.
9	Reserved	RSV		Reserved. Do not modify
A	LOF Detect			**LOF Detection Interrupt** 1b = RPL LOF detect
B	Local Fault Detect			**Local Fault Interrupt** 1b = RPL local fault detect
D:C	Reserved	RSV		Reserved. Do not modify.
E	LDPC Decode Failure	LH		**LDPC Decode Failure** This bit is set when the LDPC decoder fails to decode an LDPC block. 1b = LDPC decode failure.
F	CRC Error	LH		**CRC Error** This bit is set when a CRC-8 error is detected on the receive PCS frame. 1b = Rx CRC frame error.

10.3.30 PCS Receive Vendor Alarms 10: Address 3.EC09

Bit	Name	Type	Default	Description
0	PTP Ingress Packet Ready	RO		**PTP Ingress Packet Ready** Asserted when PTP packet is captured. Cleared when the buffer is empty. 1b = PTP Ingress packet ready.
1	PTP Ingress Time Stamp Ready	RO		**PTP Ingress Time Stamp Ready** Asserted when PTP packet timestamp is captured. Cleared when the timestamp buffer is empty. 1b = PTP Ingress packet time stamp ready.
2	PTP Ingress Packet Buffer Overflow Error	LH		**PTP Ingress Packet Buffer Overflow Error** Asserted when PTP packet is not captured because the buffer is full. 1b = PTP Ingress packet buffer overflow error detected.
3	PTP Ingress Packet Correction Field Error	LL		**PTP Ingress Packet Correction Field Error** Asserted when the packet length is too long for the Correction field to be updated due to the ingress timestamp appended at the end of packet. This should not happen for IEEE1588v2 compliant packets, but if it does, the Correction field is not changed. 1b = PTP Ingress packet correction field error.
4	PTP Ingress Packet Buffer Parity Error	LH		**PTP Ingress Packet Buffer Parity Error** Asserted when PTP packet pipeline detects FIFO error. 1b = PTP Ingress packet buffer parity error detected.
5	PTP Ingress Time Stamp Buffer Parity Error	LH		**PTP Ingress Time Stamp Buffer Parity Error** Asserted when PTP packet pipeline detects parity error. 1b = PTP Ingress time stamp buffer parity error detected.
6	PTP Ingress Packet Pipeline Parity Error	LH		**PTP Ingress Packet Pipeline Parity Error** Asserted when PTP timestamp buffer detects parity error. 1b = PTP Ingress packet pipeline parity error detected.
7	PTP Ingress Packet Pipeline FIFO Error	LL		**PTP Ingress Packet Pipeline FIFO Error** Asserted when PTP packet buffer detects parity error. 1b = PTP Ingress packet pipeline FIFO error detected.
8	PTP Ingress Packet Received	LH		**PTP Ingress Packet Received** Asserted when a valid PTP packet is received. This can be used as the valid signal for the received PTP packet information. 1b = PTP Ingress packet received.

Bit	Name	Type	Default	Description
9	PTP Ingress Packet Remove Error	LH		**PTP Ingress Packet Remove Error** Asserted when the packet length is too long for the UDP checksum to be updated during removal of the timestamp appended at the end of packet. When this happens, the UDP checksum is set to all-zeros. 1b = PTP Ingress packet remove error.
A	PTP Ingress Packet Ready FIFO Parity Error	LH		**PTP Ingress Packet Ready FIFO Parity Error** Asserted when PTP packet ready FIFO detects parity error. 1b = PTP Ingress packet ready FIFO parity error.
B	PTP Ingress Packet Ready FIFO Error	LH		**PTP Ingress Packet Ready FIFO Error** Asserted when PTP packet ready FIFO detects FIFO error 1b = PTP Ingress packet ready FIFO error.
F:C	Reserved	RSV		Reserved. Do not modify.

10.3.31 PCS Vendor Global Interrupt Flags 1: Address 3.FC00

Bit	Name	Type	Default	Description
5:0	Reserved	RSV		Reserved. Do not modify
6	Vendor Specific Rx Alarms 1 Interrupt	RO		**Vendor Specific Receive Alarms 1 Interrupt** An interrupt was generated from status register PCS Receive Vendor Alarms 1: Address 3.EC00 and the corresponding mask register.
A:7	Reserved	RSV		Reserved. Do not modify
B	Vendor Specific Tx Alarms 1 Interrupt	RO		**Vendor Specific Transmit Alarms 1 Interrupt** An interrupt was generated from status register and the corresponding mask register.
C	Reserved	RSV		Reserved. Do not modify.
D	Standard Alarm 3 Interrupt	RO		**Standard Alarm 3 Interrupt** An interrupt was generated from status register PCS 10GBASE-T Status 2: Address 3.21 and the corresponding mask register PCS Standard Interrupt Mask 3: Address 3.D002. 1b = Interrupt in standard alarms 3.
E	Standard Alarm 2 Interrupt	RO		**Standard Alarm 2 Interrupt** An interrupt was generated from status register PCS Standard Status 2: Address 3.8 and the corresponding mask register PCS Standard Interrupt Mask 2: Address 3.D001. 1b = Interrupt in standard alarms 2.
F	Standard Alarm 1 Interrupt	RO		**Standard Alarm 1 Interrupt** An interrupt was generated from status register PCS Standard Status 1: Address 3.1 and the corresponding mask register PCS Standard Interrupt Mask 1: Address 3.D000. 1b = Interrupt in standard alarms 1.

10.3.32 PCS Vendor Global Interrupt Flags 3: Address 3.FC02

Bit	Name	Type	Default	Description
5:0	Reserved	RSV		Reserved. Do not modify
6	Vendor Specific Rx Alarms 6 Interrupt	RO		**Vendor Specific Receive Alarms 6 Interrupt** An interrupt was generated from status register and the corresponding mask register. 1b = Interrupt in vendor specific Rx alarms 6
7	Vendor Specific Rx Alarms 7 Interrupt	RO		**Vendor Specific Receive Alarms 7 Interrupt** An interrupt was generated from status register and the corresponding mask register. 1b = Interrupt in vendor specific Rx alarms 7
F:8	Reserved	RSV		Reserved. Do not modify

10.4 Auto-Negotiation Registers

10.4.1 Auto-Negotiation Standard Control 1: Address 7.0

Bit	Name	Type	Default	Description
8:0	Reserved	RSV		Reserved. Do not modify.
9	Restart Auto-Negotiation	RW SC	0b	**Restart Auto-Negotiation** 0b = Normal operation. 1b = Restart auto-negotiation process.
B:A	Reserved	RSV		Reserved. Do not modify.
C	Auto-Negotiation Enable	RW PD	1b	**Auto-Negotiation Enable** When enabled, auto-negotiation determines the link speed. When disabled, the link is forced to its down state. 0b = Disable auto-negotiation. 1b = Enable auto-negotiation.
D	Extended Next Page Control	RW PD	1b	**Extended Next Page Control** This bit is OR'ed with Bit 7.10.C. 0b = Extended next pages are disabled. 1b = Extended next pages are enabled.
F:E	Reserved	RSV		Reserved. Do not modify.

10.4.2 Auto-Negotiation Standard Status 1: Address 7.1

Bit	Name	Type	Default	Description
0	Link Partner Auto-Negotiation Ability	RO		**Link Partner Auto-Negotiation Ability** 0b = Link partner not able to perform auto-negotiation. 1b = Link partner able to perform auto-negotiation.
1	Reserved	RSV		Reserved. Do not modify.
2	Link Status	LL		**Link Status** This bit is a duplicate of the *PMA Receive Link Status* bit (Bit [2]) of the PMA Standard Status 1: Address 1.1 register. 0b = Link lost since last read. 1b = Link is up. ***Note:*** This is latching low, so it can only be used to detect link drops and not the current status of the link without performing back-to-back reads.
3	Auto-Negotiation Ability	ROS	1b	**Auto-Negotiation Ability** Always set as 1b because the local device has auto-negotiation ability. 0b = PHY is not able to perform auto-negotiation. 1b = PHY is able to perform auto-negotiation.
4	Remote Fault	LH		**Remote Fault** This indicates that the remote PHY has a fault. 0b = No remote fault condition detected. 1b = Remote fault condition detected.
5	Auto-Negotiation Complete	RO		**Auto-Negotiation Complete** This indicates the status of the auto-negotiation receive link. 0b = Auto-negotiation in process. 1b = Auto-negotiation complete.

Bit	Name	Type	Default	Description
6	Page Received	LH		**Page Received** Indicates that a page has been received If it is a regular page, it is placed in 7.13 ->7.15. If it is an extended next page it is placed in registers 7.19 -> 7.1B 0b = A page has not been received. 1b = A page has been received.
7	Extended Next Page Status	RO		**Extended Next Page Status** Indicates that both the local device and the link partner have indicated support for extended next page. 0b = Extended next page is not used. 1b = Extended next page is used.
8	Reserved	RSV		Reserved. Do not modify.
9	Parallel Detection Fault	LH		**Parallel Detection Fault** 0b = A fault has not been detected via the parallel detection function. 1b = A fault has been detected via the parallel detection function.
F:A	Reserved	RSV		Reserved. Do not modify.

10.4.3 Auto-Negotiation Standard Device Identifier 1: Address 7.2

Bit	Name	Type	Default	Description
F:0	OUI LSB	RO	0x154	**OUI LSB** Bits [31:16] of the Device ID. OUI[18:3]

10.4.4 Auto-Negotiation Standard Device Identifier 2: Address 7.3

Bit	Name	Type	Default	Description
3:0	Manufacturer's Revision Number	RO	0x0	**Manufacturer's Revision Number** 4 bits containing the manufacturer's revision number.
9:4	Manufacturer's Model Number	RO	0x22	**Manufacturer's Model Number** 6 bits containing the manufacturer's part number 0x22 for the X550-AT2 or X550-BT2.
F:A	OUI MSB	RO	0x0	**OUI MSB** Bits [15:10] of Device ID. OUI[24:19]

10.4.5 Auto-Negotiation Standard Devices in Package 1: Address 7.5

Bit	Name	Type	Default	Description
0	Clause 22 Registers Present	ROS	0b	**Clause 22 Registers Present** 0b = Clause 22 registers are not present in package. 1b = Clause 22 registers are present in package. This bit is always set to 0b, as there are no Clause 22 registers in the X550.
1	PMA Present	ROS	1b	**PMA Present** 0b = PMA is not present. 1b = PMA is present in package. This bit is always set to 1b, as there is PMA functionality in the X550.
2	WIS Present	ROS	0b	**WIS Present** 0b = WIS is not present in package. 1b = WIS is present in package. This bit is always set to 0b, as there is no WIS functionality in the X550.
3	PCS Present	ROS	1b	**PCS Present** 0b = PCS is not present in package. 1b = PCS is present in package. This bit is always set to 1b, as there is PCS functionality in the X550.
4	PHY XS Present	ROS	0b	**PHY XS Present** 0b = PHY XS is not present in package. 1b = PHY XS is present in package. This bit is always set to 0b, as there is no PHY XS interface in the X550.
5	DTE XS Present	ROS	0b	**DTE XS Present** 0b = DTE XS is not present in package. 1b = DTE XS is present in package. This bit is always set to 0b, as there is no DTE XAUI interface in the X550.
6	TC Present	ROS	0b	**TC Present** 0b = TC is not present in package. 1b = TC is present in package. This bit is always set to 0b, as there is no TC functionality in the X550.
7	Auto-Negotiation Present	ROS	1b	**Auto-Negotiation Present** 0b = Auto-negotiation is not present in package. 1b = Auto-negotiation is present in package. This bit is always set to 1b, as there is auto-negotiation in the X550.
F:8	Reserved	RSV		Reserved. Do not modify

10.4.6 Auto-Negotiation Standard Devices in Package 2: Address 7.6

Bit	Name	Type	Default	Description
C:0	Reserved	RSV		Reserved. Do not modify
D	Clause 22 Extension Present	ROS	1b	**Clause 22 Extension Present** 0b = Clause 22 Extension is not present in package. 1b = Clause 22 Extension is present in package. This is always set to 1b, as the X550 uses this device for the GbE registers.
E	Vendor Specific Device #1 Present	ROS	1b	**Vendor Specific Device #1 Present** 0b = Device #1 is not present in package. 1b = Device #1 is present in package. This is always set to 1b, as the X550 uses this device for the global control registers.
F	Vendor Specific Device #2 Present	ROS	1b	**Vendor Specific Device #2 Present** 0b = Device #2 is not present in package. 1b = Device #2 is present in package. This is always set to 1b, as the X550 uses this device for the DSP PMA registers.

10.4.7 Auto-Negotiation Standard Status 2: Address 7.8

Bit	Name	Type	Default	Description
D:0	Reserved	RSV		Reserved. Do not modify
F:E	Device Present [1:0]	ROS	10b	**Device Present** 00b = No device at this address. 01b = No device at this address. 10b = Device present at this address. 11b = No device at this address. This field is always set to 10b, as auto-negotiation resides in the X550.

10.4.8 Auto-Negotiation Standard Package Identifier 1: Address 7.E

Bit	Name	Type	Default	Description
F:0	Package ID MSW [1F:10]	RO		**Package ID MSW** Bits [31:16] of the Package ID.

10.4.9 Auto-Negotiation Standard Package Identifier 2: Address 7.F

Bit	Name	Type	Default	Description
F:0	Package ID FSW [1F:10]	RO		**Package ID FSW** Bits [15:0] of the Package ID.

10.4.10 Auto-Negotiation Advertisement Register: Address 7.10

Note: This register is used in conjunction with Bit 7.C400.5 to hard-provision the auto-negotiation base page that is sent.

Bit	Name	Type	Default	Description
4:0	Selector Field [4:0]	RW PD	0x01	**Selector Field** This defines the device compatibility: 0x00 = Reserved 0x01 = IEEE 802.3 0x02 = IEEE 802.9 ISLAN-16T 0x03 = IEEE 802.5 0x04 = IEEE 1394 All other values are reserved. This field should always be set to 0x01 because the PHY is only capable of handling 802.3 Ethernet.
B:5	Technology Ability Field [6:0]	RW PD	0x00	**Technology Ability Field** This bit contains information indicating supported technologies defined in Annex 28B.2 and Annex 28D. Multiple technologies might be advertised in the link code word. A device must support the data service ability for a technology it advertises. Bit [0] = Reserved Bit [1] = Reserved Bit [2] = 100BASE-TX Bit [3] = 100BASE-TX full duplex Bit [4] = 100BASE-T4 Bit [5] = PAUSE operation for full duplex links Bit [6] = Asymmetric PAUSE operation for full duplex links ***Note:*** Technology NOT supported by the X550. Bits must be set to 0x00.
C	Extended Next Page Ability	RW PD	1b	**Extended Next Page Ability** This bit indicates that the local device supports transmission of extended next pages when set to 1b and indicates that the local device does not support extended next pages when set to 0b. 0b = Not capable of extended next pages. 1b = Extended next page capable.
D	Advertisement Remote Fault	RW	0b	**Advertisement Remote Fault** This bit provides a standard transport mechanism for the transmission of simple fault information. When the *Remote Fault (RF)* bit in the received base link code word is set to 1b, the *RF* bit is set to 1b. 1b = Remote fault ***Note:*** Not supported feature. The bit must be maintained as 0b.
E	Reserved	RSV		Reserved. Do not modify.
F	Next Page Ability	RW PD	1b	**Next Page Ability** If the X550 implements next page ability and needs to engage in a next page exchange, it must set the *Next Page (NP)* bit to 1b. The X550 might implement next page ability and choose not to engage in a next page exchange by setting the *NP* bit to 0b. 1b = Next page ability

10.4.11 Auto-Negotiation Link Partner Base Page Ability Register: Address 7.13

Bit	Name	Type	Default	Description
4:0	Link Partner Selector Field [4:0]	RO		**Link Partner Selector Field** This field encodes 32 possible messages defined in Annex 28A. Combinations not specified are reserved for future use and must not be transmitted. This defines the device compatibility: 0x00 = Reserved 0x01 = IEEE 802.3 0x02 = IEEE 802.9 ISLAN-16T 0x03 = IEEE 802.5 0x04 = IEEE 1394 All other values are reserved.
B:5	Link Partner Technology Ability Field [6:0]	RO		**Link Partner Technology Ability Field** This field contains information indicating supported technologies defined in Annex 28B.2 and Annex 28D. Multiple technologies can be advertised in the link code word. The X550 must support the data service ability for a technology it advertises. The arbitration function determines the common mode of operation shared by a link partner and resolves the multiple common modes. Bit [0] = Reserved Bit [1] = Reserved Bit [2] = 100BASE-TX Bit [3] = 100BASE-TX full duplex Bit [4] = 100BASE-T4 Bit [5] = PAUSE operation for full duplex links Bit [6] = Asymmetric PAUSE operation for full duplex links
C	Link Partner Extended Next Page Ability	RO		**Link Partner Extended Next Page Ability** This field indicates that the link partner has indicated support for the extended next page when set to 1b. When set to 0b, the link partner does not support extended next page. 0b = Not capable of extended next pages. 1b = Extended next page capable.
D	Link Partner Remote Fault	RO		**Link Partner Remote Fault** This bit provides a standard transport mechanism for transmitting simple fault information. When the *RF* bit in the received base link code word is set to 1b, the *RF* bit is set to 1b. 1b = Remote fault
E	Link Partner Base Page Acknowledge	RO		**Link Partner Base Page Acknowledge** The Acknowledge (ACK) is used by the auto-negotiation function to indicate that a device has successfully received its link partner's link code word. 1b = Acknowledge
F	Link Partner Next Page Ability	RO		**Link Partner Next Page Ability** If next page ability is not supported, the *NP* bit must always be set to 0b. If the X550 implements next page ability and needs to engage in a next page exchange, it must set the *NP* bit to 1b. 0b = Next page ability not supported or not engaged. 1b = Next page ability.

10.4.12 Auto-Negotiation Extended Next Page Transmit Register: Address 7.16

Note: This register is used in conjunction with Bit 7.C400.5 and registers 7.17 and 7.18 to hard-provision the auto-negotiation extended next page that is sent. By using this register, it is possible to hard code the 1 GbE and 10 GbE capabilities.

Bit	Name	Type	Default	Description
A:0	Message Code Field [A:0]	RW PD	0x001	**Message Code Field** Interpreted as message code (see 802.3 Appendix 28C) if the *Message Page* bit is set to 1b (7.16:1). Otherwise, interpreted as an unformatted code field. [A:0] = Message code field: 0x0 = Reserved. 0x1 = Null message. 0x2 = Reserved expansion message. 0x3 = Reserved expansion message. 0x4 = Remote fault details message. 0x5 = OUI message. 0x6 = PHY ID message. 0x7 = 100BASE-T2 message. 0x8 = 1000BASE-T message. All other values are reserved.
B	Toggle	RO		**Toggle** Value of toggle bit. Set to opposite of corresponding bit in previous page.
C	Acknowledge 2	RW	0b	**Acknowledge 2** This field is used by the next page function to indicate that a device has the ability to comply with the message. 0b = Cannot comply with corresponding message. 1b = Complies with corresponding message.
D	Message Page	RW	0b	**Message Page** Message page is used by the next page function to differentiate a message page from an unformatted page. 0b = Unformatted page. 1b = Message page.
E	Reserved	RSV		Reserved. Do not modify.
F	Next Page	RW	0b	**Next Page** Next page is used by the next page function to indicate whether this is the last next page to be transmitted. 0b = Last page. 1b = Additional next page follows.

10.4.13 Auto-Negotiation Extended Next Page Unformatted Code Register 1: Address 7.17

Bit	Name	Type	Default	Description
F:0	Unformatted Code Field 1 [1F:10]	RW PD	0x0	**Unformatted Code Field 1**

10.4.14 Auto-Negotiation Extended Next Page Unformatted Code Register 2: Address 7.18

Bit	Name	Type	Default	Description
F:0	Unformatted Code Field 2 [2F:20]	RW PD	0x0	**Unformatted Code Field 2**

10.4.15 Auto-Negotiation Link Partner Extended Next Page Ability Register: Address 7.19

Note: This register, along with 7.1A and 7.1B, are used to store the received next pages. If an extended next page is used, it is stored in 7.19:7.1B.

Bit	Name	Type	Default	Description
A:0	Link Partner Message Code Field [A:0]	RO		**Link Partner Message Code Field** Interpreted as message code (see 802.3 Appendix 28C) if the *Message Page* bit is set to 1b (7.16:1). Otherwise, interpreted as an unformatted code field. [A:0] = Message code field: 0x0 = Reserved. 0x1 = Null message. 0x2 = Reserved expansion message. 0x3 = Reserved expansion message. 0x4 = Remote fault details message. 0x5 = OUI message. 0x6 = PHY ID message. 0x7 = 100BASE-T2 message. 0x8 = 1000BASE-T message. All other values are reserved.
B	Link Partner Toggle	RO		**Link Partner Toggle** Set opposite of the corresponding bit in the previous page. Value of link partner's toggle bit.
C	Link Partner Acknowledge 2	RO		**Link Partner Acknowledge 2** 0b = Link partner cannot comply with the current next page. 1b = Link partner acknowledges that they can comply with the current next page.
D	Link Partner Message Page	RO		**Link Partner Message Page** 0b = Unformatted page. 1b = Message page.
E	Link Partner Extended Next Page Acknowledge	RO		**Link Partner Extended Next Page Acknowledge** Acknowledge is used by the auto-negotiation function to indicate that a device has successfully received its link partners link code word. 1b = Link partner acknowledges receipt of corresponding page.
F	Link Partner Next Page	RO		**Link Partner Next Page** 1b = Next page ability.

10.4.16 Auto-Negotiation Link Partner Extended Next Page Unformatted Code Register 1: Address 7.1A

Bit	Name	Type	Default	Description
F:0	Link Partner Unformatted Code Field 1 [F:0]	RO		**Link Partner Unformatted Code Field 1 [15:0]**

10.4.17 Auto-Negotiation Link Partner Extended Next Page Unformatted Code Register 2: Address 7.1B

Bit	Name	Type	Default	Description
F:0	Link Partner Unformatted Code Field 2 [F:0]	RO		**Link Partner Unformatted Code Field 2 [15:0]**

10.4.18 Auto-Negotiation 10GBASE-T Control Register: Address 7.20

Bit	Name	Type	Default	Description
0	LD Loop Timing Ability	RW PD	1b	**LD Loop Timing Ability** 0b = Do not advertise PHY as capable of loop timing. 1b = Advertise PHY as capable of loop timing
1	LD Fast Retrain Ability	RW PD	0b	**LD Fast Retrain Ability** 0b = Do not advertise PHY as 10GBASE-T fast retrain capable 1b = Advertise PHY as 10GBASE-T fast retrain capable
2	LD PMA Training Reset Request	RW PD	0b	**LD PMA Training Reset Request** 0b = Local device requests that Link Partner run PMA training PRBS continuously. 1b = Local device requests that Link Partner reset PMA training PRBS every frame.
B:3	Reserved	RSV		Reserved. Do not modify.
C	10GBASE-T Ability	RW PD	1b	**10GBASE-T Ability** 0b = Do not advertise PHY as 10GBASE-T capable. 1b = Advertise PHY as 10GBASE-T capable.
D	Port Type	RW PD	0b	**Port Type** 0b = Single port device. 1b = Multiport device.
E	Master/Slave Configuration	RW PD	0b	**Master/Slave Configuration** 0b = Slave. 1b = Master.
F	Master/Slave Manual Configuration Enable	RW PD	0b	**Master/Slave Manual Configuration Enable** 0b = Disable master/slave manual configuration. 1b = Enable master/slave manual configuration.

10.4.19 Auto-Negotiation 10GBASE-T Status Register: Address 7.21

Bit	Name	Type	Default	Description
0	Reserved	RSV		Reserved. Do not modify.
1	Link Partner Fast Retrain Ability	RO		**Link Partner Fast Retrain Ability** 0b = Link partner is not capable of 10GBASE-T fast retrain. 1b = Link partner is capable of 10GBASE-T fast retrain
8:2	Reserved	RSV		Reserved. Do not modify.
9	Link Partner Training Reset Request	RO		**Link Partner Training Reset Request** 0b = Link partner has requested that PMA PRBS training run continuously. 1b = Link partner has requested that PMA PRBS training be reset every frame.
A	Link Partner Loop Timing Ability	RO		**Link Partner Loop Timing Ability** 0b = Link partner is not capable of loop timing. 1b = Link partner is capable of loop timing.
B	Link Partner 10GBASE-T Ability	RO		**Link Partner 10GBASE-T Ability** This bit is only valid when the *Page Received* Bit 7.1.6 is set to 1b. 0b = Link partner is not 10GBASE-T capable. 1b = Link partner is 10GBASE-T capable.
C	Remote Receiver Status	RO		**Remote Receiver Status** Set by the micro controller. 0b = Remote receiver not operational. 1b = Remote receiver operational.
D	Local Receiver Status	RO		**Local Receiver Status** Set by the micro controller. 0b = Local receiver not operational. 1b = Local receiver operational.
E	Master/Slave Configuration Resolution	RO		**Master/Slave Configuration Resolution** 0b = Local PHY resolved to slave. 1b = Local PHY resolved to master.
F	Master/Slave Configuration Fault	LH		**Master/Slave Configuration Fault** 1b = Master/slave configuration fault.

10.4.20 Auto-Negotiation Vendor Provisioning 1: Address 7.C400

Bit	Name	Type	Default	Description
3:0	Retry Attempts Before Downshift [3:0]	RW PD	0x4	**Retry Attempts Before Downshift** Number of retry attempts before downshift. If automatic downshifting is enabled, this is the number of retry attempts the PHY makes to connect at the maximum mutually acceptable rate, before removing this rate from the list and trying the next lower rate.
4	Automatic Downshift Enable	RW PD	1b	**Automatic Downshift Enable** 0b = Manual downshift. 1b = Enable automatic downshift.

Bit	Name	Type	Default	Description
5	User Provided Auto-Negotiation Data	RW PD	0b	**User Provided Auto-Negotiation Data** If this bit is set, the PHY attempts to use the user-provided auto-negotiation words. If there is a mismatch (such as a legacy 1GBASE-T device attempting to connect), the PHY attempts to construct a new set of auto-negotiation words from the data provided in these words. Otherwise, the PHY constructs the correct auto-negotiation words based on the provisioned values. 0b = Construct the correct auto-negotiation words based on the register settings of 7.10, 7.20, and 7.C400. 1b = User provides the next page or extended next page data directly (7.16:7.18), and the configuration information in 7.20 and 7.C400 is ignored.
6	Exchange PHY ID Information	RW PD	1b	Exchange PHY ID Information 1b = Exchange PHY ID information.
9:7	Reserved	RSV		Reserved. Do not modify.
A	2.5G	RW PD	0b	**2.5G** 0b = Do not advertise PHY as supporting 2.5 GbE. 1b = Advertise PHY as supporting 2.5 GbE.
B	5G	RW PD	0b	**5G** 0b = Do not advertise PHY as supporting 5 GbE. 1b = Advertise PHY as supporting 5 GbE.
C	AQRate Downshift Capability	RW PD	0b	**NBASE-T Downshift Capability** 0b = Do not allow NBASE-T down-shifting. 1b = Allow NBASE-T down-shifting.
E:D	Reserved	RSV		Reserved. Do not modify.
F	1000BASE-T Full Duplex Ability	RW PD	0b	**1000BASE-T Full Duplex Ability** 0b = Do not advertise PHY as 1000BASE-T full duplex capable. 1b = Advertise PHY as 1000BASE-T full duplex capable.

10.4.21 Auto-Negotiation Reserved Vendor Provisioning 1: Address 7.C410

Bit	Name	Type	Default	Description
1:0	MDI / MDI-X Control [1:0]	RW PD	00b	**MDI/MDI-X Control** These bits are used to force a manual MDI or MDI-X configuration. 00b = Automatic MDI/MDI-X operation. 01b = Manual MDI. 10b = Manual MDI-X. 11b = Reserved.
5:2	Extra Page Count [3:0]	RW PD	0x4	**Extra Page Count** Number of extra pages to send at the end of the auto-negotiation sequence when the link partner is a legacy GbE PHY. Intervals between pages for GbE PHYs might be much longer. If this is the case, the link partner might still be in auto-negotiation when the X550 starts it's training. This might confuse the link partner MDI/MDI-X state machine. Sending extra pages seems to correct this problem.
6	WoL Enable	RW PD	0b	**WoL Enable** Setting this bit enables WoL operation. In this state, power is minimized by turning off all interfaces except the MDI receive path and the MDIO interface.

Bit	Name	Type	Default	Description
7	WoL Mode	RW PD	0b	**WoL Mode** This bit indicates whether the X550 goes into 100BASE-TX or 1000BASE-T WoL operation. 0b = 100BASE-TX 1b = 1000BASE-T
A:8	Semi-Cross Link Attempt Period [2:0]	RW PD	0x0	**Semi-Cross Link Attempt Period** Number of failed link attempts before trying semi cross. Set to 0x0 to disable semi-cross. Set to 0x7 to always use semi-cross.
F:B	Reserved	RSV		Reserved. Do not modify.

10.4.22 Auto-Negotiation Reserved Vendor Provisioning 2: Address 7.C411

Bit	Name	Type	Default	Description
A:0	Reserved	RSV		Reserved. Do not modify.
B	Auto-Negotiation Timeout Mode	RW PD	0b	**Auto-Negotiation Timeout Mode** 0b = Enable timeout for all auto-negotiation behavior. 1b = Enable timeout only if the following are enabled: LPLU, downshifting. The timeout behavior can be described as follows: Bits F:C = non-zero, Bit B = 0: Timeout exists for all auto-negotiation behavior, including LPLU and if downshift is enabled. Bits F:C = 0, Bit B = 0: No timeout enabled Bits F:C = non-zero, Bit B = 1: If we are in LPLU or downshift is enabled, timeout exists. If we are not in LPLU or downshift is not enabled, there is no timeout. Bits F:C = 0, Bit B = 1: If we are in LPLU or downshift is enabled, there is no timeout. If we are not in LPLU or downshift is not enabled, there is no timeout. In other words, there is no timeout.
F:C	Auto-Negotiation Timeout [3:0]	RW PD	0x0	**Auto-Negotiation Timeout** The length of time in (seconds*2) before auto-negotiation restarts. A value of zero indicates there is no timeout. These bits control the use of auto-negotiation timeout watchdog during the Arbit state machine.

10.4.23 Auto-Negotiation Vendor Status 1: Address 7.C800

Bit	Name	Type	Default	Description
0	Connect Type	RO		**Connect Type** This field is used in conjunction with the *Connection State* field in the Auto-Negotiation Reserved Vendor Status 1: Address 7.C810 to indicate the duplex method the PHY is connected or attempting to connect at. The duplex method the PHY connected or attempting to connect at: 0b = Reserved 1b = Full duplex
3:1	Connect Rate [2:0]	RO		**Connect Rate** This field is used in conjunction with the *Connection State* field in the Auto-Negotiation Reserved Vendor Status 1: Address 7.C810 to indicate the rate the PHY is connected or attempting to connect at. The rate the PHY connected or attempting to connect at: 000b = Reserved 001b = 100BASE-TX 010b = 1000BASE-T 011b = 10GBASE-T 100b = 2.5 GbE 101b = 5 GbE All other values are reserved.
F:4	Reserved	RSV		Reserved. Do not modify.

10.4.24 Auto-Negotiation Reserved Vendor Status 1: Address 7.C810

Bit	Name	Type	Default	Description
0	Transmit PAUSE Resolution	RO		**Transmit PAUSE Resolution** PAUSE resolution from 28B-3 0b = Transmit PAUSE disabled. 1b = Transmit PAUSE enabled.
1	Receive PAUSE Resolution	RO		Receive PAUSE Resolution PAUSE resolution from 28B-3 0b = Receive PAUSE disabled. 1b = Receive PAUSE enabled.
6:2	Reserved Status 1 [6:2]	RO		**Reserved Status 1** Reserved for future use.
7	Duplicate Link Partner Auto-Negotiation Ability	RO		**Duplicate Link Partner Auto-Negotiation Ability** The link partner is capable of auto-negotiation. This is a duplicate of the bit at 07.0001.0.
8	MDI/MDI-X	RO		**MDI/MDI-X** When auto-negotiation completes, this register indicates whether the connection was made as an MDI or MDI-X connection. 0b = MDI 1b = MDI-X

Bit	Name	Type	Default	Description
D:9	Connection State [4:0]	RO		**Connection State** This field is used in conjunction with the *Connect Rate* and *Connect Type* fields in the Auto-Negotiation Vendor Status 1: Address 7.C800 register to indicate the current state the PHY is in. The current state of the connection: 0x00 = Inactive (such as high-impedance). 0x01 = Cable diagnostics. 0x02 = Auto-negotiation. 0x03 = Training (10 GbE, 5 GbE, 2.5 GbE and 1 GbE only). 0x04 = Connected. 0x05 = Fail (auto-negotiation break link). 0x06 = Test mode. 0x07 = Loopback mode. 0x08 = Low Power mode. 0x09 = Connected WoL mode. 0x0A = System calibrating. 0x0B = Cable disconnected. 0x1C = MAC requested PHY reset. 0x1D = MAC requested PHY low-power mode. 0x1E = MAC requested PHY disable mode. 0x1F = MAC requested PHY resume normal operation. All other values are reserved.
E	Device Present	RO		**Device Present** If true, a far-end Ethernet device exists because valid link pulses have been detected in the most recent auto-negotiation session, or a valid Ethernet connection has been established. If false, no connection is established, and the most recent attempt at auto-negotiation failed to detect any valid link pulses.Specifically, when MDI/MDI-X resolution has completed, this bit is true. This bit is set false before entering auto-negotiation. 0b = No far-end Ethernet device detected. 1b = Far-end Ethernet device present.
F	Energy On Line	RO		**Energy On Line** This bit is used to indicate that the PHY has detected energy on the line. Specifically, when MDI/MDI-X resolution has completed, this bit is true. This bit is set false before entering auto-negotiation. 0b = No energy detected on line. 1b = Energy detected on line.

10.4.25 Auto-Negotiation Reserved Vendor Status 2: Address 7.C811

Bit	Name	Type	Default	Description
F:0	Auto-Negotiation Attempts [F:0]	RO		**Auto-Negotiation Attempts** The number of auto-negotiation attempts since the last successful connection (or power-up). This is a rolling counter (reverts to zero at saturation). It is cleared at reset or after a successful connection completes.

10.4.26 Auto-Negotiation Reserved Vendor Status 3: Address 7.C812

Bit	Name	Type	Default	Description
E:0	Reserved State 3 [E:0]	RO		**Reserved State 3** Reserved for future use.
F	Link Pulse Detected Status	RO		**Link Pulse Detected Status** 0b = Link pulse not detected. 1b = Link pulse detected.

10.4.27 Auto-Negotiation Reserved Vendor Status 4: Address 7.C813

Bit	Name	Type	Default	Description
F:0	Auto-Negotiation Restarts Handled [F:0]	RO		**Auto-Negotiation Restarts Handled** The number of line-side auto-negotiation restart commands received. This is a rolling counter (reverts to zero at saturation). It is cleared upon reset.

10.4.28 Auto-Negotiation Reserved Vendor Status 5: Address 7.C814

Bit	Name	Type	Default	Description
F:0	Auto-Negotiation Attempts Since Reset [F:0]	RO		**Auto-Negotiation Attempts Since Reset** The number of auto-negotiation attempts since the last PHY reset (or power-up). This is a rolling counter (reverts to zero at saturation). It is cleared upon reset.

10.4.29 Auto-Negotiation Transmit Vendor Alarms 1: Address 7.CC00

Bit	Name	Type	Default	Description
0	Connection State Change	LH		**Connection State Change** This interrupt indicates a change in the *Connection State* [D:B] in Auto-Negotiation Reserved Vendor Status 1: Address 7.C810 register. 1b = The connection state has changed. ***Note:*** This indicates any state change versus 7.CC01.0, which indicates a connect or disconnect event.
1	Automatic Downshift	LH		**Automatic Downshift** 1b = Automatic downshift has occurred.
2	Auto-Negotiation Completed for Supported Rate	LH		**Auto-Negotiation Completed For Supported Rate** 1b = Auto-negotiation has completed successfully for a rate that is supported by the X550.

Bit	Name	Type	Default	Description
3	Auto-Negotiation Completed For Non-Supported Rate	LH		**Auto-Negotiation Completed For Non-Supported Rate** This means that the X550 has completed auto-negotiation and was unable to agree on a rate that both could operate at. 1b = Auto-negotiation has completed for a rate that is not supported by the X550. ***Note:*** This indication should be ignored in the case of master/slave resolution fault.
F:4	Reserved	RSV		Reserved. Do not modify.

10.4.30 Auto-Negotiation Transmit Vendor Alarms 2: Address 7.CC01

Bit	Name	Type	Default	Description
0	Link Connect/Disconnect	LH		**Link Connect/Disconnect** This indicates whether the link has achieved a connect state or was in a connect state and disconnected. 1b = MDI link has either connected or disconnected.
E:1	Reserved Vendor Alarms 2 [D:0]	LH		**Reserved Vendor Alarms 2** Reserved for future use.
F	Link Pulse Detect	LH		**Link Pulse Detect** 1b = Link pulse detected.

10.4.31 Auto-Negotiation Standard Interrupt Mask 1: Address 7.D000

Bit	Name	Type	Default	Description
1:0	Reserved	RSV		Reserved. Do not modify.
2	Link Status Mask	RW PD	0b	**Link Status Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
3	Reserved	RSV		Reserved. Do not modify.
4	Remote Fault Mask	RW PD	0b	**Remote Fault Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
5	Reserved	RSV		Reserved. Do not modify.
6	Extended Next Page Received Mask	RW PD	0b	**Extended Next Page Received Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
8:7	Reserved	RSV		Reserved. Do not modify.
9	Parallel Detection Fault Mask	RW PD	0	**Parallel Detection Fault Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F:A	Reserved	RSV		Reserved. Do not modify.

10.4.32 Auto-Negotiation Standard Interrupt Mask 2: Address 7.D001

Bit	Name	Type	Default	Description
E:0	Reserved	RSV		Reserved. Do not modify.
F	Master/Slave Configuration Fault Mask	RW PD	0b	**Master/Slave Configuration Fault Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.

10.4.33 Auto-Negotiation Transmit Vendor Interrupt Mask 1: Address 7.D400

Bit	Name	Type	Default	Description
0	Connection State Change Mask	RW PD	0b	**Connection State Change Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
1	Automatic Downshift Mask	RW PD	0b	**Automatic Downshift Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
2	Auto-Negotiation Completed for Supported Rate Mask	RW PD	0b	**Auto-Negotiation Completed for Supported Rate Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
3	Auto-Negotiation Completed For Non-Supported Rate Mask	RW PD	0b	**Auto-Negotiation Completed For Non-Supported Rate Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F:4	Reserved	RSV		Reserved. Do not modify.

10.4.34 Auto-Negotiation Transmit Vendor Interrupt Mask 2: Address 7.D401

Bit	Name	Type	Default	Description
0	Link Connect/Disconnect Mask	RW PD	0b	**Link Connect/Disconnect Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
E:1	Reserved Vendor Alarms 2 Mask [D:0]	RW PD	0x0	**Reserved Vendor Alarms 2 Mask** Reserved for future use.
F	Link Pulse Detect Mask	RW PD	0b	**Link Pulse Detect Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.

10.4.35 Auto-Negotiation Transmit Vendor Interrupt Mask 3: Address 7.D402

Bit	Name	Type	Default	Description
F:0	Reserved	RSV		Reserved. Do not modify.

10.4.36 Auto-Negotiation Receive Link Partner Status 1: Address 7.E820

Bit	Name	Type	Default	Description
2:0	Reserved	RSV		Reserved. Do not modify.
9:3	Reserved	RSV		Reserved. Do not modify.
A	Link Partner 2.5G	RW PD	0b	**Link Partner 2.5G** 0b = Link Partner PHY is not 2.5 GbE capable. 1b = Link Partner PHY is 2.5 GbE capable.
B	Link Partner 5G	RW PD	0b	**Link Partner 5G** 0b = Link Partner PHY is not 5 GbE capable. 1b = Link Partner PHY is 5 GbE capable.
C	Link Partner AQRate	RW PD	0b	**Link Partner NBASE-T** 0b = Link Partner PHY does not support NBASE-T. 1b = Link Partner PHY supports NBASE-T.
D	Reserved	RSV		Reserved. Do not modify.
E	Link Partner 1000BASE-T Half Duplex Ability	RO		**Link Partner 1000BASE-T Half Duplex Ability** 0b = Link partner is not 1000BASE-T half-duplex capable. 1b = Link partner is 1000BASE-T half-duplex capable.
F	Link Partner 1000BASE-T Full Duplex Ability	RO		**Link Partner 1000BASE-T Full Duplex Ability** 0b = Link partner is not 1000BASE-T full-duplex capable. 1b = Link partner is 1000BASE-T full-duplex capable.

10.4.37 Auto-Negotiation Receive Link Partner Status 4: Address 7.E823

Bit	Name	Type	Default	Description
7:0	Link Partner Firmware Minor Revision Number [7:0]	RO		**Link Partner Firmware Minor/Major Revision Number** Only the lower six bits of major and minor firmware revision are exchanged in auto-negotiation when the PHYID message is sent. Consequently the upper two bits of the major and minor revision should always be zero. Bits [7:0] = Link partner firmware minor revision number. Bits [F:8] = Link partner firmware major revision number.
F:8	Link Partner Firmware Major Revision Number [7:0]	RO		

10.4.38 Auto-Negotiation Receive Vendor Alarms 1: Address 7.EC00

Bit	Name	Type	Default	Description
F:0	Reserved	RSV		Reserved. Do not modify.

10.4.39 Auto-Negotiation Receive Vendor Alarms 2: Address 7.EC01

Bit	Name	Type	Default	Description
B:0	Reserved Receive Vendor Alarms 2 [7:0]	LH		**Reserved Receive Vendor Alarms 2** Reserved for future use.
C	FLP Idle Error	LH		**FLP Idle Error** 1b = No FLP burst has been seen for 50 milliseconds forcing the receive state machine back to the Idle state. Once FLP bursts are detected on any receive channel, they must keep coming. If no burst has been detected for a period of 50 ms, the auto-negotiation process resets itself and goes back to the “break link” state.
D	Auto-Negotiation Protocol Error	LH		**Auto-Negotiation Protocol Error** 1b = Link partner has violated the auto-negotiation protocol. If the arbiter state machine detects a protocol violation, the auto-negotiation process resets itself and goes back to the “break link” state.
F:E	Reserved	LH		Reserved. Do not modify.

10.4.40 Auto-Negotiation Receive Vendor Alarms 3: Address 7.EC02

Bit	Name	Type	Default	Description
1:0	Reserved	RSV		Reserved. Do not modify.
2	10BASE-T Device Detect	LL		**10BASE-T Device Detect** This bit indicates that the detected far-end device is 10BASE-T when it is 0b. This bit is 1b when link pulses are no longer received. 0b = 10BASE-T device detected.
F:3	Reserved	RSV		Reserved. Do not modify.

10.4.41 Auto-Negotiation Receive Vendor Alarms 4: Address 7.EC03

Bit	Name	Type	Default	Description
0	100BASE-TX Parallel Detect	LH		**100BASE-TX Parallel Detect** 1b = 100BASE-TX parallel event detection circuit.
F:1	Reserved Receive Vendor Alarms 4 [E:0]	LH		**Reserved Receive Vendor Alarms 4** Reserved.

10.4.42 Auto-Negotiation Receive Vendor Interrupt Mask 1: Address 7.F400

Bit	Name	Type	Default	Description
F:0	Reserved Receive Vendor Alarms 1 Mask [F:0]	RW PD	0x0	**Reserved Receive Vendor Alarms 1 Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.

10.4.43 Auto-Negotiation Receive Vendor Interrupt Mask 2: Address 7.F401

Bit	Name	Type	Default	Description
B:0	Reserved Receive Vendor Alarms 2 Mask [B:0]	RW PD	0x0	**Reserved Receive Vendor Alarms 2 Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
C	FLP Idle Error Mask	RW PD	0b	**FLP Idle Error Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
D	Auto-Negotiation Protocol Error Mask	RW PD	0b	**Auto-Negotiation Protocol Error Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F:E	Reserved	RSV	0b	Reserved. Not supported.

10.4.44 Auto-Negotiation Receive Vendor Interrupt Mask 3: Address 7.F402

Bit	Name	Type	Default	Description
1:0	Reserved	RSV		Reserved. Do not modify.
2	10BASE-T Device Detect Mask	RW PD	0b	**10BASE-T Device Detect Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F:3	Reserved	RSV		Reserved. Do not modify.

10.4.45 Auto-Negotiation Receive Vendor Interrupt Mask 4: Address 7.F403

Bit	Name	Type	Default	Description
0	100BASE-TX Parallel Detect Mask	RW PD	0b	**100BASE-TX Parallel Detect Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F:1	Reserved Receive Vendor Alarms 4 Mask [E:0]	RW PD	0x0	**Reserved Receive Vendor Alarms 4 Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.

10.4.46 Auto-Negotiation Vendor Global Interrupt Flags 1: Address 7.FC00

Bit	Name	Type	Default	Description
0	Vendor Specific Rx Alarms 4 Interrupt	RO		**Vendor Specific Rx Alarms 4 Interrupt** An interrupt was generated from status register Auto-Negotiation Receive Vendor Alarms 4: Address 7.EC03 and the corresponding mask register Auto-Negotiation Receive Vendor Interrupt Mask 4: Address 7.F403. 1b = Interrupt
1	Vendor Specific Rx Alarms 3 Interrupt	RO		**Vendor Specific Rx Alarms 3 Interrupt** An interrupt was generated from status register Auto-Negotiation Receive Vendor Alarms 3: Address 7.EC02 and the corresponding mask register Auto-Negotiation Receive Vendor Interrupt Mask 3: Address 7.F402. 1b = Interrupt
2	Vendor Specific Rx Alarms 2 Interrupt	RO		**Vendor Specific Rx Alarms 2 Interrupt** An interrupt was generated from status register Auto-Negotiation Receive Vendor Alarms 2: Address 7.EC01 and the corresponding mask register Auto-Negotiation Receive Vendor Interrupt Mask 2: Address 7.F401. 1b = Interrupt
3	Vendor Specific Rx Alarms 1 Interrupt	RO		**Vendor Specific Rx Alarms 1 Interrupt** An interrupt was generated from status register Auto-Negotiation Receive Vendor Alarms 1: Address 7.EC00 and the corresponding mask register Auto-Negotiation Receive Vendor Interrupt Mask 1: Address 7.F400. 1b = Interrupt
8:4	Reserved	RSV		Reserved. Do not modify.
9	Vendor Specific Alarms 2 Interrupt	RO		**Vendor Specific Alarms 2 Interrupt** An interrupt was generated from status register Auto-Negotiation Transmit Vendor Alarms 2: Address 7.CC01 and the corresponding mask register Auto-Negotiation Transmit Vendor Interrupt Mask 2: Address 7.D401. 1b = Interrupt
A	Vendor Specific Alarms 1 Interrupt	RO		**Vendor Specific Alarms 1 Interrupt** An interrupt was generated from status register Auto-Negotiation Transmit Vendor Alarms 1: Address 7.CC00 and the corresponding mask register Auto-Negotiation Transmit Vendor Interrupt Mask 1: Address 7.D400. 1b = Interrupt
D:B	Reserved	RSV		Reserved. Do not modify.
E	Standard Alarms 2 Interrupt	RO		**Standard Alarms 2 Interrupt** An interrupt was generated from status register Auto-Negotiation 10GBASE-T Status Register: Address 7.21 and the corresponding mask register Auto-Negotiation Standard Interrupt Mask 2: Address 7.D001. 1b = Interrupt
F	Standard Alarms 1 Interrupt	RO		**Standard Alarms 1 Interrupt** An interrupt was generated from status register Auto-Negotiation Standard Status 1: Address 7.1 and the corresponding mask register Auto-Negotiation Standard Interrupt Mask 1: Address 7.D000. 1b = Interrupt

10.5 100BASE-TX and 1000BASE-T Registers

10.5.1 GbE Standard Device Identifier 1: Address 1D.2

Bit	Name	Type	Default	Description
F:0	Device ID MSW [1F:10]	RO		**Device ID MSW** Bits [31:16] of the Device ID.

10.5.2 GbE Standard Device Identifier 2: Address 1D.3

Bit	Name	Type	Default	Description
F:0	Device ID LSW [F:0]	RO		**Device ID LSW** Bits [15: 0] of the Device ID.

10.5.3 GbE Standard Devices in Package 1: Address 1D.5

Bit	Name	Type	Default	Description
0	Clause 22 Registers Present	ROS	0b	**Clause 22 Registers Present** 0b = Clause 22 registers are not present in the package. 1b = Clause 22 registers are present in the package. This is always set to 0b, as there are no Clause 22 registers in the X550.
1	PMA Present	ROS	1b	**PMA Present** 0b = PMA is not present. 1b = PMA is present in the package. This is always set to 1b, as there is PMA functionality in the X550.
2	WIS Present	ROS	0b	**WIS Present** 0b = WIS is not present in the package. 1b = WIS is present in the package. This is always set to 0b, as there is no WIS functionality in the X550.
3	PCS Present	ROS	1b	**PCS Present** 0b = PCS is not present in the package. 1b = PCS is present in the package. This is always set to 1b, as there is PCS functionality in the X550.
4	Control Present	ROS	1b	**Control Present** 0b = Control is not present in the package. 1b = Control is present in the package. This is always set to 1b, as there is a PHY XAUI interface in the X550.
5	DTE XS Present	ROS	0b	**DTE XS Present** 0b = DTE XS is not present in the package. 1b = DTE XS is present in the package. This is always set to 0b, as there is no MAC XAUI interface in the X550.
6	TC Present	ROS	0b	**TC Present** 0b = TC is not present in the package. 1b = TC is present in the package. This is always set to 0b, as there is no TC functionality in the X550.

Bit	Name	Type	Default	Description
7	Auto-Negotiation Present	ROS	1b	**Auto-Negotiation Present** 0b = Auto-negotiation is not present in the package. 1b = Auto-negotiation is present in the package. This is always set to 1b, as there is auto-negotiation in the X550.
F:8	Reserved	RSV		Reserved. Do not modify.

10.5.4 GbE Standard Vendor Devices in Package 2: Address 1D.6

Bit	Name	Type	Default	Description
C:0	Reserved	RSV		Reserved. Do not modify.
D	Clause 22 Extension Present	ROS	1b	**Clause 22 Extension Present** 0b = Clause 22 Extension is not present in the package. 1b = Clause 22 Extension is present in the package. This is always set to 1b, as the X550 uses this device for the global control registers.
E	Vendor Specific Device #1 Present	ROS	1b	**Vendor Specific Device #1 Present** 0b = Device #1 is not present in the package. 1b = Device #1 is present in the package. This is always set to 1b, as the X550 uses this device for the global control registers.
F	Vendor Specific Device #2 Present	ROS	1b	**Vendor Specific Device #2 Present** 0b = Device #2 is not present in the package. 1b = Device #2 is present in the package. This is always set to 1b, as the X550 uses this device for the DSP PMA registers.

10.5.5 GbE Standard Status 2: Address 1D.8

Bit	Name	Type	Default	Description
D:0	Reserved	RSV		Reserved. Do not modify.
F:E	Device Present [1:0]	ROS	10b	**Device Present** 00b = No device at this address. 01b = No device at this address. 10b = Device present at this address. 11b = No device at this address. This field is always set to 10b, as the control is present in the X550.

10.5.6 GbE Standard Package Identifier 1: Address 1D.E

Bit	Name	Type	Default	Description
F:0	Package ID MSW [1F:10]	RO		**Package ID MSW** Bits [31:16] of the Package ID.

10.5.7 GbE Standard Package Identifier 2: Address 1D.F

Bit	Name	Type	Default	Description
F:0	Package ID LSW [F:0]	RO		**Package ID LSW** Bits [15:0] of the Package ID.

10.5.8 GbE Reserved Provisioning 2: Address 1D.C501

Bit	Name	Type	Default	Description
1:0	100BASE-TX Test Mode [1:0]	RW	00b	**100BASE-TX Test Mode** 100BASE-TX IEEE test mode = MLT-3 idle sequence. ANSI jitter test = FDDI - Clause 9.1.3 Fig. 12. 00b = Normal mode. 01b = 100BASE-TX IEEE test mode. 10b = 100BASE-TX ANSI jitter test. 11b = Reserved.
C:2	Reserved Provisioning 2 [A:0]	RW PD	0x0	**Reserved Provisioning 2** Reserved for future use.
F:D	Test Mode [2:0]	RW PD	000b	**Test Mode** 000b = Normal mode. 001b = Test Mode 1 - Transmit waveform test. 010b = Test Mode 2 - Master transmit jitter test. 011b = Test Mode 3 - Slave transmit jitter test. 100b = Test Mode 4 - Transmitter distortion test. All other values are reserved.

10.6 Global Registers

10.6.1 Global Standard Control 1: Address 1E.0

Bit	Name	Type	Default	Description
A:0	Reserved	RSV		Reserved. Do not modify.
B	Low Power	RW PD	0b	**Low Power** A 1b written to this register causes the corresponding X550 PHY to enter low-power mode. This bit puts the entire PHY in low-power mode (with only the MDIO and microprocessor functioning) and turns off the Analog Front End (AFE). For example, places it in high-impedance mode. Setting this bit also sets all of the *Low Power* bits in the other MMDs. 0b = Normal operation. 1b = Low-power mode.
F:C	Reserved	RSV		Reserved. Do not modify.

10.6.2 Global Standard Device Identifier 1: Address 1E.2

Bit	Name	Type	Default	Description
F:0	Device ID MSW [1F:10]	RO		**Device ID MSW** Bits [31:16] of the Device ID.

10.6.3 Global Standard Device Identifier 2: Address 1E.3

Bit	Name	Type	Default	Description
F:0	Device ID LSW [F:0]	RO		**Device ID LSW** Bits [15:0] of the Device ID.

10.6.4 Global Standard Devices in Package 1: Address 1E.5

Bit	Name	Type	Default	Description
0	Clause 22 Registers Present	ROS	0b	**Clause 22 Registers Present** 0b = Clause 22 registers are not present in the package. 1b = Clause 22 registers are present in the package. This is always set to 0b, as there are no Clause 22 registers in the X550.
1	PMA Present	ROS	1b	**PMA Present** 0b = PMA is not present. 1b = PMA is present in the package. This is always set to 1b, as there is PMA functionality in the X550.
2	WIS Present	ROS	0b	**WIS Present** 0b = WIS is not present in the package. 1b = WIS is present in the package. This is always set to 0b, as there is no WIS functionality in the X550.

Bit	Name	Type	Default	Description
3	PCS Present	ROS	1b	**PCS Present** 0b = PCS is not present in the package. 1b = PCS is present in the package. This is always set to 1b, as there is PCS functionality in the X550.
4	PHY XS Present	ROS	0b	**PHY XS Present** 0b = PHY XS is not present in the package. 1b = PHY XS is present in the package. This is always set to 0b, as there is no PHY XS interface in the X550.
5	DTE XS Present	ROS	0b	**DTE XS Present** 0b = DTE XS is not present in the package. 1b = DTE XS is present in the package. This is always set to 0b, as there is no DTE XAUI interface in the X550.
6	TC Present	ROS	0b	**TC Present** 0b = TC is not present in the package. 1b = TC is present in the package. This is always set to 0b, as there is no TC functionality in the X550.
7	Auto-Negotiation Present	ROS	1b	**Auto-Negotiation Present** 0b = Auto-negotiation is not present in the package. 1b = Auto-negotiation is present in the package. This is always set to 1b, as there is auto-negotiation in the X550.
F:8	Reserved	RSV		Reserved. Do not modify.

10.6.5 Global Standard Vendor Devices in Package 2: Address 1E.6

Bit	Name	Type	Default	Description
C:0	Reserved	RSV		Reserved. Do not modify.
D	Clause 22 Extension Present	ROS	1b	**Clause 22 Extension Present** 0b = Clause 22 extension is not present in the package. 1b = Clause 22 extension is present in the package. This is always set to 1b, as the X550 uses this device for the GbE registers.
E	Vendor Specific Device #1 Present	ROS	1b	**Vendor Specific Device #1 Present** 0b = Device #1 is not present in the package. 1b = Device #1 is present in the package. This is always set to 1b, as the X550 uses this device for the global control registers.
F	Vendor Specific Device #2 Present	ROS	1b	**Vendor Specific Device #2 Present** 0b = Device #2 is not present in the package. 1b = Device #2 is present in the package. This is always set to 1b, as the X550 uses this device for the DSP PMA registers.

10.6.6 Global Standard Status 2: Address 1E.8

Bit	Name	Type	Default	Description
D:0	Reserved	RSV		Reserved. Do not modify.
F:E	Device Present [1:0]	ROS	10b	**Device Present** 00b = No device at this address. 01b = No device at this address. 10b = Device present at this address. 11b = No device at this address. This field is always set to 10b, as the global MMD resides here in the X550.

10.6.7 Global Standard Package Identifier 1: Address 1E.E

Bit	Name	Type	Default	Description
F:0	Package ID MSW [1F:10]	RO		**Package ID MSW** Bits [31:16] of the Package ID.

10.6.8 Global Standard Package Identifier 2: Address 1E.F

Bit	Name	Type	Default	Description
F:0	Package ID LSW [F:0]	RO		**Package ID LSW** Bits [15:0] of the Package ID.

10.6.9 Global Firmware ID: Address 1E.20

Bit	Name	Type	Default	Description
7:0	Firmware Minor Revision Number [7:0]	RO		**Firmware Minor Revision Number** The lower six bits of major and minor firmware revision are exchanged in auto-negotiation when the PHYID message is sent.
F:8	Firmware Major Revision Number [7:0]	RO		**Firmware Major Revision Number** The lower six bits of major and minor firmware revision are exchanged in auto-negotiation when the PHYID message is sent.

10.6.10 Global Diagnostic Provisioning: Address 1E.C400

Bit	Name	Type	Default	Description
E:0	Reserved	RSV		Reserved. Do not modify.
F	Enable Diagnostics	RW PD	0b	**Enable Diagnostics** 1b = The X550 performs diagnostics on power up.

10.6.11 Global Thermal Provisioning 2: Address 1E.C421

Bit	Name	Type	Default	Description
F:0	High Temp Failure Threshold [F:0]	RW PD	0x4600[1]	**High Temp Failure Threshold** [F:0] of high temperature failure threshold. 2's complement value with the LSB representing 1/256 °C. This corresponds to -40 °C = 0xD800. Default is 70 °C. ***Note:*** All Thresholds are orthogonal and can be set to any value regardless the value of the other thresholds. IN other words, High Temperature Warning (1E.C423) could be higher than High Temperature Failure (1E.C421).

1. The default value is overridden to 0x6600 by the provisioning mechanism.

10.6.12 Global Thermal Provisioning 3: Address 1E.C422

Bit	Name	Type	Default	Description
F:0	Low Temp Failure Threshold [F:0]	RW PD	0x0	**Low Temp Failure Threshold** [F:0] of low temperature failure threshold. 2's complement value with the LSB representing 1/256 °C. This corresponds to -40 °C = 0xD800. Default is 0 °C. ***Note:*** All Thresholds are orthogonal and can be set to any value regardless the value of the other thresholds. In other words, High Temperature Warning (1E.C423) could be higher than High Temperature Failure (1E.C421).

10.6.13 Global Thermal Provisioning 4: Address 1E.C423

Bit	Name	Type	Default	Description
F:0	High Temp Warning Threshold [F:0]	RW PD	0x3C00[1]	**High Temp Warning Threshold** [F:0] of high temperature warning threshold. 2's complement value with the LSB representing 1/256 °C. This corresponds to -40 °C = 0xD008. Default is 60 °C. ***Note:*** All Thresholds are orthogonal and can be set to any value regardless the value of the other thresholds. In other words, High Temperature Warning (1E.C423) could be higher than High Temperature Failure (1E.C421).

1. The default value is overridden to 0x6600 by the provisioning mechanism.

10.6.14 Global Thermal Provisioning 5: Address 1E.C424

Bit	Name	Type	Default	Description
F:0	Low Temp Warning Threshold [F:0]	RW PD	0x0A00	**Low Temp Warning Threshold** [F:0] of low temperature Warning threshold 2's complement value with the LSB representing 1/256 °C. This corresponds to -40 °C = 0xD800. Default is 10 °C. ***Note:*** All Thresholds are orthogonal and can be set to any value regardless the value of the other thresholds. In other words, High-Temperature-Warning (1E.C423) could be higher than High-Temperature-Failure (1E.C421).

10.6.15 Global Reserved Provisioning 1: Address 1E.C470

Bit	Name	Type	Default	Description
3:0	Reserved	RSV		Reserved. Do not modify.
4	Initiate Cable Diagnostics	RW SC	0b	**Initiate Cable Diagnostics** Perform cable diagnostics regardless of link state. If link is up, setting this bit causes the link to drop while diagnostics are performed. This register is self-clearing upon completion of the cable diagnostics. 1b = Perform cable diagnostics.
C:5	Reserved	RSV		Reserved. Do not modify.
E:D	Extended MDI Diagnostics Select [1:0]	RW PD	00b	**Extended MDI Diagnostics Select / Diagnostics Select** These bits select what sort of cable diagnostics to perform. For regular cable diagnostics, Bit [F] is set to 0b, and the diagnostics are triggered by setting Bit [4]. For extended diagnostics, Bit [F] is set to 1b, and the desired extended diagnostics are selected by Bits [E:D]. The routine is then triggered by setting Bit [4]. Each of the extended diagnostic routines present data for all for MDI pairs (A, B, C, D) consecutively, and after the data for each channel is gathered Bits [F:D] are reset. To get the data for the next pair, Bits [F:D] must be set back to the desired value, which must be the same as the initial channel. This continues until the data for all channels has been gathered. The address in memory where the data is stored is given in 1E.C802 and 1E.C804. For PSD, the structure is as follows: • Int32 info • Int16 data[Len] • Info = Len << 16 | TxEnable << 8 | Pair (0 = A, etc.) For TDR: • Int32 info • Int16 tdr_A[Len] • Int16 tdr_B[Len] • Int16 tdr_C[Len] • Int16 tdr_D[Len] Info = Len << 16 | Channel TDR data is from the current pair or all other pairs. At the end of retrieving extended MDI diagnostic data, the X550 is reset. Conversely, the only way to exit this routine once it starts is to issue a PMA reset. Bits [E:D] settings: 00b = TDR data. 01b = RFI channel PSD. 10b = Noise PSD while the local Tx is off. 11b = Noise PSD while the local Tx is on. Bit [F] settings: 0b = Provide normal cable diagnostics. 1b = Provide extended MDI diagnostics information.
F	Diagnostics Select	RW PD	0b	

10.6.16 Global Reserved Provisioning 3: Address 1E.C472

Bit	Name	Type	Default	Description
5:0	Reserved	RSV		Reserved. Do not modify.
6	Tunable External VDD Power Supply Present	RW PD	0b	**Tunable External VDD Power Supply Present** 0b = No tunable external VDD power supply present. 1b = Tunable external VDD power supply present. This bit must be set if tuning of external power supply is desired.
D:7	Reserved	RSV		Reserved. Do not modify.

Bit	Name	Type	Default	Description
E	Enable VDD Power Supply Tuning	RW PD	0b	**Enable VDD Power Supply Tuning** This bit controls whether the PHY attempts to tune the external VDD power supply via the SMBus. This bit is only operational if the external supply is present. See 1E.C472.6 (Section 10.6.16). 0b = Disable external VDD power supply tuning is disabled 1b = Enable external VDD power supply tuning
F	Reserved	RSV		Reserved. Do not modify.

10.6.17 Global Reserved Provisioning 5: Address 1E.C474

Bit	Name	Type	Default	Description
0	NVM Daisy Chain Kickstart	RW PD	0x0	**NVM Daisy Chain Kickstart** When in daisy chain master mode, the PHY0 can kickstart the daisy chain. The kickstart does not reload the IRAM/DRAM or reset the uP for PHY0. It just reads the flash and transfer the flash data to the daisy chain. This is for backwards compatibility with the 82599. 1b = Kickstart the Daisy Chain
F:1	Reserved	RSV		Reserved. Do not modify.

10.6.18 Global Reserved Provisioning 6: Address 1E.C475

Bit	Name	Type	Default	Description
3:0	Reserved	RSV		Reserved. Do not modify.
4	CFR Support	RW PD	0b	**CFR Support** 0b = Local PHY does support Cisco Fast Retrain. 1b = Local PHY supports Cisco Fast Retrain.
5	CFR THP	RW PD	0b	**CFR THP** 0b = Local PHY does not require local PHY to enable THP. 1b = Local PHY requires local PHY to enable THP.
6	CFR Extended Max wait	RW PD	0b	**CFR Extended Max wait** 0b = Local PHY does not require extended max wait. 1b = Local PHY requires extended max wait.
7	CFR Disable Timer	RW PD	0b	**CFR Disable Timer** 0b = Local PHY does not require cfr_disable timer. 1b = Local PHY requires cfr_disable timer.
8	CFR LP Support	RW PD	0b	**CFR LP Support** 0b = Link partner does support Cisco Fast Retrain. 1b = Link partner supports Cisco Fast Retrain.
9	CFR LP THP	RW PD	0b	**CFR LP THP** 0b = Link partner does not require local PHY to enable THP. 1b = Link partner requires local PHY to enable THP.
A	CFR LP Extended Max wait	RW PD	0b	**CFR LP Extended Max wait** 0b = Link partner does not require extended max wait. 1b = Link partner requires extended max wait.
B	CFR LP Disable Timer	RW PD	0b	**CFR LP Disable Timer** 0b = Link partner does not require cfr_disable timer. 1b = Link partner requires cfr_disable timer.
F:C	Reserved	RSV	0x0	Reserved. Do not modify.

10.6.19 Global SMBus 0 Provisioning 6: Address 1E.C485

Bit	Name	Type	Default	Description
0	Reserved	RSV		Reserved. Do not modify.
7:1	SMB 0 Slave Address [7:1]	RW	0x0	**SMB 0 Slave Address** SMB slave address configuration.
F:8	Reserved	RSV		Reserved. Do not modify.

10.6.20 Global SMBus 1 Provisioning 6: Address 1E.C495

Bit	Name	Type	Default	Description
0	Reserved	RSV		Reserved. Do not modify.
7:1	SMB 1 Slave Address [7:1]	RW	0x0	**SMB 1 Slave Address** SMB slave address configuration.
F:8	Reserved	RSV		Reserved. Do not modify.

10.6.21 Global Cable Diagnostic Status 1: Address 1E.C800

Bit	Name	Type	Default	Description
2:0	Pair D Status [2:0]	RO		**Pair D Status** This register summarizes the worst case impairment on pair D. [6:4] 000b = OK. 001b = Connected to Pair C. 010b = Connected to Pair B. 011b = Connected to Pair A. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
3	Reserved	RSV		Reserved. Do not modify.
6:4	Pair C Status [2:0]	RO		**Pair C Status** This register summarizes the worst case impairment on pair C. [9:7] 000b = OK. 001b = Connected to Pair B. 010b = Connected to Pair A. 011b = Connected to Pair D. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
7	Reserved	RSV		Reserved. Do not modify.

Bit	Name	Type	Default	Description
A:8	Pair B Status [2:0]	RO		**Pair B Status** This register summarizes the worst case impairment on pair B. [C:A]: 000b = OK. 001b = Connected to Pair A. 010b = Connected to Pair D. 011b = Connected to Pair C. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
B	Reserved	RSV		Reserved. Do not modify.
E:C	Pair A Status [2:0]	RO		**Pair A Status** This register summarizes the worst case impairment on pair A. [F:D]: 000b = OK. 001b = Connected to Pair D. 010b = Connected to Pair C. 011b = Connected to Pair B. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
F	Reserved	RSV		Reserved. Do not modify.

10.6.22 Global Cable Diagnostic Status 2: Address 1E.C801

Bit	Name	Type	Default	Description
7:0	Pair A Reflection #2 [7:0]	RO		**Pair A Reflection #2/#1** The distance in meters (accurate to ±1 m) of the second of the four worst case reflections seen by the PHY on Pair A. The distance to this reflection is given in Global Cable Diagnostic Impedance 1: Address 1E.C880 register (Section 10.6.34). A value of zero indicates that this reflection does not exist or was not computed.
F:8	Pair A Reflection #1 [7:0]	RO		

10.6.23 Global Cable Diagnostic Status 3: Address 1E.C802

Bit	Name	Type	Default	Description
F:0	Impulse Response MSW [F:0]	RO		**Impulse Response MSW** The MSW of the memory location that contains the start of the impulse response data for the extended diagnostic type in 1E.C470.E:D. See 1E.C470 (Section 10.6.15).

10.6.24 Global Cable Diagnostic Status 4: Address 1E.C803

Bit	Name	Type	Default	Description
7:0	Pair B Reflection #2 [7:0]	RO		**Pair B Reflection #2/#1** The distance in meters (accurate to ±1 m) of the first of the four worst case reflections seen by the PHY on Pair B. The distance to this reflection is given in Global Cable Diagnostic Impedance 2: Address 1E.C881 register (Section 10.6.35). A value of zero indicates that this reflection does not exist or was not computed.
F:8	Pair B Reflection #1 [7:0]	RO		

10.6.25 Global Cable Diagnostic Status 5: Address 1E.C804

Bit	Name	Type	Default	Description
F:0	Impulse Response LSW [F:0]	RO		**Impulse Response LSW** The LSW of the memory location that contains the start of the impulse response data for the extended diagnostic type in 1E.C470.E:D. See 1E.C470 (Section 10.6.15).

10.6.26 Global Cable Diagnostic Status 6: Address 1E.C805

Bit	Name	Type	Default	Description
7:0	Pair C Reflection #2 [7:0]	RO		**Pair C Reflection #2/#1** The distance in meters (accurate to ±1 m) of the first of the four worst case reflections seen by the PHY on Pair C. The distance to this reflection is given in Global Cable Diagnostic Impedance 3: Address 1E.C882 register (Section 10.6.36). A value of zero indicates that this reflection does not exist or was not computed.
F:8	Pair C Reflection #1 [7:0]	RO		

10.6.27 Global Cable Diagnostic Status 7: Address 1E.C806

Bit	Name	Type	Default	Description
F:0	Reserved	RSV		Reserved.

10.6.28 Global Cable Diagnostic Status 8: Address 1E.C807

Bit	Name	Type	Default	Description
7:0	Pair D Reflection #2 [7:0]	RO		**Pair D Reflection #2/#1** The distance in meters (accurate to ±1 m) of the first of the four worst case reflections seen by the PHY on Pair D. The distance to this reflection is given in Global Cable Diagnostic Impedance 4: Address 1E.C883 register (Section 10.6.37). A value of zero indicates that this reflection does not exist or was not computed.
F:8	Pair D Reflection #1 [7:0]	RO		

10.6.29 Global Thermal Status 1: Address 1E.C820

Bit	Name	Type	Default	Description
F:0	Temperature [F:0]	RO		**Temperature** [F:0] of temperature. 2's complement value with the LSB representing 1/256 °C. This corresponds to -40 °C = 0xD800. Default is 70 °C.

10.6.30 Global Thermal Status 2: Address 1E.C821

Bit	Name	Type	Default	Description
0	Temperature Ready	RO		**Temperature Ready** 1b = Temperature measurement is valid.
F:1	Reserved	RSV		Reserved. Do not modify.

10.6.31 Global General Status 1: Address 1E.C830

Bit	Name	Type	Default	Description
A:0	Reserved	RSV		Reserved. Do not modify.
B	Low Temperature Warning State	RO		**Low Temperature Warning State** 1b = Low temperature warning threshold has been exceeded.
C	High Temperature Warning State	RO		**High Temperature Warning State** 1b = High temperature warning threshold has been exceeded.
D	Low Temperature Failure State	RO		**Low Temperature Failure State** 1b = Low temperature failure threshold has been exceeded.
E	High Temperature Failure State	RO		**High Temperature Failure State** 1b = High temperature failure threshold has been exceeded.
F	Reserved	RSV	0b	Reserved. Do not modify.

10.6.32 Global Fault Message: Address 1E.C850

Bit	Name	Type	Default	Description
F:0	Message [F:0]	RO		**Message** Error code describing fault. Code Number: 0x8001 = Firmware is not compatible with the chip architecture. This fault occurs when firmware is compiled for a different core that is loaded. 0x8002 = VCO calibration failed. This occurs when the main PLLs on the chip fail to lock (no trigger is possible). 0x8003 = XAUI calibration failed. This occurs when the XAUI PLLs fail to lock (no trigger is possible). 0x8005 = Unexpected device ID. This occurs if the device ID programmed into the internal E-Fuse registers in not valid (no trigger is possible). 0x8006 = Computed checksum does not match expected checksum. This occurs when the Flash checksum check performed at boot time fails. This only occurs when the system boots from Flash. 0x8007 = Detected a bit error in static memory. To trigger, corrupt one of the static regions. 0xC001 = Illegal Instruction exception. This occurs when the processor attempts to execute an illegal instruction. To trigger this, write an illegal instruction to program memory. It's possible that the bit error check triggers before the illegal instruction executes. 0xC002 = Instruction Fetch Error. Internal physical address or a data error during instruction fetch (no trigger is possible). 0xC003 = Load Store Error. Internal physical address or data error during load store operation (no trigger is possible). 0xC004 = Privileged Instruction. Attempt to execute a privileged operation without sufficient privilege (no trigger is possible). 0xC005 = Unaligned Load or Store. Attempt to load or store data at an address that cannot be handled due to alignment (no trigger is possible). 0xC006 = Instruction fetch from prohibited space (no trigger is possible). 0xC007 = Data load from prohibited space (no trigger is possible). 0xC008 = Data store into prohibited space (no trigger is possible). 0xDEAD = Uncorrectable IRAM error. 0xDEAE = DRAM parity error. 0xDD00 = CRD16 IRAM check failure (IRAM load error). 0xFACA = E-Fuse CRC16 check failure (E-Fuse is corrupted).

10.6.33 Global Primary Status: Address 1E.C851

Bit	Name	Type	Default	Description
0	Primary Status	RO		**Primary Status** 0b = PHY is the secondary PHY. 1b = PHY is the primary PHY.
F:1	Reserved	RSV		Value always 0b. Writes are ignored.

10.6.34 Global Cable Diagnostic Impedance 1: Address 1E.C880

Bit	Name	Type	Default	Description
2:0	Pair A Reflection #4 [2:0]	RO		**Pair A Reflection #4** The impedance of the fourth worst case reflection on Pair A. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 1: Address 1E.C800 register (Section 10.6.21). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
3	Reserved	RSV		Reserved
6:4	Pair A Reflection #3 [2:0]	RO		**Pair A Reflection #3** The impedance of the third worst case reflection on Pair A. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 1: Address 1E.C800 register (Section 10.6.21). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
7	Reserved	RSV		Reserved
A:8	Pair A Reflection #2 [2:0]	RO		**Pair A Reflection #2** The impedance of the second worst case reflection on Pair A. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 1: Address 1E.C800 register (Section 10.6.21). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
B	Reserved	RSV		Reserved
E:C	Pair A Reflection #1 [2:0]	RO		**Pair A Reflection #1** The impedance of the first worst case reflection on Pair A. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 1: Address 1E.C800 register (Section 10.6.21). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
F	Reserved	RSV		Reserved

10.6.35 Global Cable Diagnostic Impedance 2: Address 1E.C881

Bit	Name	Type	Default	Description
2:0	Pair B Reflection #4 [2:0]	RO		**Pair B Reflection #4** The impedance of the fourth worst case reflection on Pair B. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 2: Address 1E.C801 register (Section 10.6.22). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
3	Reserved	RSV		Reserved
6:4	Pair B Reflection #3 [2:0]	RO		**Pair B Reflection #3** The impedance of the third worst case reflection on Pair B. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 2: Address 1E.C801 register (Section 10.6.22). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
7	Reserved	RSV		Reserved
A:8	Pair B Reflection #2 [2:0]	RO		**Pair B Reflection #2** The impedance of the second worst case reflection on Pair B. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 2: Address 1E.C801 register (Section 10.6.22). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
B	Reserved	RSV		Reserved
E:C	Pair B Reflection #1 [2:0]	RO		**Pair B Reflection #1** The impedance of the first worst case reflection on Pair B. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 2: Address 1E.C801 register (Section 10.6.22). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
F	Reserved	RSV		Reserved

10.6.36 Global Cable Diagnostic Impedance 3: Address 1E.C882

Bit	Name	Type	Default	Description
2:0	Pair C Reflection #4 [2:0]	RO		**Pair C Reflection #4** The impedance of the fourth worst case reflection on Pair C. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 3: Address 1E.C802 register (Section 10.6.23). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
3	Reserved	RSV		Reserved
6:4	Pair C Reflection #3 [2:0]	RO		**Pair C Reflection #3** The impedance of the third worst case reflection on Pair C. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 3: Address 1E.C802 register (Section 10.6.23). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
7	Reserved	RSV		Reserved
A:8	Pair C Reflection #2 [2:0]	RO		**Pair C Reflection #2** The impedance of the second worst case reflection on Pair C. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 3: Address 1E.C802 register (Section 10.6.23). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
B	Reserved	RSV		Reserved
E:C	Pair C Reflection #1 [2:0]	RO		**Pair C Reflection #1** The impedance of the first worst case reflection on Pair C. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 3: Address 1E.C802 register (Section 10.6.23). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
F	Reserved	RSV		Reserved

10.6.37 Global Cable Diagnostic Impedance 4: Address 1E.C883

Bit	Name	Type	Default	Description
2:0	Pair D Reflection #4 [2:0]	RO		**Pair D Reflection #4** The impedance of the fourth worst case reflection on Pair D. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 4: Address 1E.C803 register (Section 10.6.24). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
3	Reserved	RSV		Reserved
6:4	Pair D Reflection #3 [2:0]	RO		**Pair D Reflection #3** The impedance of the third worst case reflection on Pair D. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 4: Address 1E.C803 register (Section 10.6.24). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
7	Reserved	RSV		Reserved
A:8	Pair D Reflection #2 [2:0]	RO		**Pair D Reflection #2** The impedance of the second worst case reflection on Pair D. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 4: Address 1E.C803 register (Section 10.6.24). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> 115 Ω). 111b = Open circuit (> 300 Ω).
B	Reserved	RSV		Reserved
E:C	Pair D Reflection #1 [2:0]	RO		**Pair D Reflection #1** The impedance of the first worst case reflection on Pair D. The corresponding length of this reflection from the PHY is given in the Global Cable Diagnostic Status 4: Address 1E.C803 register (Section 10.6.24). 0xxb = No information available. 100b = Short circuit (< 30 Ω). 101b = Low mismatch (< 85 Ω). 110b = High mismatch (> Ω Ω). 111b = Open circuit (> 300 Ω).
F	Reserved	RSV		Reserved

10.6.38 Global Status: Address 1E.C884

Bit	Name	Type	Default	Description
7:0	Cable Length [7:0]	RO		**Cable Length** The estimated length of the cable in meters. The length of the cable shown here is estimated from the cable diagnostic engine and should be accurate to ± 1 m.
F:8	Reserved Status 0 [7:0]	RO		Reserved.

10.6.39 Global Reserved Status 1: Address 1E.C885

Bit	Name	Type	Default	Description
3:0	Provisioning ID [3:0]	ROS PD	0x0	**Provisioning ID** Customers may receive multiple ROM images that differ only in their provisioning. This field is used to differentiate those images. This field is used in conjunction with the firmware major and minor revision numbers to uniquely identify ROM images.
7:4	Firmware Build ID [3:0]	ROS PD	0x0	**Firmware Build ID** Customers might receive multiple ROM images that differ only in their provisioning. This field is used to differentiate those images. This field is used in conjunction with the firmware major and minor revision numbers to uniquely identify ROM images.
9:8	NVM Status [1:0]	ROS PD	00b	**NVM Status** This register indicates the status of the last NVM operation. Status of NVM: 00b = NVM not enabled. 01b = Last NVM operation succeeded. 10b = Last NVM operation failed. 11b = Reserved.
F:A	Nearly Seconds MSW [5:0]	RO		**Nearly Seconds MSW** Bits [21:16] of the 22-bit nearly seconds uptime counter. The nearly seconds counter is incremented every 1024 ms.

10.6.40 Global Reserved Status 2: Address 1E.C886

Bit	Name	Type	Default	Description
F:0	Nearly Seconds LSW [F:0]	RO		**Nearly Seconds LSW** Bit [15:0] of the 22-bit nearly seconds uptime counter. The nearly seconds counter is increased every 1024 ms.

10.6.41 Global Reserved Status 3: Address 1E.C887

Bit	Name	Type	Default	Description
D:0	Reserved Status 3 [D:0]	RSV	0b	**Reserved Status 3** Reserved for future use.
E	Power Up Stall Status	ROS	0b	**Power Up Stall Status** 0b = Firmware is unstalled. 1b = Firmware is stalled at power up.
F	DTE Status	ROS		**DTE Status** 0b = Does not need power. 1b = Needs power.

10.6.42 Global Reserved Status 4: Address 1E.C888

Bit	Name	Type	Default	Description
2:0	Rate [2:0]	RO	000b	**Rate** These bits report the selected rate for the loopback and packet generation. 000b = Invalid 001b = 100 Mb/s 010b = 1 GbE 011b = 10 GbE 100b = 2.5 GbE 101b = 5 GbE 110b = Reserved 111b = Reserved
3	System I/F Packet Generation Status	RO	0b	**System I/F Packet Generation Status** Reports whether the CRPAT packet generator in the PHY outputs on the selected system interface at the selected rate. 0b = No CRPAT packet generation out 10 GbE system interface. 1b = CRPAT packet generation out 10 GbE system interface.
4	Reserved	RSV		Reserved. Do not modify
5	MDI Packet Generation Status	RO	0b	**MDI Packet Generation Status** Reports whether the CRPAT packet generator in the PHY outputs on the MDI interface at the selected rate. 0b = No CRPAT packet generation out MDI interface. 1b = CRPAT packet generation out MDI interface.
A:6	Reserved Status 4 [5:0]	RO	0x0	**Reserved Status 4** Reserved for future use.
F:B	Loopback Status [5:0]	RW PD	0x0	**Loopback Status** These bits, in conjunction with the chip configuration and the rate (Bits 1:0), report the selected loopback. 0x00 = No loopback 0x01 = System Interface — System Loopback 0x02 = System Interface — System Loopback with Pass-through 0x03 = System Interface — Network Loopback 0x04 = System Interface — Network Loopback with Pass-through 0x05 = System Interface — Network Loopback with Pass-through and Merge 0x06 = System Interface — Peer-to-peer loopback 0x09 = Network Interface — System Loopback 0x0A = Network Interface — System Loopback with Pass-through 0x0B = Network Interface — Network Loopback 0x0C = Network Interface — Network Loopback with Pass-through 0x0D = Network Interface — Peer-to-peer loopback 0x10 = Cross-connect System Loopback 0x11 = Cross-connect Network Loopback 0x14 = Network Interface — System Loopback via Loopback Plug All other values are reserved.

10.6.43 Global Alarms 1: Address 1E.CC00

Bit	Name	Type	Default	Description
0	Reserved Alarm D	LH		**Reserved Alarm D** Reserved for future use.
1	Reserved Alarm C	LH		**Reserved Alarm C** Reserved for future use.
2	Reserved Alarm B	LH		**Reserved Alarm B** Reserved for future use.
3	Reserved Alarm A	LH		**Reserved Alarm A** Reserved for future use.
4	Device Fault	LH		**Device Fault** When set, a fault has been detected by the µP and the associated 16-bit error code is visible in the Global Fault Message: Address 1E.C850 register (Section 10.6.32). 1b = Fault
5	Reserved	RSV		Reserved. Do not modify.
6	Reset Completed	LH		**Reset Completed** This bit is set by the µP completing it's initialization sequence. This bit is mirrored in 1.CC02.0 (Section 10.6.45) 1b = Chip wide reset completed.
A:7	Reserved	RSV		Reserved. Do not modify.
B	Low Temperature Warning	LH		**Low Temperature Warning** This bit mirrors the matching bit in 1.A070 and 1.A074. If latched high behavior is enabled for this bit via 1E.C440.0, reading one register is the same as reading the other. 1b = Low temperature warning threshold has been exceeded.
C	High Temperature Warning	LH		**High Temperature Warning** This bit mirrors the matching bit in 1.A070 and 1.A074. If latched high behavior is enabled for this bit via 1E.C440.0, reading one register is the same as reading the other. 1b = High temperature warning threshold has been exceeded.
D	Low Temperature Failure	LH		**Low Temperature Failure** This bit mirrors the matching bit in 1.A070 and 1.A074. If latched high behavior is enabled for this bit via 1E.C440.0, reading one register is the same as reading the other. 1b = Low temperature failure threshold has been exceeded.
E	High Temperature Failure	LH		**High Temperature Failure** This bit mirrors the matching bit in 1.A070 and 1.A074. If latched high behavior is enabled for this bit via 1E.C440.0, reading one register is the same as reading the other. 1b = High temperature failure threshold has been exceeded.
F	Reserved	RSV	0b	Reserved. Do not modify.

10.6.44 Global Alarms 2: Address 1E.CC01

Bit	Name	Type	Default	Description
0	Diagnostic Alarm	LH		**Diagnostic Alarm** A diagnostic alarm used to test system alarm circuitry. 1b = Alarm triggered by a write to 1E.C470.7.
1	MAC PHY Clear Reset Request Handled	LH		**MAC PHY Clear Reset Request Handled** 1b = PHY handled the MAC clear reset request.
2	MAC PHY Normal State Request Handled	LH		**MAC PHY Normal State Request Handled** 1b = PHY handled the MAC normal state request.
3	MAC PHY Low Power Request Handled	LH		**MAC PHY Low Power Request Handled** 1b = PHY handled the MAC low power request.
4	MAC PHY Disable Request Handled	LH		**MAC PHY Disable Request Handled** 1b = PHY handled the MAC disable request.
5	MAC Reset Request Handled	LH		**MAC Reset Request Handled** 1b = PHY handled the MAC reset request.
6	Reserved Alarm E	LH		**Reserved Alarm E** Reserved.
7	MDIO Command Handling Overflow	LH		**MDIO Command Handling Overflow** Assertion of this bit means that more MDIO commands were issued than firmware could handle. 1b = PHY was issued more MDIO requests than it could service in it's request buffer
9:8	Reserved Alarms [1:0]	LH		**Reserved Alarms** Reserved.
A	Fast Link Drop	LH		**Fast Link Drop** This alarm is asserted before entering PHY link recovery state. This alarm can be used as an early sign for a link drop in case of unsuccessful link recovery. 1b = PHY has entered link recovery state.
B	DTE Status Change	LH		**DTE Status Change** Change in 1E.C887.F (Section 10.6.41). 1b = DTE status change.
C	IP Phone Detect	LH		**IP Phone Detect** Assertion of this bit means that the presence of an IP Phone has been detected. 1b = IP Phone Detect
F:D	Reserved	RSV		Reserved. Do not modify.

10.6.45 Global Alarms 3: Address 1E.CC02

Bit	Name	Type	Default	Description
0	Watchdog Timer Alarm	LH		**Watchdog Timer Alarm** 1b = Watchdog timer alarm
1	MDIO Timeout Error	LH		**MDIO Timeout Error** 1b = MDIO timeout detected
2	MDIO MMD Error	LH		**MDIO MMD Error** 1b = Invalid MMD address detected
4:3	Reserved	RSV		Reserved. Do not modify.
5	Tx Enable State Change	LRF		**Transmit Enable State Change** 1b = TX_EN pin has changed state
7:6	Reserved	RSV		Reserved. Do not modify.
9:8	uP IRAM Parity Error [1:0]	LH		**uP IRAM Parity Error** Bit [0] indicates a parity error was detected in the uP IRAM but was corrected. Bit [1] indicates a multiple parity errors were detected in the uP IRAM and could not be corrected. The uP IRAM is protected with ECC. 1b = Parity error detected in the uP IRAM
A	uP DRAM Parity Error	LH		**uP DRAM Parity Error** 1b = Parity error detected in the uP DRAM
D:B	Reserved	RSV		Reserved. Do not modify.
E	Mailbox Operation Complete	LH		**Mailbox Operation Complete** Mailbox interface is ready interrupt for registers 1b = Mailbox operation is complete
F	NVM Operation Complete	LH		**NVM Operation Complete** NVM interface is ready interrupt for registers 1b = NVM operation is complete

10.6.46 Global Interrupt Mask 1: Address 1E.D400

Bit	Name	Type	Default	Description
0	Reserved Alarm D Mask	RW PD	0b	**Reserved Alarm D Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
1	Reserved Alarm C Mask	RW PD	0b	**Reserved Alarm C Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
2	Reserved Alarm B Mask	RW PD	0b	**Reserved Alarm B Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
3	Reserved Alarm A Mask	RW PD	0b	**Reserved Alarm A Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
4	Device Fault Mask	RW PD	0b	**Device Fault Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
5	Reserved	RSV		Reserved. Do not modify.
6	Reset Completed Mask	RW PD	0b	**Reset Completed Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
A:7	Reserved	RSV		Reserved. Do not modify.
B	Low Temperature Warning Mask	RW PD	0b	**Low Temperature Warning Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
C	High Temperature Warning Mask	RW PD	0b	**High Temperature Warning Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
D	Low Temperature Failure Mask	RW PD	0b	**Low Temperature Failure Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
E	High Temperature Failure Mask	RW PD	1b	**High Temperature Failure Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F	Reserved	RSV		Reserved. Do not modify.

10.6.47 Global Interrupt Mask 2: Address 1E.D401

Bit	Name	Type	Default	Description
0	Diagnostic Alarm Mask	RW PD	0b	**Diagnostic Alarm Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
1	MAC PHY Clear Reset Request Handled Mask	RW PD	0b	**MAC PHY Clear Reset Request Handled Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
2	MAC PHY Normal State Request Handled Mask	RW PD	0b	**MAC PHY Normal State Request Handled Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
3	MAC PHY Low Power Request Handled Mask	RW PD	0b	**MAC PHY Low Power Request Handled Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
4	MAC PHY Disable Request Handled Mask	RW PD	0b	**MAC PHY Disable Request Handled Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
5	MAC Reset Request Handled Mask	RW PD	0b	**MAC Reset Request Handled Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
6	Reserved Alarm E Mask	RW PD	0b	**Reserved Alarm E Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
7	MDIO Command Handling Overflow Mask	RW PD	0b	**MDIO Command Handling Overflow Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
9:8	Reserved Alarms Mask [1:0]	RW PD	00b	**Reserved Alarms Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
A	Fast Link Drop Mask	RW PD	0b	**Fast Link Drop Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
B	DTE Status Change Mask	RW PD	0b	**DTE Status Change Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
C	IP Phone Detect Mask	RW PD	0b	**IP Phone Detect Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F:D	Reserved	RSV		Reserved. Do not modify.

10.6.48 Global Interrupt Mask 3: Address 1E.D402

Bit	Name	Type	Default	Description
0	Watchdog Timer Alarm Mask	RW PD	1b	**Watchdog Timer Alarm Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
1	MDIO Timeout Error Mask	RW PD	0b	**MDIO Timeout Error Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
2	MDIO MMD Error Mask	RW PD	0b	**MDIO MMD Error Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
4:3	Reserved	RSV		Reserved. Do not modify.
5	Tx Enable State Change Mask	RW PD	0b	**Transmit Enable State Change Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
7:6	Reserved	RSV		Reserved. Do not modify.
9:8	uP IRAM Parity Error [1:0] Mask	RW PD	00b	**uP IRAM Parity Error Mask** Bit [0] indicates a parity error was detected in the uP IRAM but was corrected. Bit [1] indicates a multiple parity errors were detected in the uP IRAM and could not be corrected. The uP IRAM is protected with ECC. 0b = Disable interrupt generation. 1b = Enable interrupt generation.
A	uP DRAM Parity Error Mask	RW PD	0b	**uP DRAM Parity Error Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
D:B	Reserved	RSV		Reserved. Do not modify.
E	Mailbox Operation Complete Mask	RW PD	0b	**Mailbox Operation Complete Mask** Mailbox interface is ready interrupt for registers 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F	NVM Operation Complete Mask	RW PD	0b	**NVM Operation Complete Mask** NVM interface is ready interrupt for registers 0b = Disable interrupt generation. 1b = Enable interrupt generation.

10.6.49 Global Chip-Wide Standard Interrupt Flags: Address 1E.FC00

Bit	Name	Type	Default	Description
0	All Vendor Alarms Interrupt	RO		**All Vendor Alarms Interrupt** An interrupt was generated from status register Global Chip-Wide Vendor Interrupt Flags: Address 1E.FC01 and the corresponding mask register Global Interrupt Chip-Wide Vendor Mask: Address 1E.FF01. 1b = Interrupt in all vendor alarms.
5:1	Reserved	RSV		Reserved. Do not modify.
6	GbE Standard Alarms Interrupt	RO		**GbE Standard Alarms Interrupt** An interrupt was generated from the TGE core. 1b = Interrupt in GbE standard alarms.
7	Auto-Negotiation Standard Alarms 2 Interrupt	RO		**Auto-Negotiation Standard Alarms 2 Interrupt** An interrupt was generated from status register Auto-Negotiation 10GBASE-T Status Register: Address 7.21 and the corresponding mask register Auto-Negotiation Standard Interrupt Mask 2: Address 7.D001. 1b = Interrupt in auto-negotiation standard alarms 2.
8	Auto-Negotiation Standard Alarms 1 Interrupt	RO		**Auto-Negotiation Standard Alarms 1 Interrupt** An interrupt was generated from status register Auto-Negotiation Standard Status 1: Address 7.1 and the corresponding mask register Auto-Negotiation Standard Interrupt Mask 1: Address 7.D000. 1b = Interrupt in auto-negotiation standard alarms 1.
A:9	Reserved	RSV		Reserved. Do not modify.
B	PCS Standard Alarm 3 Interrupt	RO		**PCS Standard Alarm 3 Interrupt** An interrupt was generated from status register PCS 10GBASE-T Status 2: Address 3.21 and the corresponding mask register PCS Standard Interrupt Mask 3: Address 3.D002. 1b = Interrupt in PCS standard alarms 3.
C	PCS Standard Alarm 2 Interrupt	RO		**PCS Standard Alarm 2 Interrupt** An interrupt was generated from status register PCS Standard Status 2: Address 3.8 and the corresponding mask register PCS Standard Interrupt Mask 2: Address 3.D001. 1b = Interrupt in PCS standard alarms 2.
D	PCS Standard Alarm 1 Interrupt	RO		**PCS Standard Alarm 1 Interrupt** An interrupt was generated from status register PCS Standard Status 1: Address 3.1 and the corresponding mask register PCS Standard Interrupt Mask 1: Address 3.D000. 1b = Interrupt in PCS standard alarms 1.
E	PMA Standard Alarm 2 Interrupt	RO		**PMA Standard Alarm 2 Interrupt** An interrupt was generated from either Bit 1.8.B or 1.8.A. An interrupt was generated from status register PMA Standard Status 2: Address 1.8 and the corresponding mask register PMA Transmit Standard Interrupt Mask 2: Address 1.D001. 1b = Interrupt in PMA standard alarms 2.
F	PMA Standard Alarm 1 Interrupt	RO		**PMA Standard Alarm 1 Interrupt** An interrupt was generated from Bit 1.1.2. An interrupt was generated from status register PMA Standard Status 1: Address 1.1 and the corresponding mask register PMA Transmit Standard Interrupt Mask 1: Address 1.D000. 1b = Interrupt in PMA standard alarms 1.

10.6.50 Global Chip-Wide Vendor Interrupt Flags: Address 1E.FC01

Bit	Name	Type	Default	Description
0	Global Alarms 3 Interrupt	RO		**Global Alarms 3 Interrupt** An interrupt was generated from status register Global Alarms 3: Address 1E.CC02 and the corresponding mask register Global Interrupt Mask 3: Address 1E.D402. 1b = Interrupt in Global alarms 3.
1	Global Alarms 2 Interrupt	RO		**Global Alarms 2 Interrupt** An interrupt was generated from status register Global Alarms 2: Address 1E.CC01 and the corresponding mask register Global Interrupt Mask 2: Address 1E.D401. 1b = Interrupt in Global alarms 2.
2	Global Alarms 1 Interrupt	RO		**Global Alarms 1 Interrupt** An interrupt was generated from status register Global Alarms 1: Address 1E.CC00 and the corresponding mask register Global Interrupt Mask 1: Address 1E.D400. 1b = Interrupt in Global alarms 1.
B:3	Reserved	RSV		Reserved. Do not modify.
				1b =
C	Auto-Negotiation Vendor Alarm Interrupt	RO		**Auto-Negotiation Vendor Alarm Interrupt** An auto-negotiation alarm was generated. See Auto-Negotiation Vendor Global Interrupt Flags 1: Address 7.FC00. 1b = Interrupt in auto-negotiation vendor specific alarm.
D	Reserved	RSV		Reserved. Do not modify.
E	PCS Vendor Alarm Interrupt	RO		**PCS Vendor Alarm Interrupt** A PCS alarm was generated. See PCS Vendor Global Interrupt Flags 1: Address 3.FC00. 1b = Interrupt in PCS vendor specific alarm.
F	PMA Vendor Alarm Interrupt	RO		**PMA Vendor Alarm Interrupt** A PMA alarm was generated. See PMA Vendor Global Interrupt Flags 1: Address 1.FC00. 1b = Interrupt in PMA vendor specific alarm.

10.6.51 Global Interrupt Chip-Wide Standard Mask: Address 1E.FF00

Bit	Name	Type	Default	Description
0	All Vendor Alarms Interrupt Mask	RW PD	1b	**All Vendor Alarms Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
5:1	Reserved	RSV		Reserved. Do not modify.
6	GbE Standard Alarms Interrupt Mask	RW PD	0b	**GbE Standard Alarms Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
7	Auto-Negotiation Standard Alarms 2 Interrupt Mask	RW PD	0b	**Auto-Negotiation Standard Alarms 2 Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
8	Auto-Negotiation Standard Alarms 1 Interrupt Mask	RW PD	0b	**Auto-Negotiation Standard Alarms 1 Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
A:9	Reserved	RSV		Reserved. Do not modify.
B	PCS Standard Alarm 3 Interrupt Mask	RW PD	0b	**PCS Standard Alarm 3 Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
C	PCS Standard Alarm 2 Interrupt Mask	RW PD	0b	**PCS Standard Alarm 2 Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
D	PCS Standard Alarm 1 Interrupt Mask	RW PD	0b	**PCS Standard Alarm 1 Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
E	PMA Standard Alarm 2 Interrupt Mask	RW PD	0b	**PMA Standard Alarm 2 Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F	PMA Standard Alarm 1 Interrupt Mask	RW PD	0b	**PMA Standard Alarm 1 Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.

10.6.52 Global Interrupt Chip-Wide Vendor Mask: Address 1E.FF01

Bit	Name	Type	Default	Description
0	Global Alarms 3 Interrupt Mask	RW PD	1b	**Global Alarms 3 Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
1	Global Alarms 2 Interrupt Mask	RW PD	0b	**Global Alarms 2 Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
2	Global Alarms 1 Interrupt Mask	RW PD	0b	**Global Alarms 1 Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
A:3	Reserved	RSV		Reserved. Do not modify.
B	GbE Vendor Alarm Interrupt Mask	RW PD	0b	**GbE Vendor Alarm Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
C	Auto-Negotiation Vendor Alarm Interrupt Mask	RW PD	0b	**Auto-Negotiation Vendor Alarm Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
D	Reserved	RSV		Reserved. Do not modify.
E	PCS Vendor Alarm Interrupt Mask	RW PD	0b	**PCS Vendor Alarm Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.
F	PMA Vendor Alarm Interrupt Mask	RW PD	0b	**PMA Vendor Alarm Interrupt Mask** 0b = Disable interrupt generation. 1b = Enable interrupt generation.

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

Chapter 11 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 fill 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 (BMC).

This chapter describes the supported management interfaces and hardware configurations for platform system management. It describes the interfaces to an external BMC, the partitioning of platform manageability among system components, and the functionality provided by the X550 in each platform configuration.

11.1 Pass-Through (PT) Functionality

Pass-Through (PT) is the term used when referring to the process of sending and receiving Ethernet traffic over the sideband interface. The X550 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 BMC. See Figure 11-1.

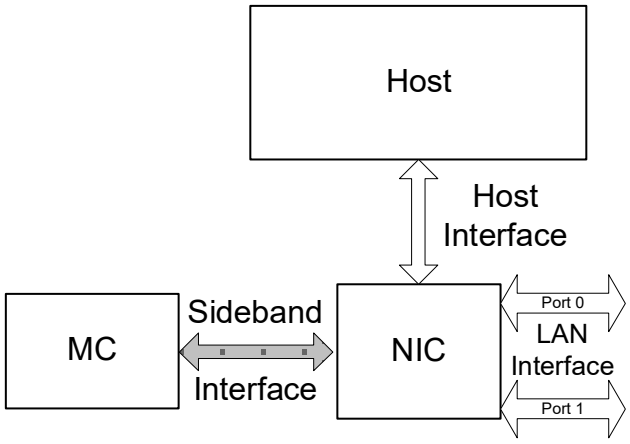


Figure 11-1. Sideband Interface

The sideband interface provides a mechanism by which the X550 can be shared between the host and the BMC. By providing this sideband interface, the BMC can communicate with the LAN without requiring a dedicated Ethernet controller. The X550 supports three sideband interfaces:

- SMBus
- NC-SI
- PCIe (together with MCTP) - when the system is up.

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 X550 processing capabilities and can sustain any network bandwidth. The X550 should support MCTP over PCIe pass though traffic at a rate of up to 1 Mb/s. Only one mode of sideband can be active at any given time. The configuration is done using an NVM setting.

The maximal packet size supported for traffic received from the LAN to the BMC is 1518 bytes and additional VLAN or E-tag. For traffic from the BMC to the LAN the maximal 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.

11.1.1 Supported Topologies

The X550 can support some management topologies. The following connections are available:

- Single connection via legacy SMBus (See Section 11.5).
- Single connection via NC-SI over RMII (See Section 11.6)
- Single connection via NC-SI over MCTP. This connection can be over SMBus, PCI Express or both. This connection can be used either for pass-through or for control only (See Section 11.7).

The topology used is defined in the *Redirection Sideband Interface* field in the *Common Firmware Parameters* NVM word (see Section 6.2.16) and is common to all the ports in the device.

11.1.2 Pass-Through Packet Routing

When an Ethernet packet reaches the X550, it is examined and compared to a number of configurable filters. These filters are configurable by the BMC 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 BMC. Otherwise it is not passed.

The packet filtering process is described in Section 11.3.

11.2 Components of the Sideband Interface

There are two components to a sideband interface:

- Physical Layer
- Logical Layer

11.2.1 Physical Layer

This is the electrical connection between the X550 and BMC.

11.2.1.1 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 X550 to notify the BMC that there is data available for reading. 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* in *SMBus Notification Timeout and Flags* NVM word.

11.2.1.1.1 PEC Support

SMBus transactions can be protected by using Packet Error Code (PEC). Packet Error Checking, whenever 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.

Table 11-1 describes the behavior of the device in each PEC configured mode for transactions directly handled by the hardware upon reception of packets with or without PEC.

Table 11-1. SMBus PEC Modes[1]

SMBus Transaction (Relative to the X550)	X550 PEC Mode	Target PEC Mode	
		PEC Enabled	PEC Disabled
Master Write[2]	Enabled	(A) Target ACKs the PEC byte.	(A) Target NACKs the PEC byte.
	Disabled	(A) Target receives stop before expected PEC byte.	(A) PEC byte is not expected.
Slave 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 Slave.
	Disabled	(A) Target ACKs last data byte; PEC byte is 0xFF.	(A) Target NACKs last data byte and generates Stop after that.

Table 11-1. SMBus PEC Modes[1] [continued]

SMBus Transaction (Relative to the X550)	X550 PEC Mode	Target PEC Mode	
		PEC Enabled	PEC Disabled
Slave Read[4]	Enabled	(A) Target sends PEC byte; PEC byte is ACKed by Slave.	(A) Target does not send PEC byte and generates Stop after that.
	Disabled	(R) Target sends PEC byte; PEC byte is NACKed by Slave.	(A) Target does not send PEC byte and generates Stop after that.

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 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 device.

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.

11.2.1.2 NC-SI

The X550 uses the DMTF standard Sideband Interface. This interface consists of 6 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.

11.2.1.3 PCIe

The X550 uses the VDMs (Vendor Defined Messages) 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.

11.2.2 Logical Layer

11.2.2.1 Legacy SMBus

The protocol layer for SMBus consists of commands the BMC issues to configure filtering for the X550 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 X550 is Intel proprietary. The Legacy SMBus protocol is described in Section 11.5.

11.2.2.2 NC-SI

The DMTF also 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 X550 are discussed in Section 11.6.3. 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 11.2.1.2.
2. MCTP over PCIe.This protocol allows 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 11.7.
3. MCTP over SMBus. As described above, this layer is paired with the MCTP over PCIe to support Sx modes.

The X550 exposes one NC-SI package with up to two channels, one per port.

If the number of ports available changes, the package moves to initialization required state and should be re-enumerated. If the transition occurs while one of the ports is in Keep PHY link up state, the transition is delayed until the Keep PHY link up state is cleared.

The X550 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.

11.2.2.2.1 Package ID Setting

The package ID can be set either from the *Package ID* field in the *NC-SI Configuration 1* NVM word (Section 6.2.17.5), or from an SDP pin. If set from SDP, the Package ID is {0,SDP0_0 value, 0}. The mode used is set by the *NC-SI Package ID from SDP* field in the *NC-SI Configuration 2* NVM word (Section 6.2.17.6). Note that when the package ID is set from the SDP pin, SDP0_0 should be set as input is *ESDP.SDP0_IODIR* field.

The internal channel ID matches the lowest numbered PCIe function number through which this port is exposed to the host.

11.2.2.2.2 Channel ID Mapping

The mapping of the channels to physical ports is according to the *NC-SI Channel to Port Mapping* NVM word (Section 6.2.17.17) if the *Table Valid* bit is set. If this bit is not set, the following algorithm should be used:

```
CHANNEL_ID = 0
If (FACTPS. LAN_FUNCTION_SEL == 0 )  // no swap
    For (x = 0 to 1) {
        PORT = x;
        If (FACTPS.LANx_VALID) NC-SI_Channel[PORT] = CHANNEL_ID++;
    }
Else  // swap
    For (x = 0 to 1) {
        PORT =  1 - x;
        If (FACTPS.LANx_VALID) NC-SI_Channel[PORT] = CHANNEL_ID++;
    }
```

11.3 Packet Filtering

Since both the host operating system and BMC use the X550 to send and receive Ethernet traffic, there needs to be a mechanism by which incoming Ethernet packets can be identified as those that should be sent to the BMC rather than the host operating system.

There are two different types of filtering available. The first is filtering based upon the MAC Address. With this filtering, the BMC has at least one dedicated MAC Address and incoming Ethernet traffic with the matching MAC Address(es) are passed to the BMC. This is the simplest filtering mechanism to utilize and it allows an BMC to receive all types traffic (including, but not limited to, IPMI, NFS, HTTP etc).

The other mechanism available utilizes a highly configurable mechanism by which packets can be filtered using a wide range of parameters. Using this method, the BMC can share a MAC Address (and IP Address, if desired) with the host OS and receive only specific Ethernet traffic. This method is useful if the BMC is only interested in specific traffic, such as IPMI packets.

11.3.1 Manageability Receive Filtering

This section describes the manageability receive packet filtering flow. Packet reception by the X550 can generate one of the following results:

- Discarded.
- Sent to host memory.
- Sent to the external BMC.
- Sent to both the BMC and host memory.

The decisions regarding forwarding of packets to the host and to the BMC are separate and are configured through two sets of registers. However, the BMC may define some types of traffic as exclusive. This traffic is forwarded only to the BMC, even if it passes the filtering process of the host. These types of traffic are defined using the MNGONLY register (Section 8.2.2.19.8).

An example of packets that might be necessary to send exclusively to the BMC might be specific TCP/UDP ports of a shared MAC Address or a MAC Address dedicated to the BMC. If the BMC configures the manageability filters to send these ports to the BMC, it should configure the settings to not send them to the host, otherwise, these ports are received and handled by the host operating system.

The BMC controls the types of packets that it receives by programming receive manageability filters. The following filters are accessible to the BMC:

Table 11-2. Filters Accessible to BMC

Filters	Functionality	When Reset
Filters Enable	General configuration of the manageability filters	LAN_PWR_GOOD
Manageability Only	Enables routing of packets exclusively to the manageability.	LAN_PWR_GOOD
Manageability Decision Filters [7:0]	Configuration of manageability decision filters	LAN_PWR_GOOD
MAC Address [3:0]	Four unicast MAC manageability addresses	LAN_PWR_GOOD
VLAN Filters [7:0]	Eight VLAN tag values	LAN_PWR_GOOD
UDP/TCP Port Filters [15:0]	16 destination port values	LAN_PWR_GOOD
Flexible 128 bytes TCO Filter	Length and values for one flex TCO filter	LAN_PWR_GOOD
IPv4 and IPv6 Address Filters [3:0]	IP Address for manageability filtering	LAN_PWR_GOOD

All filtering capabilities are available on both the NC-SI and legacy SMBus interfaces. However, in NC-SI mode, to program part of the capabilities, the Intel OEM commands described in Section 11.6.3 should be used.

All filters are reset only on Internal Power On Reset. Register filters that enable filters or functionality are also reset by firmware reset in NC-SI mode. These registers can be loaded from the NVM following a reset in SMBus mode. See Section 6.2.16.3 for description of their location in the NVM map.

The high-level structure of manageability filtering is done using two steps.

1. The packet is parsed and fields in the header are compared to programmed filters.
2. A set of decision filters are applied to the result of the first step.

Some general rules apply:

- Fragmented packets are passed to manageability but not parsed beyond the IP header.
- Packets with L2 errors (CRC, alignment, etc.) are not forwarded to the BMC.
- Packets longer than 2KB are filtered out.
- The filtering relates only to the outer L3/L4 header. In tunneled packets, the inner L2, L3 and L4 header parameters are not considered for manageability filtering.

The following sections describe the manageability filtering, followed by the final filtering rules.

The filtering rules are created by programming the decision filters as described in Section 11.3.4.

11.3.2 L2 Filters

11.3.2.1 MAC and VLAN Filters

The manageability MAC filters allow comparison of the Destination MAC Address to one of 4 filters defined in the MMAH and MMAL registers.

The VLAN filters allow comparison of the 12-bit VLAN tag to one of 8 filters defined in the MAVTV registers.

11.3.2.2 EtherType Filters

Manageability L2 EtherType filters allow filtering of received packets based on the Layer 2 EtherType field. The L2 type field of incoming packets is compared against the EtherType filters programmed in the Manageability EtherType Filter (METF; up to 4 filters); the result is incorporated into decision filters.

Each Manageability EtherType filter can be configured as pass (positive) or reject (negative) using a polarity bit. For the reverse polarity mode to be effective and block certain type of packets, the EtherType filter should be part of all the enabled decision filters.

An examples for usage of L2 EtherType filters is to determine the destination of 802.1X control packets. The 802.1X protocol is executed at different times in either the management controller or by the Host. L2 EtherType filters are used to route these packets to the proper agent.

In addition to the flexible EtherType filters, the X550 supports 2 fixed EtherType filters used to block NC-SI control traffic (0x88F8) and flow control traffic (0x8808) from reaching the manageability interface. The NC-SI EtherType is used for communication between the management controller on the NC-SI link and the X550. Packets coming from the network are not expected to carry this EtherType and such packets are blocked to prevent attacks on the management controller. Flow control packets should be consumed by the MAC and as such are not expected to be forwarded to the management interface.

11.3.3 L3/L4 Filtering

The manageability filtering stage combines checks done at previous stages with additional L3/L4 checks to make a the decision on whether to route a packet to the BMC. The following sections describe the manageability filtering done at layers L3/L4 and final filtering rules.

Note: The L3 filters do not match tunneled fields (L3, L4 or ARP).

11.3.3.1 ARP Filtering

ARP filtering — The X550 supports filtering of ARP request packets (initiated externally) and ARP responses (to requests initiated by the BMC).

In legacy SMBus mode, the ARP filters can be used as part of the ARP offload described in Section 11.5.4. ARP offload is not specifically available when using NC-SI. However, the general filtering mechanism is utilized to filter incoming ARP traffic as requested using the Enable Broadcast Filtering NC-SI command.

To limit the reception of ARP packets to the ARP packets dedicated to this station (ARP target IP = BMC IP), the ARP request/response filter can be bind to specific IP Address, by setting both the ARP Request/Response and the IP AND bits in an MDEF filter, as the *IP* bit is set also if there is a match on the target IP (the TPA field in the ARP packet) of an ARP request or an ARP response.

Note: If the OR section of the MDEF is all cleared and one of the IPv4 address are set, ARP packets matching the IP Address pass the filter. If these packets should be dropped, an OR EtherType filter with a a value of 0x0800 (IPv4) should be added.

11.3.3.2 Neighbor Discovery Filtering and MLD

The X550 supports filtering of the following ICMPv6 packets.

Neighbor Discovery packets:

0x86 (134d) - Router Advertisement.
0x87 (135d) - Neighbor Solicitation.
0x88 (136d) - Neighbor Advertisement.
0x89 (137d) - Redirect.

MLD packets:

0x82 (130d) - MLD Query
0x83 (131d) - MLDv1 Report
0x84 (132d) - MLD Done
0x8F (143d) - MLDv2 Report

The Neighbor Discovery packets has dedicated enables for each type in the decision filters. For MLD, a single enable controls the forwarding of all the MLD packets. This means that either all the MLD packets types are selected for reception or none of them.

11.3.3.3 RMCP Filtering

The X550 supports filtering by fixed destination port numbers, port 0x26F and port 0x298. These ports are IANA reserved for RMCP.

In SMBus mode, there are filters that can be enabled for these ports. When using NC-SI, they are not specifically available. However, the general filtering mechanism can be utilized to filter incoming ARP traffic.

11.3.3.4 Flexible Port Filtering

The X550 implements 16 flex destination port filters. The X550 directs packets whose L4 destination port matches to the BMC. The BMC must ensure that only valid entries are enabled in the decision filters.

11.3.3.5 IP Address Filtering

The X550 supports filtering by destination IP Address using IPv4 and IPv6 address filters. These are dedicated to manageability. The X550 provides four IPv6 address filters or three IPv6 addresses and four IPv4 address filters.

The IPv4 match rises also for ARP packets for which the Target IP matches the IP Address in the MIPAF4 register.

11.3.3.6 Checksum Filtering

If bit MANC.*EN_XSUM_FILTER* is set, the X550 directs packets to the BMC only if they match all other filters previously described as well as pass L3/L4 checksum (if it exists).

The X550 may be instructed to directs packets to the BMC only if they pass L3/L4 checksum (if they exist) in addition to matching other filters previously described.

Enabling the XSUM filter when using the SMBus interface is accomplished by setting the *Enable XSUM Filtering to Manageability* bit (MANC.*EN_XSUM_FILTER*). This is done using the Update Management Receive Filter Parameters command (see Section 11.5.11.1.6), or using Set Common Filters Receive Control Bytes (data 2:4, command 0xC2) (see Section 11.5.11.1.7).

To enable the XSUM filtering when using NC-SI, use the Enable Checksum Offloading command (see Section 11.6.3.15).

11.3.4 Flexible 128 Byte Filter

The X550 provides one flex TCO filter. This filter looks for a pattern match within the first 128 bytes of the packet.

Flex filters are temporarily disabled when read from or written to by the host. Any packet received during a read or write operation is dropped. Filter operation resumes once the read or write access completes.

11.3.4.1 Flexible Filter Structure

The filter is composed of the following fields:

- **Flexible Filter Length** — This field indicates the number of bytes in the packet header that should be inspected. The field also indicates the minimal length of packets inspected by the filter. Packet below that length are not inspected. Valid values for this field are: 8*n, where n=1…16.
- **Data** — This is a set of up to 128 bytes comprised of values that header bytes of packets are tested against.
- **Mask** — This is a set of 128 bits corresponding to the 128 data bytes that indicate for each corresponding byte if is tested against its corresponding byte. The general filter is 128 bytes that the BMC configures; all of these bytes may not be needed or used for the filtering, so the mask is used to indicate which of the 128 bytes are used for the filter.

Each filter tests the first 128 bytes (or less) of a packet, where not all bytes must necessarily be tested.

11.3.4.2 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 described below:

Group	Test Bytes
0x0	0-29
0x1	30-59
0x2	60-89
0x3	90-119
0x4	120-127

   Each group of bytes need to be configured using a separate command, where the group number is given as a parameter. The command has the following parameters:
   - Group number — A 1-byte field indicating the current group addressed.
   - Data bytes — Up to 30 bytes of test-bytes for the current group.

11.3.5 Configuring Manageability Filters

There are a number of pre-defined filters that are available for the BMC to enable, such as ARPs and IPMI ports 298h 26Fh. These are generally enabled by setting the appropriate bit within the MANC register using specific commands.

For more advanced filtering needs, the BMC has the ability to configure a number of configurable filters. It is a two-step process to use these filters. They must first be configured and then enabled.

11.3.5.1 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 BMC and defines which of the L2, VLAN, EtherType and L2/L3 filters participate in decision making. Any packet that passes at least one rule is directed to manageability and possibly to the host.

The inputs to each decision filter are:

- Packet passed a valid management L2 exact address filter.
- Packet is a broadcast packet.
- Packet has a VLAN header and it passed a valid manageability VLAN filter.
- Packet matched one of the valid IPv4 or IPv6 manageability address filters.
- Packet is a multicast packet.
- Packet passed ARP filtering (request or response).
- Packet passed neighbor solicitation filtering.
- Packet passed MLD filtering.
- Packet passed 0x298/0x26F port filter.
- Packet passed a valid flex port filter.
- Packet passed a valid flex TCO filter.
- Packet passed or failed an L2 EtherType filter.
- Packet passed or failed Flow Control or NC-SI L2 EtherType Discard filter.

The structure of each decision filter is shown in Figure 11-2. 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 a register. If all bits are cleared (MDEF/MDEF_EXT = 0x0000), the decision filter is disabled and ignored.
- All enabled AND filters must match for the decision filter to match. An AND filter not enabled in the MDEF/MDEF_EXT registers is ignored. If an AND filter is preceded by a OR filter, at least one of the enabled OR inputs must match for the filter to pass.
- If no OR filter is enabled in the register, the OR filters are ignored in the decision (the filter might still match).
- If one or more OR filters are enabled in the register, at least one of the enabled OR filters must match for the decision filter to match.

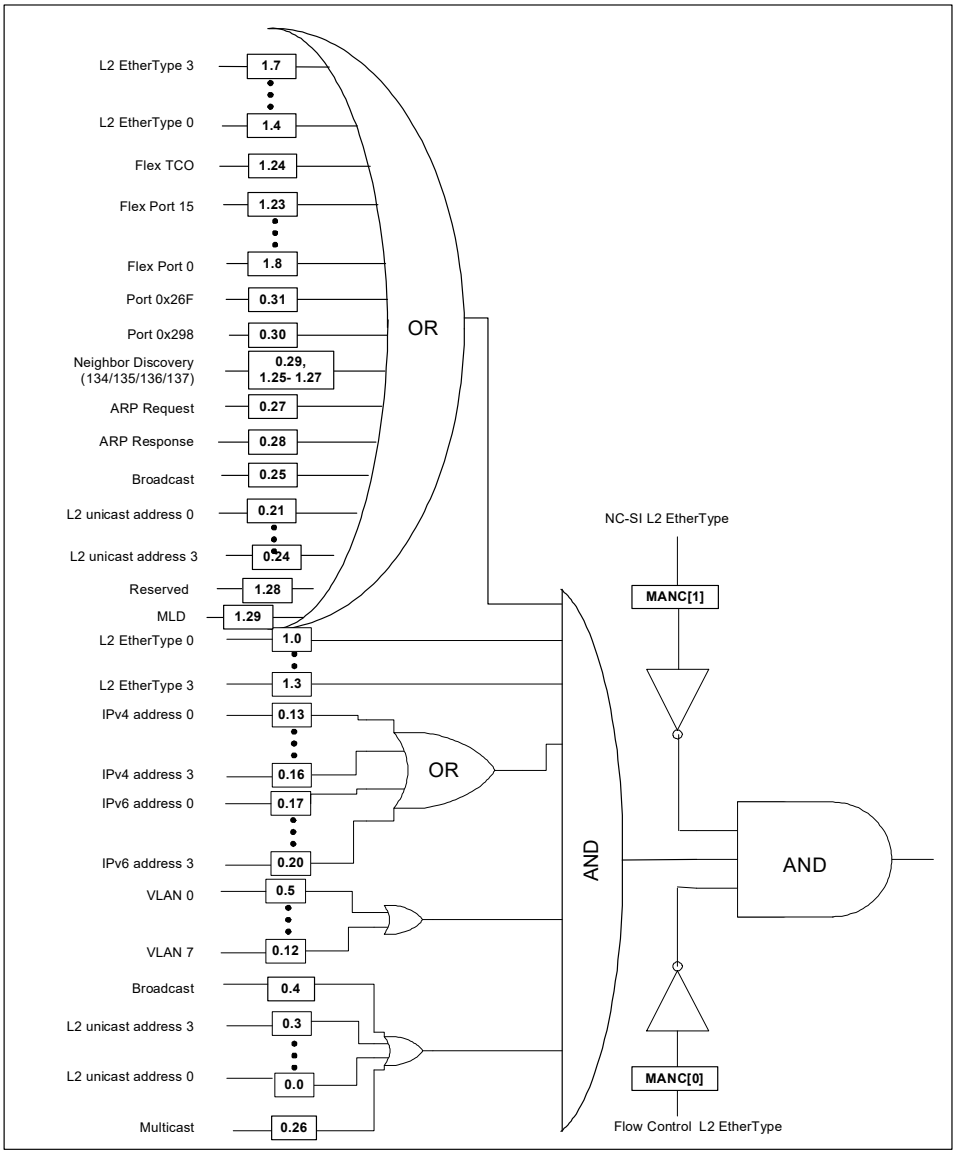


Figure 11-2. Manageability Decision Filters

A decision filter (for any of the 8 filters) defines which of the above inputs is enabled as part of a filtering rule. The BMC programs two 32-bit registers per rule (MDEF[7:0] & MDEF_EXT[7:0]) with the settings as described in Section 8.2.2.19.9 and Section 8.2.2.19.5. A set bit enables its corresponding filter to participate in the filtering decision.

In addition to the controls described above, the MDEF_EXT.*APPLY_TO_HOST_TRAFFIC* and MDEF_EXT.*APPLY_TO_NETWORK_TRAFFIC* bits defines which traffic is compared to this filter. At least one of these bits must be set for the filter to be valid.

If the MDEF_EXT.*APPLY_TO_HOST_TRAFFIC* bit is set, the traffic from the host is candidate for this filter. If the MDEF_EXT.*APPLY_TO_NETWORK_TRAFFIC* bit is set, the traffic from the network is candidate for this filter. If both bits are set, this filter is applied to all traffic.

11.3.5.2 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 BMC may define some types of traffic as exclusive. The behavior for such traffic is defined by the using the bits corresponding to the decision filter in the MNGONLY register (one bit per each of the eight decision rules) and the MDEF_EXT.*APPLY_TO_HOST_TRAFFIC* and MDEF_EXT.*APPLY_TO_NETWORK_TRAFFIC* bits.

Table 11-3 describes the behavior in each case. If one or more filters match the traffic and at least one of the filters is set as exclusive, the traffic is treated as exclusive.

Table 11-3. Exclusive Traffic Behavior

Traffic Source	Filter Matches		Filter Does Not Match
	MNGONLY = 0	MNGONLY = 1	N/A
From network	Traffic is forwarded to the manageability. Traffic is forwarded to the host according to host filtering	Traffic is forwarded only to manageability.	Traffic is forwarded to the host according to host filtering
From host	Traffic is forwarded to the manageability and to the LAN	Traffic is forwarded only to manageability.	Traffic is forwarded to the LAN

Any traffic matching any of the configurable filters (see Section 11.3.5.1) can be used as filters to pass traffic to the host.

Table 11-4. 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 BMC enables these filters by issuing the Update Management Receive Filter Parameters command (see Section 11.5.11.1.6) with the parameter of 0x0F.

The MNGONLY register is also configurable when using NC-SI using the Set Intel Filters — Manageability Only Command (see Section 11.6.3.7.2).

All manageability filters are controlled by the BMC only and not by the LAN device driver.

11.3.5.3 Global Controls

On top of the MDEF filters, the MANC register contains some global controls applied to all the packets to be candidate for manageability filtering:

- Receive Enable bits:
  - The *RCV_TCO_EN* field controls the reception of manageability traffic. It should be set only if one of the following bits is set also.
  - The *EN_BMC2OS* bit controls the reception of manageability traffic from the host.
  - The *EN_BMC2NET* bit controls the reception of manageability traffic from the network.
- VLAN filtering:

  To support the NC-SI VLAN modes the following controls are provided:
  - The *FIXED_NET_TYPE* field controls if only VLAN tagged or VLAN un-tagged traffic is received. If this bit is cleared both types are received. If it is set, only the type described by the *NET_TYPE* field is accepted.
  - If set, the *NET_TYPE* field indicates that only VLAN tagged traffic is received, if cleared only packets without VLAN is accepted. This field is validated by the *FIXED_NET_TYPE* field.

  Both fields relates to the inner VLAN.

11.3.6 Filtering Programming Interfaces

The X550 provides multiple options to program the forwarding filters, depending on the interface used and the level of flexibility needed. Table 11-5 describes the different options and points to the description of the relevant commands.

Table 11-5. Filtering Programming Interfaces

Interface	Flexible/Abstract	Description
NC-SI (over RMII or over MCTP)	Abstract (dedicated MAC Address)	The regular NC-SI commands can be used to allow forwarding based on a dedicated MAC Address. The list of supported commands can be found in Section 11.6.2.1. When using these commands, one of the two other modes can be used to add finer grain filtering.
	Flexible	This interface described in most of the subsections of Section 11.6.3.4. 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 11.5.11.1.7) 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 11.5.11.1.6) can be used to define the exact filtering rules to be applied.

11.3.7 Possible Configurations

This section describes ways of using management filters. Actual usage may vary.

11.3.7.1 Dedicated MAC Packet Filtering

- Select one of the eight rules for dedicated MAC filtering.
- Load Host MAC Address to one of the management MAC Address filters and set the appropriate bit in field 3:0 of the MDEF register.
- Set other bits to qualify which packets are allowed to pass through. For example:
  - Set bit 5 in MDEF to qualify with the first manageability VLAN.
  - Set relevant bits 13 to 20 in MDEF to qualify with a match to one of the IP Addresses.
  - Set any L3/L4 bits (bits 27 to 31 in MDEF and bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

11.3.7.2 Broadcast Packet Filtering

- Select one of the eight rules for broadcast filtering.
- Set bit 25 in MDEF of the decision rule to enforce broadcast filtering.
- Set other bits to qualify which broadcast packets are allowed to pass through. For example:
  - Set bit 5 in MDEF to qualify with the first manageability VLAN.
  - Set relevant bits 13 to 20 in MDEF to qualify with a match to one of the IP Addresses.
  - Set any L3/L4 bits (bits 27 to 31 in MDEF, and bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

11.3.7.3 VLAN Packet Filtering

- Select one of the eight rules for VLAN filtering.
- Set bit 5 to 12 in MDEF to qualify with the relevant manageability VLANs.
- Set other bits to qualify which VLAN packets are allowed to pass through. For example:
  - Set any L3/L4 bits (bits 27 to 31 in MDEF, and bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

11.3.7.4 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 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 X550 supports IPv6 neighbor Discovery protocol. Since the protocol relies on multicast packets, the X550 supports filtering of these packets. IPv6 multicast addresses are translated into corresponding Ethernet multicast addresses in the form of 33-33-xx-xx-xx-xx, where the last 32 bits of address are taken from the last 32 bits of the IPv6 multicast address. As a result, two direct MAC filters can be used to filter IPv6 solicited-node multicast packets as well as IPv6 all node multicast packets.

11.3.7.5 Receive Filtering with Shared IP

When using the Legacy SMBus interface, it is possible to share the host MAC and IP Address with the BMC. This functionality is also available when using base NC-SI using Intel OEM commands.

When the BMC shares the MAC and IP Address with the host, receive filtering is based on identifying specific flows through port allocation. The following setting might be used when using the legacy SMBus interface:

- Select one of the eight rules.
- Set a manageability dedicated MAC filter to the host MAC Address and set the matching bit (0-3) in the MDEF register.
- If VLAN is used for management, load one or more management VLAN filters and set the matching bit (5-12) in the MDEF register.

ARP filter/neighbor Discovery filter is enabled when the BMC is responsible for handling the ARP protocol. Set bit 27 or bit 28 in the MDEF register for this functionality.

11.3.8 Determining Manageability MAC Address

If the BMC wishes to use a dedicated MAC Address or configure the automatic ARP response mechanism (only available in SMBus mode), it may be beneficial for the BMC to be able to determine the MAC Address used by the host.

Both the NC-SI and SMBus interfaces provide an Intel OEM command to read the System MAC Address.

A possible use for this is that the MAC Address programmed at manufacturing time does not increment by one each time, but rather by two. In this way, the BMC can read the System MAC Address and add one to it and be guaranteed of a unique MAC Address.

Determining the IP Address being used by the host is beyond the scope of this document.

11.4 OS-to-BMC Traffic

11.4.1 Overview

Traditionally, the communication between a host and the local BMC is not handled through the network interface and requires a dedicated interface such as an IPMI KCS interface. The X550 allows the host and the local BMC communication via the regular pass-through interface, and thus allow management of a local console using the same interface used to manage any BMC in the network.

When this flow is used, the host sends packets to the BMC through the network interface. The X550 examines these packets and then decides if they should be forwarded to the BMC. On the inverse path, when the BMC sends a packet on the pass-through interface, the X550 checks if it should be forwarded to the network, the host, or both. Figure 11-3 describes the flow for OS-to-BMC traffic for the NC-SI over RMII case. OS2BMC is available also when working over MCTP. It is not supported in legacy SMBus mode.

The OS-to-BMC flow can be enabled using the *OS2BMC enable* field for the relevant port in the *OS 2 BMC configuration* structure of the NVM.

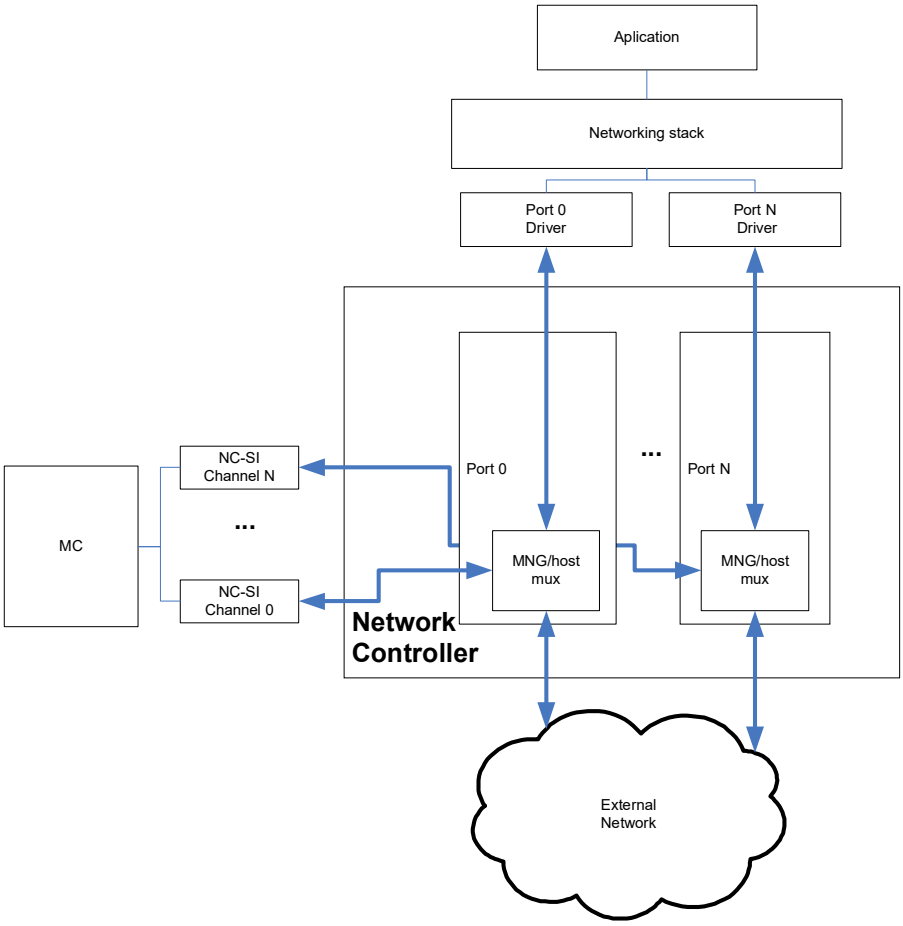


Figure 11-3. OS-to-BMC Diagram

The OS-to-BMC flow is enabled only for ports enabled by the NC-SI "Enable Channel" command or via the *OS-to-BMC Enable* field for the relevant port in the OS-to-BMC configuration structure of the NVM. When the MC uses IPsec for the flow directed to the host, it should make sure the X550 does not offload the IPsec flow by exposing the IP Address used via the *Set IP Address* NC-SI OEM command.

OS2BMC traffic must comply with NC-SI specifications and is therefore limited to maximum sized frames of 1536 bytes (in both directions).

11.4.2 Filtering

When OS-to-BMC traffic is enabled, the filters used for network to BMC traffic are also used for OS-to-BMC traffic. Traffic considered as exclusive to the BMC (Relevant bit in MNGONLY is set) is also considered as exclusive to the BMC when sent from the Host and not forwarded to the network.

11.4.2.1 Handling of OS-to-BMC Packets

All the regular transmit offloads are also available for OS-to-BMC packets.

11.4.2.2 BMC-to-OS Filtering

When OS-to-BMC is enabled, as with regular BMC transmit traffic, the port (OS or network) to which the packet is sent is fixed according to the source MAC Address of the packet. After that, the BMC traffic is filtered according to the L2 Host filters of the selected port (as described in Section 7.1.2). According to the results of the filtering the packet can be forwarded to the OS, the network or both.

The following rules apply to the forwarding of OS packets:

- If *BMC to net* is disabled, all the traffic from the BMC is sent to the Host.
- If *BMC to host* is disabled, all the traffic from the BMC is sent to the network.
- If both *BMC to net* and *BMC to host are enabled,* the packet is forwarded only according to the destination MAC Address and VLAN tag. Unicast packets that matches one of the exact filters (RAH/RAL) are sent only to the Host. Other packets that pass the L2 Host filtering are sent to both the Host and the network. Packets that do not pass the L2 host filtering are sent only to the network.

Note: In virtualization mode, if packet is received by host only due to default pooling (PFVTCTL.*DIS_DEF_POOL* bit is cleared), it is not sent to the host (even if *BMC to net* is disabled).

11.4.2.3 Queuing of Packets Received from the BMC

Packets received from the BMC are queued in the default queue.

11.4.2.4 Offloads of Packets Received from the BMC

Packets received from the BMC and forwarded to the OS do not pass the same path as regular network packets. Thus parts of the offloads provided for the network packets are not available for the BMC packets. Packet received from the BMC are identified by the *RDESC.STATUS.BMC* bit.

The following list describes which offloads are available for BMC packets:

- CRC is checked and removed on the BMC packets.
- The BMC packets are not detected as time sync packet. The *RDESC.STATUS.TS* is always clear for these packets.
- In systems where the double VLAN feature is enabled (CTRL_EXT.*EXTENDED_VLAN* is set), the *VEXT* bit is valid for BMC packets and reflects the presence of a tag with an EtherType matching the value in EXVET register.

  Note: In systems that uses double VLAN, the BMC is expected to send all packets (apart from NC-SI commands) with the outer VLAN included. Failing to do so may cause corruptions to the packet received by the OS

- The *RDESC.ERRORS* field is always cleared for these packets.

  Note: Traffic sent from the BMC does not cause a PME event, even if it matches one of the wake-up filters set by the port.

11.4.3 Blocking of Network to BMC Flow

In some systems the BMC may have its own private connection to the network and may use the X550 port only for the OS-to-BMC traffic. In this case, the BMC to network flow should be blocked while enabling the OS-to-BMC and OS to network flows.

This can be done by clearing the MANC.*EN_BMC2NET* bit for the relevant port. The BMC can control this functionality using the "Enable Network to BMC flow" and "Disable Network to BMC flow" NC-SI OEM commands. This can also be controlled using the *Network to BMC disable* field in the NVM "OS2BMC Configuration Structure".

Notes: When network to BMC flow is blocked and OS-to-BMC flow is enabled, all the traffic from the BMC is sent to the OS without any check. The OS traffic filtering is still done using the regular decision filters.

The NC-SI channel should not be enabled for receive or transmit before at least one of the *EN_BMC2NET* or *EN_BMC2OS* fields is set, unless used for AEN transmissions only. In this case, the channel may be enabled for receive, but all receive filters should be cleared.

11.4.4 OS2BMC and Flow Control

The traffic between the host and manageability uses the same buffers as any loopback traffic. Thus it flows through the transmit buffer and then through the receive buffer. If the transmit buffer is flow controlled, the host to BMC traffic is also stopped. If the receive buffer is full, the traffic is dropped or the transmit is stopped according to the flow control policy of this traffic class.

Packets received to the manageability (either from host or from network) may be dropped if the manageability internal buffers are full.

11.4.5 Statistics

Packets sent from the OS to the BMC should be counted by all statistical counters as packets sent by the OS. If they are sent to both the network and to the BMC, they are counted once.

Packets sent from the BMC to the host are counted as packets received by the host. If they are sent to the host and to the network, they are counted both as received packets and as packet transmitted to the network.

In addition, the X550 supports the following statistical counters that measure just the BMC-to-OS and OS-to-BMC traffic:

- O2BGPTC: OS-to-BMC packets received by BMC
- O2BSPC: OS-to-BMC packets transmitted by OS and received by manageability buffer.
- B2OSPC: BMC-to-OS packets sent by BMC
- B2OGPRC: BMC-to-OS packets received by OS.

The driver can use these statistics to count packets dropped by the X550 during the transfer between the OS and the BMC or the LAN and the BMC as follows:

- Dropped packets in BMC-to-OS path = *B2OSPC - B2OGPRC*
- Dropped packet in BMC to LAN path = *B2OSDPC - (B2OSPC - B2OGPRC)*
- Dropped packets in OS-to-BMC path = *O2BSPC - O2BGPTC*
- Dropped packets in LAN to BMC path = *MNGPDC*

See Section 7.1.7.2 and Section 7.1.5.2 for details of the statistics hierarchy.

11.4.6 OS-to-BMC Enablement

The X550 supports the unified network software model for OS-to-BMC traffic, where the OS-to-BMC traffic is shared with the regular traffic. In this model, there is no need for a special configuration of the OS networking stack or the BMC stack. However, if the link is down, the OS-to-BMC communication is stopped.

To enable OS-to-BMC, either:

- Enable OS2BMC in the *Port Traffic Type* field in the *Traffic Type Parameters* NVM word for the relevant port.
- Send an Enable Network to BMC Command

Note: When OS2BMC is enabled, OS must avoid sending packets longer than 1.5 KB to BMC.

11.5 SMBus Pass-Through 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 pass-through mode.

11.5.1 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.

11.5.2 Pass-Through Capabilities

This section details manageability capabilities the X550 provides while in SMBus mode. Pass-through traffic is carried by the sideband interface as described in Section 11.1.

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 BMC, the X550 provides the following manageability services to the BMC:

- **ARP handling** — The X550 can be programmed to auto-ARP replying for ARP request packets to reduce the traffic over the BMC 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.

11.5.3 Port to SMBus Mapping

The X550 is presented on the SMBus manageability link as two different devices (for example, via two different SMBus addresses on which each device is connected to a different LAN port). There is no logical connection between the devices.

The fail-over between the LAN ports is done by the BMC (by sending/receiving packets through different devices). The status report to the BMC, ARP handling, DHCP, and other pass-through functionality are unique for each port and configured by the BMC.

If the number of enable ports changes, the port enable state is updated, and an alarm is set according to the programmed notification method. Bit 6 in Status data byte 2 of all the currently-enabled functions is set to indicate the change. The BMC can then detect the change in the configuration and do a new ARP flow or try to discover if new ports where added or removed.

11.5.4 Automatic Ethernet ARP Operation

The X550 can offload the Ethernet Address Resolution Protocol (ARP) for the BMC to reduce the bandwidth required on the SMBus link.

Automatic Ethernet ARP parameters are loaded from the NVM when the X550 is powered up or configured through the sideband management interface. The following parameters should be configured 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 X550 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 X550, it replies with an ARP response.

The X550 responds to ARP request targeted to the ARP IP Address with the configured ARP MAC Address. In case that there is no match, the X550 silently discards the packets. If the X550 is not configured to do auto-ARP response, it can be configured to forward the ARP packets to the BMC (which can respond to ARP requests).

When the external BMC uses the same IP and MAC Address of the OS, 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 X550 provides the ability to read the stem MAC Address, allowing the BMC to share the MAC Address. There is no mechanism however provided by the X550 to read the IP Address. The host OS (or an agent within) and BMC must coordinate the sharing of IP Addresses.

11.5.5 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 11-6. 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 11-7 and are fully defined in the SMBus specification.

Table 11-7. Shorthand Field Names

Field Name	Definition
S	SMBus START Symbol
P	SMBus STOP Symbol
PEC	Packet Error Code
A	ACK (Acknowledge)
N	NACK (Not Acknowledge)
Rd	Read Operation (Read Value = 1b)
Wr	Write Operation (Write Value = 0b)

11.5.5.1 SMBus Addressing

The SMBus is presented as up to two SMBus devices on the SMBus (two addresses). All pass-through 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 BMC.

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 X550's power up and resets.

11.5.5.2 SMBus ARP Functionality

The X550 supports the SMBus ARP protocol as defined in the SMBus 2.0 specification. The X550 is a persistent slave address device so its SMBus address is valid after power-up and loaded from the NVM. The X550 supports all SMBus ARP commands defined in the SMBus specification both general and directed.

SMBus ARP capability can be disabled through the NVM.

11.5.5.3 SMBus ARP Flow

SMBus ARP flow is based on the status of two flags:

- **AV (Address Valid)** — This flag is set when the X550 has a valid SMBus address.
- **AR (Address Resolved)** — This flag is set when the X550 SMBus address is resolved (SMBus address was assigned by the SMBus ARP process).

These flags are internal X550 flags and are not exposed to external SMBus devices.

Since the X550 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 X550 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 X550 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 X550 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 X550 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 X550 SMBus address by issuing an Assign Address command. The X550 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 X550 does not respond to the Get UDID General command. Note that all other commands are processed even if the AR flag is set. If the address changed, from the one previously stored in the NVM, the X550 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. This process uses the NVM update flow described in Section 3.4.2.1.

Figure 11-4 shows the X550 SMBus ARP flow.

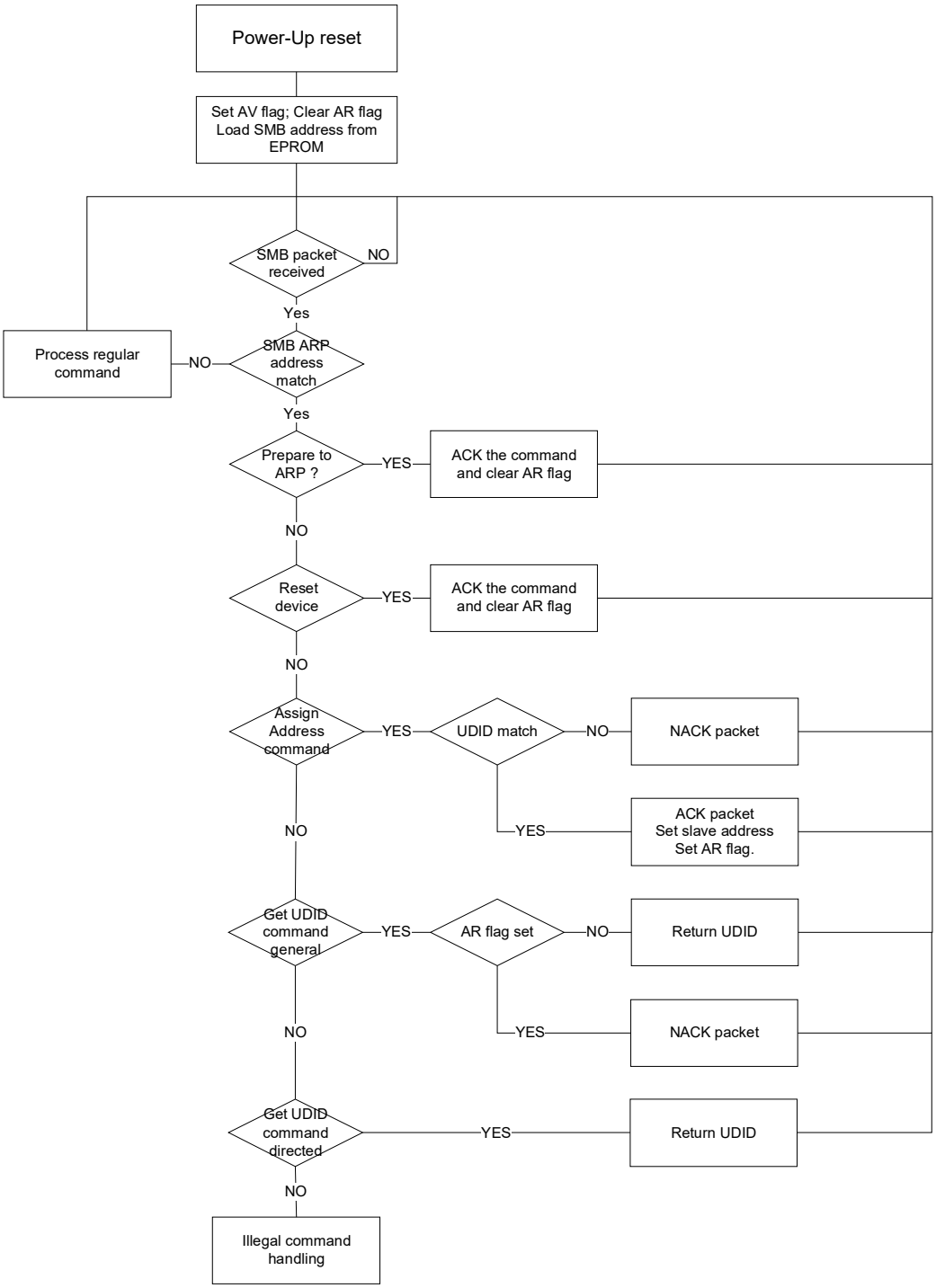


Figure 11-4. SMBus ARP Flow

11.5.5.4 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 11-8. 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	0x1562	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: 0x1562

Interface: Identifies the protocol layer interfaces supported over the SMBus connection by the device.

Bits 3:0 = 0x4 indicates SMBus Version 2.0
Bit 5 (ASF bit) = 1 in MCTP mode.

Subsystem Fields: These fields are not supported and return zeros.

Device Capabilities: Dynamic and Persistent Address, *PEC Support* bit:

7	6	5	4	3	2	1	0
Address Type		Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)	PEC Supported
0b	1b	0b	0b	0b	0b	0b	0b/1b[1]
MSB							LSB

1. The value is set according to the *SMBus Transaction PEC* bit in the NVM

Version/Revision: UDID Version 1, Silicon Revision:

7	6	5	4	3	2	1	0
Reserved (0)	Reserved (0)	UDID Version			Silicon Revision ID		
0b	0b	001b			See the following table		
MSB							LSB

Silicon Revision ID:

Silicon Version	Revision ID
A0	000b
B0	001b

Vendor Specific ID: Four LSB bytes of the device Serial Number combined with the port number. The Serial Number is taken from the NVM and is reflected in the PCI_SERL and PCI_SERH registers.

1 Byte	1 Byte	1 Byte	1 Byte
Port Number, MAC Address, Byte 3[6:0]	MAC Address, Byte 2	MAC Address, Byte 1	MAC Address, Byte 0
MSB			LSB

11.5.5.5 SMBus ARP and Multi-Port

The X550 responds as two SMBus devices having two sets of AR/AV flags (one for each port). The X550 responds two 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 different ports in the *Version ID* field. The X550 first responds as port 0, and only when an address is assigned, then starts responding as port 1 to the Get UDID command.

11.5.5.6 Concurrent SMBus Transactions

The SMBus interface is single threaded. Thus, concurrent SMBus transactions are not permitted. Once a transaction is started, it must be completed before additional transaction can be 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 11.5.11.2.
- The single write SMBus commands described in Section 11.5.11.1.

11.5.6 SMBus Notification Methods

The X550 supports three methods of notifying the BMC that it has information that needs to be read by the BMC:

- SMBus alert - Refer to Section 11.5.6.1.
- Asynchronous notify - Refer to Section 11.5.6.2.
- Direct receive - refer to Section 11.5.6.3.

The notification method used by the X550 can be configured from the SMBus using the Receive Enable command (Section 11.5.11.1.3). The default method is set by the NVM in the *Notification Method* field in LAN Receive Enable 1 (Section 6.2.18.48).

Note: The SMBus notification method used must be the same for all ports.

The following events cause the X550 to send a notification event to the BMC:

- Receiving a LAN packet that is designated to the BMC.
- The firmware was reset and requires re-initialization.
- Receiving a Request Status command from the BMC initiates a status response.
- The X550 is configured to notify the BMC 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
  - Thermal Sensor Event

There can be cases where the BMC is hung and not responding to the SMBus notification. The X550 has a time-out value (defined in the NVM) to avoid hanging while waiting for the notification response. If the BMC 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 BMC cannot modify this value.

11.5.6.1 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 X550 asserts this signal each time it has a message that it needs the BMC 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 X550 can be connected on the same alert pin. As a result, the BMC needs a mechanism to distinguish between the alert sources.

The BMC can respond to the alert by issuing an ARA Cycle command to detect the alert source device. The X550 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 BMC issues a read command to retrieve the X550 message.

Some BMCs do not implement the ARA cycle transaction. These BMCs respond to an alert by issuing a Read command to the X550 (0xC0/0xD0 or 0xDE). The X550 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 X550 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. If two ports have events to notify, the second alert is asserted only after the first event is handled. Hence, if an ARA cycle is not sent, a read status transaction should be done to all the ports in event of an ALERT_N assertion.

The ARA cycle is an SMBus receive byte transaction to SMBus Address 0001-100b. 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	

If the BMC does not react to the alert in the delay defined by the *SMBus Notification Timeout* NVM field (Section 6.2.17.3), the ALERT pin is de-asserted and the Rx packet indication in the status word is cleared (and packet is dropped)

11.5.6.2 Asynchronous Notify Method

When configured using the asynchronous notify method, the X550 acts as a SMBus master and notifies the BMC of one of the events listed in Section 11.5.6 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 11.5.11.1.3) 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
	BMC 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 6.2.18.48 and Section 6.2.18.49).

If the BMC does not read the status in the delay defined by the *SMBus Notification Timeout* NVM field (Section 6.2.17.3), the Rx packet indication in the status word is cleared (and packet is dropped).

11.5.6.3 Direct Receive Method

If configured, the X550 has the capability to send a message it needs to transfer to the external BMC as a master over the SMBus instead of alerting the BMC 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 BMC in the Block Read command. The opcode that the X550 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	...
	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	

11.5.7 Receive Pass-Through Flow

The X550 is used as a channel for receiving packets from the network link and passing them to the external BMC. The BMC configures the X550 to pass these specific packets to the BMC. Once a full packet is received from the link and identified as a manageability packet that should be transferred to the BMC, the X550 starts the receive TCO flow to the BMC.

The X550 uses the SMBus notification method to notify the BMC 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 X550 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 BMC notification method, the X550 notifies the BMC on each fragment of a multi-fragment packet. When asynchronous notify is selected as the BMC notification method, the X550 notifies the BMC only on the first fragment of a received packet. It is the BMC'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 BMC causes the fragment to be re-transmitted to the BMC on the next Receive Packet command.

The maximum size of the received packet is limited by the X550 to 1536 bytes. Packets larger than 1536 bytes are silently discarded. Any packet smaller than 1536 bytes is processed.

11.5.8 Transmit Pass-Through Flow

The X550 is used as the channel for transmitting packets from the external BMC to the network link. The network packet is transferred from the BMC over the SMBus and then, when fully received by the X550, 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 X550 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 specification is 64 bytes. The X550 pads packets that are less than 64 bytes to meet the specification requirements (there is no need for the external BMC to pad packets less than 64 bytes). If the packet sent by the BMC is larger than 1536 bytes, the X550 silently discards the packet.

The X550 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 BMC (the X550 does not change any field in the transmitted packet, other than adding padding and CRC bytes).

If the network link is down when the X550 has received the last fragment of the packet from the BMC, 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 X550 waits for the last fragment to end to see whether the network link is up again.

11.5.8.1 Transmit Errors in Sequence Handling

Once a packet is transferred over the SMBus from the BMC to the X550, 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 11-9 lists the different flag options in transmit packet transactions.

Table 11-9. 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 on, 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.

11.5.8.2 TCO Command Aborted Flow

The X550 indicates to the BMC an error or an abort condition by setting the *TCO Abort* bit in the general status. The X550 might also be configured to send a notification to the BMC (see Section 11.5.11.1.3.3).

Following is a list of possible error and abort conditions:

- Any error in the SMBus protocol (NACK, SMBus timeouts, etc.).
- If the BMC does not respond until the notification timeout (programmed in the EEPROM) 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 X550 does not have space to store the transmitted packet from the BMC (in its internal buffer space) before sending it to the link, the packet is discarded and the external BMC is notified via the *Abort* bit.
- Error in the *F*/*L* bit sequence during multi-fragment transactions.
- An internal reset to the X550's firmware.

11.5.9 SMBus Link State Control

While in SMBus mode, the default setting of the link is defined by the *Enable All PHYs in D3 N* bit in *Common Firmware Parameters* 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, the link may move back to the default defined by the *Enable All PHYs in D3 N* if not needed for other purposes.

Note: When the link is taken down due to *RCV_EN* being cleared the transmit traffic from BMC is also stopped.

Note: Before transitioning to D3 it is the responsibility of the driver to request the PHY to be active for wake-up activities.

11.5.10 SMBus ARP Transactions

All SMBus ARP transactions include the PEC byte.

11.5.10.1 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	

11.5.10.2 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	

11.5.10.3 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	

11.5.10.4 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 the X550 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	

11.5.10.5 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	···
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 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.

11.5.11 SMBus Pass-Through Transactions

This section details commands (both read and write) that the X550 SMBus interface supports for pass-through.

11.5.11.1 Write SMBus Transactions

This section details the commands that the BMC can send to the X550 over the SMBus interface. The SMBus write transactions table lists the different SMBus write transactions supported by the X550.

TCO Command	Transaction	Command	Fragmentation	Section Number
Transmit Packet	Block Write	First: 0x84 Middle: 0x04 Last: 0x44	Multiple	11.5.11.1.1
Transmit Packet	Block Write	Single: 0xC4	Single	11.5.11.1.1
Request Status	Block Write	Single: 0xDD	Single	11.5.11.1.2
Receive Enable	Block Write	Single: 0xCA	Single	11.5.11.1.3
Force TCO	Block Write	Single: 0xCF	Single	11.5.11.1.4
Management Control	Block Write	Single: 0xC1	Single	11.5.11.1.5
Update MNG RCV Filter Parameters	Block Write	Single: 0xCC	Single	11.5.11.1.6
Set Common Filters	Block Write	Single: 0xC2	Single	11.5.11.1.7
Clear All Filters	Block Write	Single: 0xC3	Single	11.5.11.1.8
Set Thermal Sensor Configuration	Block Write	Single: 0xCB (index = 1)	Single	11.5.11.1.9
Perform Thermal Sensor Action	Block Write	Single: 0xCB(index = 2)	Single	11.5.11.1.10

11.5.11.1.1 Transmit Packet Command

The Transmit Packet command behavior is detailed in Section 11.5.8. 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				

11.5.11.1.2 Request Status Command

An external BMC can initiate a request to read the X550 manageability status by sending a Request Status command. When received, the X550 initiates a notification to an external BMC when status is ready. After this, the external controller is able to read the status, by issuing a read status command (see Section 11.5.11.2.2).

The format is as follows:

Function	Command	Byte Count	Data 1
Request Status	0xDD	1	0

11.5.11.1.3 Receive Enable Command

The Receive Enable command is a single fragment command used to configure the X550. 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 BMC (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 BMC. 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.

Field	Bit(s)	Description
EN_ARP_RES	3	Enable ARP Response. 0b = Disable the X550 ARP response — The X550 treats ARP packets as any other packet, for example, packet is forwarded to the BMC if it passed other (non-ARP) filtering. 1b = Enable the X550 ARP response — The X550 automatically responds to all received ARP requests that match the IP Address programmed by the BMC. The BMC IP Address is provided as part of the Receive Enable message (bytes 8:11). If a short version of the command is used, the X550 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, the X550 uses the IP Address configured in the NVM as ARP Response IPv4 Address in the pass-through LAN configuration structure. If the *CBDM* bit is set, the X550 uses the BMC dedicated MAC Address in ARP response packets. If the *CBDM* bit is not set, the BMC uses the host MAC Address. When the ARP offload feature is activated, the X550 uses the following registers to filter the ARP traffic addressed to the BMC. BMC should not modify these registers: Manageability Decision Filter – MDEF6 (and corresponding bit 6 in Management Only traffic Register – *MNGONLY*). IPv4 address - MIPAF4[3]
NM	5:4	Notification Method. Define the notification method the X550 uses. 00b = SMBUS Alert. 01b = Asynchronous notify. 10b = Direct receive. 11b = Not supported. 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 BMC Dedicated MAC Address. ***Note:*** This bit should be 0b when the *RCV_EN* bit (bit 0) is not set. 0b = The X550 shares the MAC Address for MNG traffic with the host MAC Address, which is specified in NVM words 0x0-0x2. 1b = The X550 uses the BMC dedicated MAC Address as a filter for incoming receive packets. The BMC MAC Address is set in bytes 2-7 in this command. If a short version of the command is used, the X550 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 X550 uses the following registers to filter in all the traffic addressed to the BMC MAC. BMC should not modify these registers: Manageability Decision Filter – MDEF7 (and corresponding bit 7 in Management Only traffic Register – *MNGONLY*) Manageability MAC Address Low – *MMAL[3]* Manageability MAC Address High – *MMAH[3]* 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.

11.5.11.1.3.1 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.

11.5.11.1.3.2 Management IP Address (Data Bytes 11:8)

This IP Address is used to filter ARP request packets.

11.5.11.1.3.3 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.

11.5.11.1.3.4 Interface Data (Data Byte 13)

Interface data byte used in asynchronous notification.

11.5.11.1.3.5 Alert Value Data (Data Byte 14)

Alert Value data byte used in asynchronous notification.

11.5.11.1.4 Force TCO Command

This command causes the X550 to perform a TCO reset, TCO isolate, or firmware reset.

TCO Reset:

The Force TCO reset clears the data path (Rx/Tx) of the X550 to enable the BMC to transmit/receive packets through the X550 by assertion of a global reset. Force TCO reset is asserted only to the port related to the SMBus address the command. This command should only be used when the BMC is unable to transmit receive and suspects that the X550 is inoperable. The command also causes the LAN device driver to unload. It is recommended to perform a system restart to resume normal operation.

Firmware Reset:

This command causes re-initialization of all the embedded controller functions and re-load of related NVM words. A firmware reset is achieved by setting the GSCR.*SET_FWRST* Aux bit.

The X550 considers the Force TCO reset command as an indication that the operating system is hung. 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 may yield unexpected results.

11.5.11.1.5 Management Control

This command is used to set generic manageability parameters. The parameters list is shown in Table 11-10. 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 BMC sets is not supported by the X550. The X550 does not NACK the transaction. After the transaction ends, the X550 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 11-10. 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 BMC needs to keep connectivity for a manageability session. Bit [7:1] Reserved. 0b = Disabled. 1b = Enabled.

11.5.11.1.6 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 BMC sets is not supported by the X550, the X550 does not NACK the transaction. After the transaction ends, the X550 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 11-11 lists the different parameters and their content.

Table 11-11. 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 (MANC) register (see Table 11-12). ***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 is never directed to the host memory. Data 2:5 = *MNGONLY* register bytes (see Section 8.2.2.19.8) - 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 anymore. Use 0x68 instead.
VLAN Filters	0x62	3 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	3 to 4 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.
IPv4 Filters	0x64	5 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	7 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 11-11. Management Receive Filter Parameters [continued]

Parameter	Number	Parameter Data
EtherType Filters	0x67	5 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	9 bytes to load the extended decision filters (MDEF_EXT & MDEF) Data 2 — MDEF filter index (valid values are 0...5). 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.

Table 11-12. 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 11.5.11.1.3).
18	Reserved	Reserved.
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	EN_IPV4_FILTER	Enable IPv4 address Filters 0b = These bits store a single IPv6 filter. 1b = The last 128 bits of the MIPAF register are used to store 4 IPv4 addresses for IPv4 filtering.
25	FIXED_NET_TYPE	Fixed Net Type 0b = Both tagged and un-tagged packets can be forwarded to the manageability engine. 1b = Only packets matching the net type defined by the *NET_TYPE* field passes to manageability.
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.

11.5.11.1.7 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 Filters 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	2:4	5:10	11	12	13	14:17
Set Common Filters	0xC2	17	Opcode = 0	Receive Control (see Table 11-13)	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv4 Address

IPv6 Format:

Function	Command	Byte Count	Data						
			1	2:4	5:10	11	12	13	14:29
Set Common Filters	0xC2	29	Opcode = 0	Receive Control (see Table 11-13)	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv6 Address

Table 11-13. Set Common Filters Receive Control Bytes

Byte	Bit	Field	Description
1	0	RCV_EN	Receive TCO Packets Enabled. 1b = 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 X550 and ARP requests to the IP Address set in the command are responded. Mutually exclusive to *Configure ARP/ Neighborhood Filter* bit. If this bit is set, the IP Address must be valid and contain the IP Address of the MC. This bit is ignored if *RCV_EN* is cleared. When set, ARP requests to the BMC IP defined below (unicast or Broadcast) are sent to the internal firmware for processing. ARP response are dropped.
	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 X550 uses. 00b = SMBUS Alert. 01b = Asynchronous notify. 10b = Direct receive. 11b = Not supported.

Table 11-13. Set Common Filters Receive Control Bytes [continued]

Byte	Bit	Field	Description
2	8	CBDM	Configure the BMC Dedicated MAC Address. 0b = The X550 shares the MAC Address for MNG traffic with the host MAC Address, which is specified in RAH/RAL[0] registers 1b = The X550 uses the BMC dedicated MAC Address as a filter for incoming receive packets. The BMC MAC Address is set in bytes 5-10 in this command. This bit is ignored if *RCV_EN* is cleared. If this bit is cleared, at least one of bits 9,10 or 11 must be set. If only bit 9 is set, the IP Address should be different than the address used by the host.
	9	Configure IP Address Filter	Automatically configure an IP Address Filter. If this bit is set, only packets matching this IP Address and the MAC Address defined by the *CBDM* bit are forwarded. This bit is ignored if *RCV_EN* is cleared.
	10	Configure RMCP 26Fh Filter	Automatically configure standard IPMI port 26Fh filters. If this bit is set, only packets matching this port and the MAC Address defined by the *CBDM* bit are forwarded. If the *Configure IP Address Filter* bit is set, only packets matching the IP Address and this port are forwarded. The other port enable bit (11) may add additional forwarding condition. This bit is ignored if *RCV_EN* is cleared.
	11	Configure RMCP 298h Filter	Automatically configure standard IPMI port 298h filter. If this bit is set, only packets matching this port and the MAC Address defined by the *CBDM* bit are forwarded. If the *Configure IP Address Filter* bit is set, only packets matching the IP Address and this port are forwarded. The other port enable bit (10) may add additional forwarding condition. This bit is ignored if *RCV_EN* is cleared.
	12	Configure ARP/ Neighborhood Filter	Automatically Configure filters to allow this traffic to BMC (mutually exclusive to *Auto ARP* bit). If this bit is set, broadcast or unicast ARP packets are forwarded to BMC. These packets may be sent to the host also. In IPv4 mode, setting this bit allows forwarding of broadcast or unicast ARP request and response. If IP Address is set, only request or response to this address is forwarded. In IPv6 mode, setting this bit allows 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 This bit is ignored if *RCV_EN* is cleared.
	13	Configure DHCP port 44h Filter (DHCP server packets)	Automatically configure DHCP port 44 filter to BMC. If this bit is set, broadcast packets matching this port are forwarded. Otherwise, this port is not added to the broadcast filtering option. This bit is ignored if *RCV_EN* is cleared or in IPv6 mode.
	15:14	Reserved	
3	16	Disable Host ARP	Configure ARP Requests and Network Neighborhood packets not to go to host. This bit should be cleared in regular 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 44h) not to go to host. This bit should be cleared in regular operation. Ignored if bit 13 is cleared, *RCV_EN* is cleared, or in IPv6 mode.
	24:18	Reserved	

11.5.11.1.8 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

11.5.11.1.9 Set Thermal Sensor Configuration

This command sets the thermal sensor configuration for threshold *Index* in direction *Direction*, where the threshold is measured in "unit types", and the *Actions* field describes the actions to activate upon crossing of the threshold in the requested direction according to Table 11-39.

Direction is encoded as follows:

0 = High Going
1 = Low Going

Function	Command	Byte Count	Data						
			1	2	3	4:5	6:9	10:13	14
Set Thermal Sensor Configuration	0xCB	14	Sub Opcode (0x1)	Index	Direction	Threshold	Actions "Going High"	Actions "Going Low"	Hysteresis

11.5.11.1.10 Perform Thermal Sensor Action

This command executes actions immediately.

The *Actions* field describes the actions to activate according to Table 11-39.

Function	Command	Byte Count	Data 1	Data 2:5
Perform Thermal Sensor Action	0xCB	5	Sub Opcode (0x2)	Actions

11.5.11.2 Read SMBus Transactions

This section details the pass-through read transactions that the BMC can send to the X550 over SMBus.

SMBus read transactions lists the different SMBus read transactions supported by the X550. All the read transactions are compatible with SMBus read block protocol format.

Table 11-14. SMBus Read Transactions

TCO Command	Transaction	Write Command[1]	Read Command	Opcode	Fragments	Section Number
Receive TCO Packet	Block Read	N/A	0xD0 or 0xC0	First: 0x90 Middle: 0x10 Last[2]: 0x50	Multiple	11.5.11.2.1
Read Status	Block Read	N/A	0xD0 or 0xC0 or 0xDE	Single: 0xDD	Single	11.5.11.2.2
Get System MAC Address	Block Read	N/A	0xD4	Single: 0xD4	Single	11.5.11.2.3
Read Management Parameters	Block Read	0xC1	0xD1	Single: 0xD1	Single	11.5.11.2.4
Read Management RCV Filter Parameters	Block Read	0xCC	0xCD	Single: 0xCD	Single	11.5.11.2.5
Get Controller Information	Block Read	0xD5	0xD5	Single: 0xD5	Single	11.5.11.2.6
Get Common filters	Block Read	0xD3	0xD3	Single: 0xD3	Single	11.5.11.2.7
Read Receive Enable Configuration	Block Read	N/A	0xDA	Single: 0xDA	Single	11.5.11.2.8
Get Thermal Sensor Capabilities	Block Read	0xCB (index = 0)	0xDB (index = 0)	Single: 0xDB	Single	11.5.11.2.9
Get Thermal Sensor Configuration	Block Read	0xCB (index = 1)	0xDB (index = 1)	Single: 0xDB	Single	11.5.11.2.10
Get Thermal Sensor Status	Block Read	0xCB (index = 2)	0xDB (index = 2)	Single: 0xDB	Single	11.5.11.2.11

1. In some commands, a preliminary write command is sent to ask the firmware to prepare the data for the upcoming read command. This column describes the opcode used for the write command.
2. The last fragment of the receive packet is the packet status.

0xC0 or 0xD0 commands are used for more than one payload. If BMC issues these read commands, and the X550 has no pending data to transfer, it always returns as default opcode 0xDD with the X550 status and does not NACK the transaction.

If an SMBus quick read command is received, it is handled as a X550 Request Status command (See Section 11.5.11.1.2 for details).

11.5.11.2.1 Receive TCO LAN Packet Transaction

The BMC uses this command to read packets received on the LAN and its status. When the X550 has a packet to deliver to the BMC, it asserts the SMBus notification for the BMC to read the data (or direct receive). Upon receiving notification of the arrival of a LAN receive packet, the BMC begins issuing a Receive TCO packet command using the block read protocol.

A packet can be transmitted to the BMC in at least two fragments (at least one for the packet data and one for the packet status). As a result, BMC should follow the *F* and *L* bit of the opcode.

The opcode can have these values:

- 0x90 — First Fragment
- 0x10 — Middle Fragment
- 0x50 — Indicates the last fragment of the packet, which contains packet status.

If a notification timeout is defined (in the NVM - Section 6.2.17.3) and the BMC does not finish reading the whole 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 X550.

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 11.5.11.2.1.1		

11.5.11.2.1.1 Receive TCO LAN Status Payload Transaction

This transaction is the last transaction that the X550 issues when a packet received from the LAN is transferred to the BMC. 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 11-15 lists the content of the status data.

Table 11-15. TCO LAN Packet Status Data

Name	Bit(s)	Description
Packet Length	13:0	Packet length including CRC, only 14 LSB bits.
Reserved	15:14	Reserved.
Packet status	31:16	See Table 11-16.
VLAN	47:32	The two bytes of the VLAN header tag.
MNG status	63:48	See Table 11-17. This field should be ignored if Receive is not enabled.

Table 11-16. Packet Status Info

Field	Bit(s)	Description
CRC stripped	0	Insertion of CRC is needed.
VP	1	VLAN Stripped (indicates if the VLAN is part of the packet, or was removed).
LAN#	3:2	Indicates the source port of the packet: 00b = Port 0 01b = Port 1 10b = Reserved 11b = Reserved
Reserved	15:4	Reserved.

Table 11-17. MNG Status

Name	Bits	Description
Pass MNG VLAN Filter Index	2:0	Indicates which of the VLAN filters match the packet.
MNG VLAN Address Match	3	Set when the MNG packet matches one of the MNG VLAN 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.
Decision Filter match	8	Set when there is a match to one of the Decision filters.
Reserved	15:9	Reserved.

11.5.11.2.2 Read Status Command

The BMC should use this command after receiving a notification from the X550 (such as SMBus Alert). The X550 also sends a notification to the BMC in either of the following two cases:

- The BMC asserts a request for reading the status.
- The X550 detects a change in one of the Status Data 1 bits (and was set to send status to the BMC on status change) in the Receive Enable command.

Note: Commands 0xC0/0xD0 are for backward compatibility and can be used for other payloads. The X550 defines these commands in the opcode as well as which payload this transaction is. When the 0XDE command is set, the X550 always returns opcode 0XDD with the X550 status. The BMC reads the event causing the notification, using the Read Status command as follows.

The X550 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 11-18 lists the status data byte 1 parameters.

Table 11-18. Status Data Byte 1

Bit	Name	Description
1:0	Power State	00b = Dr state 01b = D0u state 10b = D0 state 11b = D3 state
2	Reserved	
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].
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.
5	Link Status Indication	0b = LAN link down. 1b = LAN link up.
6	TCO Command Aborted	0b = A TCO command abort event did not occur since the last read status cycle. 1b = A TCO command abort event occurred since the last read status cycle.
7	LAN Port	LAN port. Defines the port that sent status.

1. This indication is asserted when the X550 manageability block reloads the NVM and its internal database is updated to the NVM default values. This is an indication that the external BMC should reconfigure the X550, if other values other than the NVM default should be configured.

Status data byte 2 is used by the BMC to indicate whether the LAN device driver is alive and running, and for thermal events.

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 11-19 lists status data byte 2.

Table 11-19. Status Data Byte 2

Bit	Name	Description
0	Thermal Sensor event	0b = No thermal event 1b = Thermal event asserted.
2:1	Reserved	Reserved.
3	Driver Valid Indication	0b = LAN driver is not alive. 1b = LAN driver is alive.
5:4	Reserved	Reserved.
6	Configuration Change	0 = No change. 1 = Number of enabled ports changed.
7	Reserved	Reserved.

Table 11-20 lists the possible values of bits 2 and 1 and what the BMC can assume from the bits.

Table 11-20. Status Data Byte 2 (Bits 2 and 1)

Previous	Current	Description
Don't Care	00b	Interrupt is not pending (OK).
00b	01b	New interrupt is asserted (OK).
10b	01b	New interrupt is asserted (OK).
11b	01b	Interrupt is waiting for reading (OK).
01b	01b	Interrupt is waiting for reading by the driver for more than one read cycle (not OK). Possible drive hang state.
Don't Care	11b	Previous interrupt was read and current interrupt is pending (OK).
Don't Care	10b	Interrupt is not pending (OK).

BMC reads should consider the time it takes for the LAN device driver to deal with the interrupt (in µs). Note that excessive reads by the BMC can give false indications.

11.5.11.2.3 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 following the MAC Address configured in RAL0, RAH0 registers.

Get system MAC Address format:

Function	Command
Get system MAC Address	0xD4

Data returned from the X550:

Function	Byte Count	Data 1 (Opcode)	Data 2	...	Data 7
Get system MAC Address	7	0xD4	MAC Address MSB	...	MAC Address LSB

11.5.11.2.4 Read Management Parameters

To read the management parameters the BMC should execute two SMBus transactions. The first transaction is a block write that sets the parameter that the BMC 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 BMC 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 BMC command.

The returned data is as follows:

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. When set, 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 X550 only. This parameter is returned on read transaction, if in the previous read command the BMC sets a parameter that is not supported by the X550.
The X550 is not ready	0xFF	Returned by the X550 only, on read parameters command when the data that should have been read is not ready. This parameter has no data. The BMC 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 BMC. The BMC 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 X550 is not ready yet, or that the adequate write command was not given.The BMC should retry the read transaction or send the write transaction.

It is responsibility of the BMC to follow the procedure previously defined. When the BMC sends a Block Read command (as previously described) that is not preceded by a Block Write command with bytecount=1, the X550 sets the parameter number in the read block transaction to be 0xFF.

11.5.11.2.5 Read Management Receive Filter Parameters

To read the management receive filter parameters, the BMC should execute two SMBus transactions. The first transaction is a block write that sets the parameter that the BMC 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 update Management receive filter parameters.

Following the block write the BMC 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 X550:

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 BMC. The BMC 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 X550 is not ready yet, or that the adequate write command was not given. The BMC should retry the read transaction or send the write transaction.

It is BMC responsibility to follow the procedure previously defined. When the BMC sends a Block Read command (as previously described) that is not preceded by a Block Write command with bytecount=1, the X550 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
Decision Filters	0x61	This command is obsolete. Use 0x68 instead.
VLAN Filters	0x62	One byte to define the accessed VLAN tag filter (MAVTV) Data 2 — VLAN Filter number

Parameter	#	Parameter Data
Flex Ports Filters	0x63	One byte to define the accessed manageability flex port filter (MFUTP). Data 2 — Flex Port Filter number
IPv4 Filter	0x64	One byte to define the accessed IPv4 address filter (MIPAF4) Data 2 — IPv4 address filter number
IPv6 Filters	0x65	One byte to define the accessed IPv6 address filter (MIPAF6) Data 2 — Pv6 address filter number
MAC Filters	0x66	One byte to define the accessed MAC Address filters pair (MMAL, MMAH) Data 2 — MAC Address filters pair number (0-3)
EtherType Filters	0x67	1 byte to define EtherType filters (METF) Data 2 — METF filter index (valid values are 0 - 3)
Extended Decision Filter	0x68	1 byte to define the extended decisions filters (MDEF_EXT & MDEF) Data 2 — MDEF filter index (valid values are 0 - 5)
Wrong parameter request	0xFE	Returned by the X550 only. This parameter is returned on read transaction, if in the previous read command the BMC sets a parameter that is not supported by the X550.
The X550 is not ready	0xFF	Returned by the X550 only, on 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.

11.5.11.2.6 Get Controller Information Command

The BMC uses this command to get the controller identification. Each parameter is returned using a different opcode

To read the controller information, the BMC should execute two SMBus transactions. The first transaction is a block write that sets the parameter that the BMC wants to read. The second transaction is block read that read the parameter.

Block write transaction:

Function	Command	Byte Count	Data 1 (Opcode)
Get Controller Information	0xD5	1	Parameter number

Following the block write the BMC 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 X550:

Function	Command	Byte Count	Data 2 (Opcode)	Data 3 -n
Get Controller Information	0xD5	Per Table 11-21	Per Table 11-21	See Table 11-21 for the data for each opcode

Table 11-21. Get Controller Information Data

Opcode	Byte Count	Description	Notes
0x00	5	Data 4:3: Generic device ID. Data 5: Silicon Revision (RevID)	This is the hardware default value, not any value programmed via 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 firmware version	MajorVersion.MinorVersion.Build.SubBuild
0x0F	4	Data 4:3: iSCSI firmware version	
0x10	4	Data 4:3: uEFI firmware version	
0x16	4	Data 4:3: FCoE Boot firmware version	
0x17	6	Data 6:3: Firmware Mini Loader Internal version	
0xFE	2	Wrong parameter request	Returned by the X550 only. This parameter is returned on read transaction, if in the previous read command the BMC sets a parameter that is not supported by the X550.
0xFF	2	The X550 is not ready	Returned by the X550 only, on read parameters command when the data that should have been read is not ready. This parameter has no data. The BMC 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. This value is returned also if opcode 0x00 is received while device is in Dr state.

11.5.11.2.7 Get Common Filters Command

The BMC 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 BMC 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 BMC should issue a block read that reads the filter settings:

Function	Command
Get Common filters	0xD3

Data returned from the X550:

IPv4 Format:

Function	Command	Byte Count	Data						
			1	2:4	5:10	11	12	13	14:17
Get Common Filters	0xD3	18	0	Receive Control (See Table 11-13)	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv4 Address

IPv6 Format:

Function	Command	Byte Count	Data						
			1	2:4	5:10	11	12	13	14:29
Get Common Filters	0xD3	30	0	Receive Control (see Table 11-13)	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv6 Address

If case of error the following answers may be returned:

Function	Command	Byte Count	Data 1
Get Common Filters	0xD3	1	0xFF

This response is by the X550, on read common filter command when the data that should have been read is not ready. This parameter has no data. The BMC should retry the read transaction.

This value is also returned if the byte count is illegal, if the read command is not preceded by a write command, or if the filters were not previously configured with a Set Common Filters command (Section 11.5.11.1.7).

11.5.11.2.8 Read Receive Enable Configuration

The BMC uses this command to read the receive configuration data. This data can be configured when using Receive Enable command or through the NVM.

Read Receive Enable Configuration command format (SMBus Read Block) is as follows:

Function	Command
Read Receive Enable	0xDA

Data returned from the X550:

Function	Byte Count	Data										
		1 (Opcode)	2	3	...	8	9	...	12	13	14	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 11.5.11.1.3.

11.5.11.2.9 Get Thermal Sensor Capabilities Command

The BMC can use this function to read the thermal sensor capabilities. It uses a write command to set the needed index and then a read block command to read the data.

The write command is:

Function	Command	Byte Count	Sub Opcode
Get Thermal Capabilities Request	0xCB	1	0x0

The read command is:

Function	Command
Get Thermal Capabilities Request	0xDB

The data returned from the X550 is:

Function	Byte Count	Data 1 (Opcode)	Data 2 (Sub Opcode)	Data 3 (Version)	Data 4 (Unit Types)	Data 5:6 (Number of Thresholds)	Data 7 (Accuracy)
Get Thermal Sensor Capabilities	26	0xDB	0x0	1	See Table 11-40	See below	

Data 8 (Hysteresis)	Data 9 (M)	Data 10 (B)	Data 11 (K1)	Data 12 (K2)	Data 13:16 (Valid High Going Actions)	Data 17:20 (Valid Low Going Actions)	Data 21:24 (Valid Immediate Actions)	Data 25:26 (Tjunction Max)
See below	See below				See Table 11-39	See Table 11-39	See Table 11-39	See below

- Version should always be 1.
- Unit Types describes the unit types measured according to the encoding in Table 11-40.
- Accuracy describes the accuracy of the reported measurements as follows:
  - 7:4: Max deviation of actual value above measurement in “unit types”.
  - 3:0: Max deviation of actual value below measurement in “unit types”.
- Max hysteresis — Defines the max hysteresis value allowed in the implementation. A value of zero means hysteresis is not supported.
- Number of thresholds describes the number of up and down thresholds as follows:
  - 15:12: Reserved
  - 11:8: Max Number of mixed thresholds.
  - 7:4: Max number of up thresholds.
  - 3:0: Max number of down thresholds.
- Valid Actions “High Going” thresholds — Describes the actions that can be activated by the device as described in Table 11-39 when a high going threshold is crossed or the upper hysteresis of a “High Going” Threshold is crossed.
- Valid Actions “Low going” thresholds — Describes the actions that can be activated by the device as described in Table 11-39 when a low going threshold is crossed or the lower hysteresis of a “Low Going” Threshold is crossed.
- Valid Actions “Immediate” — Describes the actions that can be activated by the device as described in Table 11-39 using a “Perform Thermal Sensor Action” command.
- M, B, K1, K2 — Parameters used to translate the raw data read to a meaningful value according to the following formula:

  $$\mathbf{Y} = (M\mathbf{X} + (B * 10^{K1})) * 10^{K2}$$

  where X is the measured value, and Y is the value presented to the user.

  Note: This formula is compliant with the definition of section 36.3 “Sensor Reading Conversion Formula” in IPMI 2.0
- Tjunction Max - The maximal junction temperature supported (105 °C)

11.5.11.2.10 Get Thermal Sensor Configuration Command

The BMC can use this function to read the thermal sensor configuration for a given threshold. It uses a write command to set the needed index and then a read block command to read the data.

The write command is:

Function	Command	Byte Count	Sub Opcode	Data 1
Get Thermal Configuration Request	0xCB	2	0x1	Index

The read command is:

Function	Command
Get Thermal Configuration Request	0xDB

The data returned from the X550 is:

Function	Byte Count	Data 1 (Opcode)	Data 2 (Sub Opcode)	Data 3 (Index)	Data 4:5 (Threshold)	Data 6:9 (Action "Going High")
Get Thermal Sensor Configuration	15	0xDB	0x1	Index	The programmed threshold	The programmed actions as described in Table 11-39

Data 10:13 (Action "Going Low")	Data 14 (Direction)	Data 15 (Hysteresis)
The programmed actions as described in Table 11-39	See below	See below

The threshold and the Hysteresis are measured in "unit types".

Note: If the Max hysteresis reported in the Thermal Sensor capabilities is zero, the Hysteresis value is ignored.

The *Actions Going High* field describes the actions to activate upon crossing of the threshold for "Going High" thresholds or when crossing the hysteresis for "Going Low" thresholds according to Table 11-39.

The *Actions Going Low* field describes the actions to activate upon crossing of the threshold for "Going Low" thresholds or when crossing the hysteresis for "Going High" thresholds according to Table 11-39.

Direction is encoded as follows:

0 = High Going
1 = Low Going

If the opcode is 0xFF, it means that the data that was requested from the X550 is not ready yet, or that the adequate write command was not given. The BMC should retry the read transaction or send the write transaction.

11.5.11.2.11 Get Thermal Sensor Status Command

The BMC can use this function to read the thermal sensor status. It uses a write command to set the needed index and then a read block command to read the data.

The write command is:

Function	Command	Byte Count	Sub Opcode
Get Thermal Sensor Status Request	0xCB	1	0x2

The read command is:

Function	Command
Get Thermal Sensor Status Request	0xDB

The data returned from the X550 is:

Function	Byte Count	Data 1 (Opcode)	Data 2 (Sub Opcode)	Data 3:4 (Measured Value)	Data 5:8 (Active Actions)	Data 9:10 (Threshold Cross Event)
Get Thermal Sensor Status	10	0xDB	0x2	The value measured in "unit types"	The currently active actions as described in Table 11-39	See below

"Threshold cross events" is a bitmap that describes which events were crossed since the last read of the status or since the activation of the thermal sensor (the latest of the two). Bit 'n' in the bitmap represent threshold index 'n' as configured in the Set Thermal Sensor Configuration command (Section 11.5.11.1.9)

If the opcode is 0xFF, it means that the data that was requested from the X550 is not ready yet, or that the adequate write command was not given. The BMC should retry the read transaction or send the write transaction.

11.5.12 Example Configuration Steps

This section provides sample configuration settings for common filtering configurations. Three 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.

11.5.12.1 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 (26Fh & 298h) is filtered. For this example, the BMC must issue gratuitous ARPs because no filter is enabled to pass ARP requests to the BMC.

11.5.12.1.1 Example 1 Pseudo Code

Step 1: Disable existing filtering

```
Receive Enable[00]
```

Utilizing the simple form of the Receive Enable command, this prevents any packets from reaching the BMC by disabling filtering:

Receive Enable Control 00h:

- 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 68h). This updates MDEF[0], as indicated by the 2nd parameter (0).

MDEF[0] value of C0000000h:

- Bit 30 [1] – port 298h
- Bit 31 [1] – port 26Fh

MDEF_EXT[0] value of 0000000h:

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 298h and 26Fh would not be sent to the host.

- Bit [0] - *MDEF[0]* is exclusive to the BMC.

Step 4: - Enable Filtering

```
Receive Enable [05]
```

Using the simple form of the Receive Enable command:

Receive Enable Control 05h:

- 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 11-22. 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							
Port 0x26F	OR	X							
Flex Port 7:0	OR								
Flex TCO	OR								

11.5.12.2 Example 2 - Dedicated MAC, Auto ARP Response and RMCP Port Filtering

This example shows a common configuration; the BMC has a dedicated MAC and IP Address. Automatic ARP responses are enabled as well as RMCP port filtering. By enabling Automatic ARP responses the BMC 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 1 do 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, refer to product-specific documentation.

11.5.12.2.1 Example 2 - Pseudo Code

Step 1: Disable existing filtering

```
Receive Enable[00]
```

Utilizing the simple form of the Receive Enable command, this prevents any packets from reaching the BMC by disabling filtering:

Receive Enable Control 00h:

- Bit 0 [0] – Disable Receiving of packets

Step 2: Read System MAC Address

```
Get System MAC Address []
```

Reads the System MAC Address. Assume 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 set the Manageability Control (MANC) Register.

MANC Register 00800000h:

- Bit 23 [1] - XSUM Filter enable

Note that some of the following configuration steps manipulate the MANC register indirectly, this command sets all bits except XSUM to 0. It is important to either do this step before the others, or to read the value of the MANC and then write it back with only bit 32 changed. Also note that the XSUM enable bit may 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 68h). This updates MDEF[0], as indicated by the 2nd parameter (0).

MDEF value of 00000C00h:

- Bit 30 [1] – port 298h
- Bit 31 [1] – port 26Fh

MDEF_EXT[0] value of 0000000h:

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 61h). This updates MDEF[1], as indicated by the 2nd parameter (1).

MDEF value of 10000000:

- Bit 28 [1] – ARP Requests

MDEF_EXT[1] value of 0000000h:

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 298h and 26Fh would not be sent to the host.

- Bit [0] - MDEF[0] is exclusive to the BMC.

This allows 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, not only would RMCP traffic (ports 26Fh and 298h) be sent only to the BMC, ARP requests would have also been sent to the BMC 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 8Dh:

- 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 11-23. Example 2 MDEF Results

Filter		Manageability Decision Filter (MDEF)							
		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		X						
ARP Response	OR								
Neighbor Discovery	OR								
Port 0x298	OR	X							
Port 0x26F	OR	X							
Flex Port 7:0	OR								
Flex TCO	OR								

11.5.12.3 Example 3 - Dedicated MAC & IP Address

This example provided the BMC with a dedicated MAC and IP Address and allows it to receive ARP requests. The BMC 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 1 do 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.

For the BMC to be able to receive ARP Requests, it needs to specify a filter for this. That filter must be included in the Manageability To Host filtering so that the host OS may also receive ARP Requests.

11.5.12.3.1 Example 3 - Pseudo Code

Step 1: Disable existing filtering

```
Receive Enable[00]
```

Utilizing the simple form of the Receive Enable command, this prevents any packets from reaching the BMC by disabling filtering:

Receive Enable Control 00h:

- Bit 0 [0] – Disable Receiving of packets

Step 2: Read System MAC Address

```
Get System MAC Address []
```

Reads the System MAC Address. Assume 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 1st parameter (64h) specifies that we are configuring an IPv4 filter.

The 2nd parameter (00h) indicates which IPv4 filter is being configured, in this case filter 0.

The 3rd 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 1st parameter (66h) specifies that we are configuring a MAC Address filter.

The 2nd parameter (00h) indicates which MAC Address filter is being configured, in this case filter 0.

The 3rd 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 68h). This updates MDEF[0], as indicated by the 2nd 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 0000000h:

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 68h). This updates MDEF[1], as indicated by the 2nd parameter (1).

MDEF value of 10000000:

- Bit 28 [1] – ARP Requests

MDEF_EXT[1] value of 0000000h:

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 BMC.

Step 8: Enable Filtering

```
Receive Enable [05]
```

Using the simple form of the Receive Enable command:

Receive Enable Control 05h:

- 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 11-24. 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								

11.5.12.4 Example 4 - Dedicated MAC and VLAN Tag

This example shows an alternate configuration; the BMC has a dedicated MAC and IP Address, along with a VLAN tag of 32h is required for traffic to be sent to the BMC. This means that all traffic with VLAN a matching tag is sent to the BMC.

For demonstration purposes, the dedicated MAC Address is calculated by reading the System MAC Address and adding 1 do it, assume the System MAC is AABBCCDC. The IP Address for this example is 1.2.3.4 and the VLAN tag is 0032h.

Additionally, the XSUM filtering is enabled.

11.5.12.4.1 Example 4 - Pseudo Code

Step 1: Disable existing filtering

```
Receive Enable[00]
```

Utilizing the simple form of the Receive Enable command, this prevents any packets from reaching the BMC by disabling filtering:

Receive Enable Control 00h:

- Bit 0 [0] – Disable Receiving of packets

Step 2: - Read System MAC Address

```
Get System MAC Address []
```

Reads the System MAC Address. Assume 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 set the Manageability Control (MANC) Register.

MANC Register 00800000h:

- Bit 23 [1] – XSUM Filter enable

Note that some of the following configuration steps manipulate the MANC register indirectly, this command sets all bits except XSUM to 0. It is important to either do this step before the others, or to read the value of the MANC and then write it back with only bit 32 changed. Also note that the XSUM enable bit may 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 62h indicates update to VLAN Filter, the 2nd parameter indicates which VLAN filter (0 in this case), the last parameter is the VLAN ID (0032h).

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 68h). This updates MDEF[0], as indicated by the 2nd parameter (0).

MDEF value of 00000020:

- Bit 5 [1] – VLAN[0] AND

MDEF_EXT[0] value of 0000000h:

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 85h:

- 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.

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 11-25. 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								
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								

11.5.13 SMBus Troubleshooting

This section outlines the most common issues found while working with pass-through using the SMBus sideband interface.

11.5.13.1 TCO Alert Line Stays Asserted After a Power Cycle

After the X550 resets, all its ports indicates a status change. If the BMC only reads status from one port (slave address), the other one continues to assert the TCO alert line.

Ideally, the BMC should use the ARA transaction (see Section 11.5.10) to determine which slave asserted the TCO alert. Many customers only wish to use one port for manageability thus using ARA might not be optimal.

An alternate 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 BMC; 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 BMC 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.

11.5.13.2 When SMBus Commands are Always NACK'd

There are several reasons why all commands sent to the X550 from a BMC 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 pass-through image, as such SMBus connectivity is disabled.
- The BMC is not using the correct SMBus address — Many BMC vendors hard-code the SMBus address(es) into their firmware. If the incorrect values are hard-coded, the X550 does not respond.
  - The SMBus address(es) can be dynamically set using the SMBus ARP mechanism.
- The BMC is using the incorrect SMBus interface — The NVM might be configured to use one physical SMBus port; however, the BMC is physically connected to a different one.
- Bus Interference — The bus connecting the BMC and the X550 might be unstable.

11.5.13.3 SMBus Clock Speed is 16.6666 KHz

This can happen when the SMBus connecting the BMC and the X550 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 X550 and the BMC to this device.

11.5.13.4 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 OS and BMC.

The management control to host configuration, in this situation, was setup not to send RMCP traffic to the OS (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.

11.5.13.5 Unable to Transmit Packets from the BMC

If the BMC has been transmitting and receiving data without issue for a period of time, and then begins to receive NACKs from the X550 when it attempts to write a packet, the problem is most likely due to the fact that the buffers internal to the X550 are full of data that has been received from the network but has yet to be read by the BMC.

Being an embedded device, the X550 has limited buffers that are shared for receiving and transmitting data. If a BMC does not keep the incoming data read, the X550 can be filled up This prevents the BMC form transmitting more data, resulting in NACKs.

If this situation occurs, the recommended solution is to have the BMC issue a Receive Enable command to disable more incoming data, read all the data from the X550, and then use the Receive Enable command to enable incoming data.

11.5.13.6 SMBus Fragment Size

The SMBus specification indicates a maximum SMBus transaction size of 32 bytes. Most of the data passed between the X550 and the BMC 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 may therefore be required to move data from the X550 to the BMC or to send a data from the BMC to the X550.

Recognizing this bottleneck, the X550 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 X550 allocates buffers based upon the SMBus fragment size setting within the NVM. The X550 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, BMC implementations must take care to send data over the SMBus efficiently.

For example, the X550 firmware has 3 KB of RAM it can use for buffering SMBus fragments. If the SMBus fragment size is 32 bytes, the firmware could allocate 96 buffers of size 32 bytes each. As a result, the BMC 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 BMC 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 X550 firmware can create 12 buffers of 240 bytes each to receive SMBus fragments. The BMC 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 BMC does not also reflect this change. If a programmer changes the SMBus fragment size in the X550 to 240 bytes, and then wants to send 800 bytes of KVM data, the BMC 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 BMC is only sending fragments of size 32 bytes. This results in a memory waste of 208 bytes per fragment. Then when the BMC attempts to send more than 12 fragments in a single transaction, the X550 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 BMC implementation reflects this change and uses that fragment size to its fullest when sending SMBus fragments.

11.5.13.7 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 session to the BMC, this connection may be lost. To avoid this possible situation, the BMC can use the Management Control command detailed in Section 11.5.11.1.5 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 driver negotiates the maximum link speed, the link speed remains the same when the system powers down or resets. This may have undesirable power consumption consequences. Currently, when using NC-SI, the BMC can re-negotiate the link speed. That functionality is not available when using the SMBus interface.

11.5.13.8 Enable Checksum Filtering

If Checksum filtering is enabled, the BMC 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 BMC. All others are silently discarded.

This is a way to offload some work from the BMC.

11.5.13.9 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 X550, rather than the NVM image provided by Intel. Parts of the NVM image are changed after writing (such as the physical NVM size).

11.6 NC-SI Pass-Through Interface

The Network Controller Sideband Interface (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 supported by the X550 can be found at:

http://www.dmtf.org/sites/default/files/standards/documents/DSP0222_1.0.1.pdf

11.6.1 Overview

11.6.1.1 Terminology

The terminology in this document is taken from the NC-SI specification.

Table 11-26. 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.

Table 11-26. NC-SI Terminology [continued]

Term	Definition
Package	One or more NC-SI channels in a network controller that share a common set of electrical buffers and common buffer control for the NC-SI bus. Typically, there is 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 X550 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 X550.
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
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.

11.6.1.2 System Topology

In NC-SI each physical endpoint (NC package) can have several logical slaves (NC channels).

NC-SI defines that one management controller and up to four network controller packages can be connected to the same NC-SI link.

Figure 11-5 shows an example topology for a single BMC and a single NC package. In this example, the NC package has two NC channels.

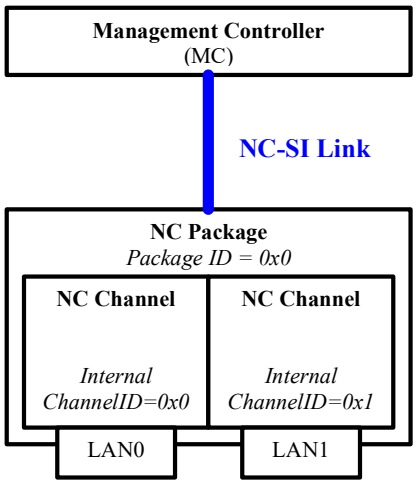


Figure 11-5. Single NC Package, Two NC Channels

Figure 11-6 shows an example topology for a single BMC 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 11-6) mandate an arbitration mechanism. The arbitration mechanism is described in Section 11.6.7.1.

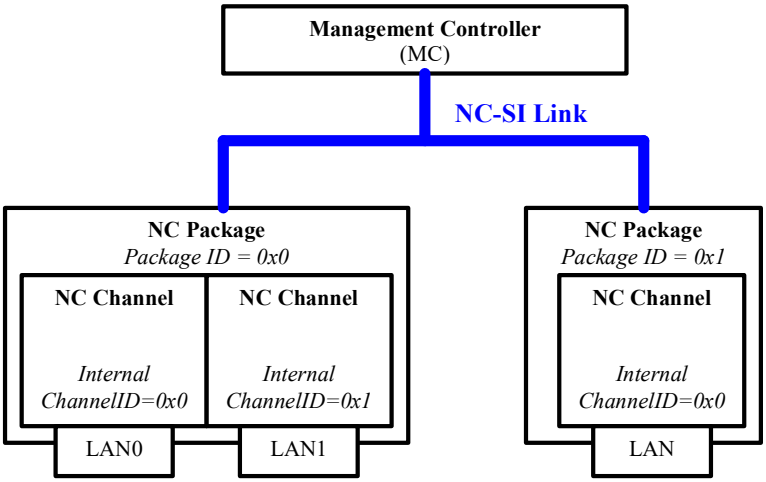


Figure 11-6. Two NC Packages (Left, with Two NC Channels and Right, with One NC Channel)

Note: Channel numbers should match PCI function numbers. If more than one function is defined on a port, the function with the lowest value associated with this port is used (it is assumed that the first function numbers are assigned to different ports). The association of functions to ports is reflected in the *PFGEN_PORTNUM* registers.

11.6.1.3 Data Transport

Since NC-SI is based upon the RMII transport layer, data is transferred in the form of Ethernet frames.

NC-SI defines two types of transmitted frames:

- Control frames:
  - Configures and control the interface
  - Identified by a unique EtherType in their L2 header
- Pass-through frames:
  - Actual LAN pass-through frames transferred from/to the BMC
  - Identified as not being a control frame
  - Attributed to a specific NC channel by their source MAC Address (as configured in the NC by the BMC)

11.6.1.3.1 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 BMC and can only be sent one at a time. Each command from the BMC is followed by a single response from the NC (command-response flow), after which the BMC 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 BMC.

AEN functionality by the NC must be disabled by default, until activated by the BMC using the Enable AEN commands.

To be considered a valid command, a control frame must:

- Comply with the NC-SI header format.
- Be targeted to a valid channel in the package via the *Package ID* and *Channel ID* fields. For example, to target a NC channel with package ID of 0x2 and internal channel ID of 0x5, the BMC must set the channel ID inside the control frame to 0x45. The channel ID is composed of three bits of package ID and five bits of internal channel ID.
- Contain a correct payload checksum (if used).
- Meet any other condition defined by NC-SI.

There are also commands (such as select package) targeted to the package as a whole. These commands must use an internal channel ID of 0x1F.

For details, refer to the NC-SI specification.

11.6.1.3.2 NC-SI Frames Receive Flow

Figure 11-7 shows the flow for frames received on the NC from the BMC.

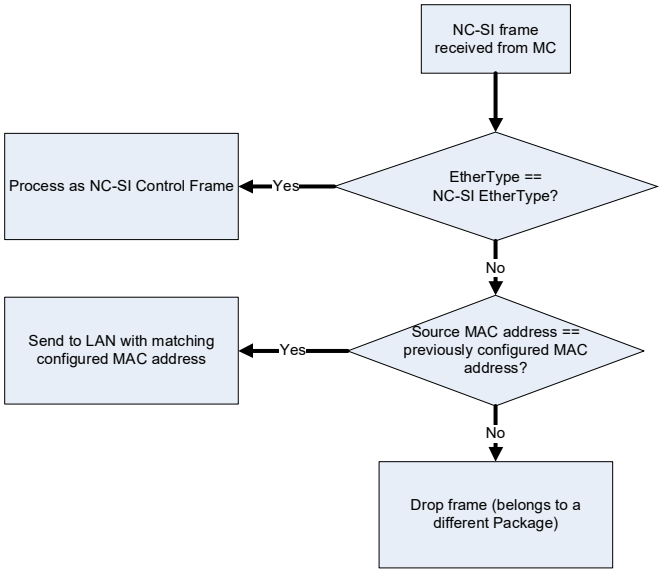


Figure 11-7. NC-SI Frames Receive Flow for the NC

11.6.2 NC-SI Standard Support

11.6.2.1 Supported Features

The X550 supports all the mandatory features of the NC-SI specification (rev 1.0.1). Table 11-27 lists the supported commands.

Table 11-28 lists optional features supported.

Table 11-27. 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[4]	Yes	Yes	No[3]
Get NC-SI Statistics	Yes, partially	Yes, partially	Yes, partially
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
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[5]	Yes	Yes	No
Disable Channel Network Tx	Yes	Yes	No
OEM Command[6]	Yes	Yes	Yes

1. In cases that one of the LAN devices is assigned for the sole use of the manageability and its LAN PCI-E 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 BMC 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 X550 does not support filtering of User priority/DEI Bits of VLAN.

3. In MCTP without pass-through mode, only control commands are supported and not pass-through traffic - thus many of the regular NC-SI commands are not supported or are supported in a limited manner, only to allow control and status reporting for the device.
4. Set MAC Address command fails with a 0x002 (Parameter Is Invalid, Unsupported, or Out-of-Range) reason code if received on a Tx enabled port with a unicast MAC Address already configured on another Tx enabled port.
5. Enable Channel Network TX command fails with a 0x002 (Parameter Is Invalid, Unsupported, or Out-of-Range) reason code if received on a port configured with an unicast MAC address equal to the MAC address of another Tx enabled port.
6. See Section 11.6.3 for details.

Table 11-28. Optional NC-SI Features Support

Feature	Implement	Details
AENs	Yes	The Driver state AEN may be emitted up to 1 minute after actual driver change if the driver was taken down unexpectedly. A "Re-configuration Required" AEN is sent before a firmware reset initiated due to a firmware code update
Get Controller Packet Statistics command	Yes, partially	Supports the following counters[1]: 2-8, 13-16[2] The statistics are cleared between reads.
Get NC-SI statistics	Yes	Supports all counters
Get NC-SI Pass-Through Statistics	Yes, partially	Support the following counters[3]: 1[4], 2, 6[5], 7
VLAN Modes	Yes, partially	Support only modes 1, 3.
Buffering Capabilities	Yes	8 Kb[6]
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 *DEI* 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*, *MANC.RCV_EN* or *GRC.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. As described in the Get NC-SI Pass-through Statistics Counters table in NC-SI specification.
4. If OS2BMC traffic is enabled, packets sent both to the net and to the host are counted twice by this counter.
5. The Total Pass-through RX Packets Received On the LAN Interface counter includes also OS2BMC traffic.
6. For OS2BMC traffic, there may be drops after 4 KB.

11.6.2.2 ALD Support

NC-SI PHY power down conditions:

In NC-SI mode, the device may 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 init time, if the manageability mode is NC-SI, PHY is required to be active only if the *Enable All PHYs in D3 N* bit in *Common Firmware Parameters* NVM word is set.
- Once a channel is enabled via Enable Channel NC-SI command, The PHY is powered up.
- If the channel is disabled via a Disable Channel command with *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 init value as define by the *Enable All PHYs in D3 N* bit.

Note: Before transitioning to D3 it is the responsibility of the driver to request the PHY to be active for wake-up activities using WUC register (setting WUC.*PROXYE* forces PHY to be active in D3).

11.6.2.3 AEN Handling

Asynchronous events may occur when the device is not allowed to send them. The following rules defines the behavior of the X550 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 package was deselected are transmitted when package is selected.
3. On a transition from Channel Disabled to Channel Enabled, all outstanding event are erased to prevent stale events notifications.

11.6.3 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 BMC to access some of the Intel-specific features present in the X550.

11.6.3.1 Overview

The following features are available via the NC-SI OEM specific commands:

- Receive filters.
- Packet Addition Decision Filters 0x0…0x4.
- Packet Reduction Decision Filters 0x5…0x7.
- *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 BMC to retrieve the system MAC Address used by the NC. This MAC Address can be used for a shared MAC Address mode.
- Keep PHY Link Up (*Veto* bit) Enable/Disable — This feature enables the BMC to block PHY reset, which might cause session loss.
- TCO Reset — Enables the BMC to reset the X550.
- Checksum offloading — Offloads IP/UDP/TCP checksum checking from the BMC.
- OS2BMC control commands.
- Firmware Version 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.

11.6.3.2 OEM Command (0x50)

The OEM command can be used by the BMC to request the sideband interface to provide vendor-specific information. The Vendor Enterprise Number (VEN) is the unique MIB/SNMP private enterprise number assigned by IANA per organization. Vendors are free to define their own internal data structures in the vendor data fields.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...	Intel Command Number	Optional Data		
...	...			
...	Optional Data		Padding to 32 bits (0x00)	
...	Checksum			

11.6.3.2.1 OEM Response (0xD0)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	Intel Command Number	Optional Return Data		
...	...			
...	Optional Return Data		Padding to 32 bits (0x00)	
...	Checksum			

Note: Responses have no command-specific reason code, unless otherwise specified within the command.

Note: The commands/responses described below includes only the part up to the Data. The padding and checksum are implied.

11.6.3.3 OEM Commands Summary

Table 11-29. 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

Table 11-30. OEM Commands Summary

Intel Command	Parameter	Command Name	Supported in MCTP without PT	Section Number
0x00	0x00	Set IP Filters Control	No	11.6.3.5
0x01	0x00	Get IP Filters Control	No	11.6.3.6
0x02	0x0F	Set Manageability Only	No	11.6.3.7.2
	0x10	Set Flexible 128 Filter Mask and Length		11.6.3.7.3
	0x11	Set Flexible 128 Filter Data		11.6.3.7.4
	0x63	Set Flex TCP/UDP Port Filters		11.6.3.7.5
	0x64	Set Flex IPv4 Address Filters		11.6.3.7.6
	0x65	Set Flex IPv6 Address Filters		11.6.3.7.7
	0x67	Set EtherType Filter		11.6.3.7.8
	0x68	Set Packet Addition Extended Filter		11.6.3.7.9
0x03	0x0F	Get Manageability Only	No	11.6.3.8.2
	0x10	Get Flexible 128 Filter Mask and Length		11.6.3.8.3
	0x11	Get Flexible 128 Filter Data		11.6.3.8.4
	0x63	Get Flex TCP/UDP Port Filters		11.6.3.8.5
	0x64	Get Flex IPv4 Address Filters		11.6.3.8.6
	0x65	Get Flex IPv6 Address Filters		11.6.3.8.7
	0x67	Get EtherType Filter		11.6.3.8.8
	0x68	Get Packet Addition Extended Filter		11.6.3.8.9
0x04	0x10	Set Extended Unicast Packet Reduction	No	11.6.3.9.1
	0x11	Set Extended Multicast Packet Reduction		11.6.3.9.2
	0x12	Set Extended Broadcast Packet Reduction		11.6.3.9.4
0x05	0x10	Get Extended Unicast Packet Reduction	No	11.6.3.10.1
	0x11	Get Extended Multicast Packet Reduction		11.6.3.10.2
	0x12	Get Extended Broadcast Packet Reduction		11.6.3.10.3
0x06	N/A	Get System MAC Address	Yes	11.6.3.11
0x20	N/A	Set Intel Management Control	No	11.6.3.12
0x21	N/A	Get Intel Management Control	No	11.6.3.13
0x22	N/A	Perform TCO Reset	Yes	11.6.3.14
0x23	N/A	Enable IP/UDP/TCP Checksum Offloading	No	11.6.3.15.1
0x24	N/A	Disable IP/UDP/TCP Checksum Offloading	No	11.6.3.15.3
0x40	0x01	Enable OS2BMC Flow	No	11.6.3.16.1
	0x02	Enable Network to BMC Flow		11.6.3.16.2
	0x03	Enable Both Network to BMC and Host to BMC Flow		11.6.3.16.3
	0x04	Set BMC IP Address	Yes	11.6.3.16.4
0x41	N/A	Get OS2BMC Parameters	No	11.6.3.16.5
0x48	0x1	Get Controller Information	Yes	11.6.3.17.1

Table 11-30. OEM Commands Summary [continued]

Intel Command	Parameter	Command Name	Supported in MCTP without PT	Section Number
0x4C	0x0	Get Thermal Sensor Capabilities	Yes	11.6.3.18.2
	0x1	Get Thermal Sensor Configuration		11.6.3.18.3
	0x2	Get Thermal Sensor Status		11.6.3.18.4
0x4D	0x1	Set Thermal Sensor Configuration	Yes	11.6.3.19.1
	0x2	Set Thermal Action		11.6.3.19.2

Note: All the commands are supported both over RMII NC-SI and over MCTP.

11.6.3.4 Proprietary Commands Format

11.6.3.4.1 Set Intel Filters Control Command (Intel Command 0x00)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x00	Filter Control Index		

11.6.3.4.2 Set Intel Filters Control Response Format (Intel Command 0x00)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x00	Filter Control Index		

11.6.3.5 Set Intel Filters Control — IP Filters Control Command (Intel Command 0x00, Filter Control Index 0x00)

This command controls different aspects of the Intel filters.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x00	0x00	IP Filters control (3-2)	
24...27	IP Filters Control (1-0)			

Where "IP Filters Control" has the following format.

Bit(s)	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
31:1	Reserved		

11.6.3.5.1 Set Intel Filters Control — IP Filters Control Response (Intel Command 0x00, Filter Control Index 0x00)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x00	0x00		

11.6.3.6 Get Intel Filters Control Commands (Intel Command 0x01)

11.6.3.6.1 Get Intel Filters Control — IP Filters Control Command (Intel Command 0x01, Filter Control Index 0x00)

This command controls different aspects of the Intel filters.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x01	0x00		

11.6.3.6.1.1 Get Intel Filters Control — IP Filters Control Response (Intel Command 0x01, Filter Control Index 0x00)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x01	0x00	IP Filters Control (3-2)	
28...29	IP Filters Control (1-0)			

11.6.3.7 Set Intel Filters Formats

11.6.3.7.1 Set Intel Filters Command (Intel Command 0x02)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x02	Parameter Number	Filters Data (optional)	

11.6.3.7.1.1 Set Intel Filters Response (Intel Command 0x02)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...	0x02	Filter Control Index	Return Data (Optional)	

11.6.3.7.2 Set Intel Filters — Manageability Only Command (Intel Command 0x02, Filter Parameter 0x0F)

This command sets the MNGONLY register. The MNGONLY register controls whether pass-through packets destined to the BMC are not forwarded to the Host OS. The MNGONLY register is described in Table 11-4.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x0F	Manageability Only (3-2)	
24...25	Manageability Only (1-0)			

11.6.3.7.2.1 Set Intel Filters — Manageability Only Response (Intel Command 0x02, Filter Parameter 0x0F)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x0F		

11.6.3.7.3 Set Intel Filters — Flex Filter Enable Mask and Length Command (Intel Command 0x02, Filter Parameter 0x10)

The following command sets the Intel flex filters mask and length.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x10	Mask Byte 1	Mask Byte 2
24...27	...	...	...	...
28...31	...	...	...	...
32...35	...	...	...	...
36...37	Mask Byte 15	Mask Byte 16	Reserved	Reserved
38	Length			

11.6.3.7.3.1 Set Intel Filters — Flex Filter Enable Mask and Length Response (Intel Command 0x02, Filter Parameter 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x10		

11.6.3.7.4 Set Intel Filters — Flex Filter Data Command (Intel Command 0x02, Filter Parameter 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...	0x02	0x11	Filter Data Group	Filter Data 1
	...	Filter Data N		

The Filter Data Group parameter defines which bytes of the Flex filter are set by this command:

Table 11-31. Filter Data Group

Code	Bytes Programmed	Filter Data Length
0x0	Bytes 0–29	1–30
0x1	Bytes 30–59	1–30
0x2	Bytes 60–89	1–30
0x3	Bytes 90–119	1–30
0x4	Bytes 120–127	1–8

Note: Using this command to configure the filters data must be done after the flex filter mask command is issued and the mask is set.

11.6.3.7.4.1 Set Intel Filters — Flex Filter Data Response (Intel Command 0x02, Filter Parameter 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x11		

11.6.3.7.5 Set Intel Filters — Flex TCP/UDP Port Filter Command (Intel Command 0x02, Filter Parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x63	Port filter index	TCP/UDP Port MSB
24	TCP/UDP Port LSB			

Filter index range: 0x0...0xA.

If the filter index is bigger than 10, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

11.6.3.7.5.1 Set Intel Filters — Flex TCP/UDP Port Filter Response (Intel Command 0x02, Filter Parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x63		

11.6.3.7.6 Set Intel Filters — IPv4 Filter Command (Intel Command 0x02, Filter Parameter 0x64)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x64	IP filter index	IPv4 Address (3)
24...26	IPv4 Address (2-0)			

Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command.

IPv4 Mode: Filter index range: 0x0...0x3.

IPv6 Mode: This command should not be used in IPv6 mode.

11.6.3.7.6.1 Set Intel Filters — IPv4 Filter Response (Intel Command 0x02, Filter Parameter 0x64)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x64		

If the IPv4 address equals all zero, a command failed Response Code is returned, with Invalid Intel Parameter Number reason (0x5082).

11.6.3.7.7 Set Intel Filters — IPv6 Filter Command (Intel Command 0x02, Filter Parameter 0x65)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	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.

11.6.3.7.7.1 Set Intel Filters — IPv6 Filter Response (Intel Command 0x02, Filter Parameter 0x65)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x65		

If the IP filter index does not match the ranges above, a command failed Response Code is returned, with Invalid Intel Parameter Number reason (0x5082).

If the IPv6 address equals all zero, a command failed Response Code is returned, with Invalid Intel Parameter Number reason (0x5082).

11.6.3.7.8 Set Intel Filters - EtherType Filter Command (Intel Command 0x02, Filter Parameter 0x67)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x67	EtherType Filter Index	EtherType Filter MSB
24...27	...	...	EtherType Filter LSB	

Where the EtherType Filter has the format as described in Section 8.2.2.19.6.

11.6.3.7.8.1 Set Intel Filters - EtherType Filter Response (Intel Command 0x02, Filter Parameter 0x67)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x67		

If the EtherType filter Index is larger than 3, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

11.6.3.7.9 Set Intel Filters - Packet Addition Extended Decision Filter Command (Intel Command 0x02, Filter Parameter 0x68)

See Figure 11-2 for description of the decision filters structure.

This command overwrites any previously stored value. The value set is not checked.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x68	Extended Decision filter Index	Extended Decision filter 1 MSB
24...27	...	...	Extended Decision filter 1 LSB	Extended Decision filter 0 MSB
28...30	...	...	Extended Decision filter 0 LSB	

Extended Decision filter Index Range: 0...4

Filter 0: See Table 11-32.

Filter 1: See Table 11-33.

Table 11-32. Filter Values

Bit(s)	Name	Description
1:0	Unicast (AND)	If set, packets must match unicast filter 0 to 1, respectively.
3:2	Reserved	Reserved.
4	Broadcast (AND)	If set, packets must match the broadcast filter.
12:5	VLAN (AND)	If set, packets must match VLAN filter 0 to 7, respectively.
16:13	IPv4 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively
20:17	IPv6 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively
22:21	Unicast (OR)	If set, packets can pass if match unicast filter 0 to 1, respectively, or a different OR filter.
24:23	Reserved	Reserved.
25	Broadcast (OR)	If set, packets can pass if match the broadcast filter or a different OR filter.
26	Multicast (AND)	If set, packets must match the multicast filter.
27	ARP Request (OR)	If set, packets can pass if match the ARP request filter or a different OR filter.
28	ARP Response (OR)	If set, packets can pass if match the ARP response filter or a different OR filter.
29	Neighbor Discovery - 134 (OR)	If set, packets can pass if match the neighbor discovery filter (type134 - router advertisement) or a different OR filter.
30	Port 0x298 (OR)	If set, packets can pass if match a fixed TCP/UDP port 0x298 filter or a different OR filter.
31	Port 0x26F (OR)	If set, packets can pass if match a fixed TCP/UDP port 0x26F filter or a different OR filter.

Table 11-33. Extended Filter 1 Values

Bit(s)	Name	Description
3:0	EtherType 0 -3 (AND)	If set, packets must match the EtherType filter 0 to 3, respectively.
7:4	EtherType 0 -3 (OR)	If set, packets must match the EtherType filter 0 to 3, respectively, or a different OR filter.
18:8	Flex port 10:0 (OR)	If set, packets can pass if match the TCP/UDP Port filter 10:0.
19	DHCPv6 (OR)	If set, packets can pass if match the DHCPv6 port (0x0223).
20	DHCP Client (OR)	If set, packets can pass if match the DHCP Server port (0x0043).
21	DHCP Server (OR)	If set, packets can pass if match the DHCP Client port (0x0044).
22	NetBIOS Name Service (OR)	If set, packets can pass if match the NetBIOS Name Service port (0x0089).
23	NetBIOS Datagram Service (OR)	If set, packets can pass if match the NetBIOS Datagram Service port (0x008A).
24	Flex TCO (OR)	If set, packets can pass if match the Flex 128 TCO filter.
25	Neighbor Discovery - 135 (OR)	If set, packets must also match the neighbor discovery filter (type135 - Neighbor Solicitation. or a different OR filter.
26	Neighbor Discovery - 136 (OR)	If set, packets must also match the neighbor discovery filter (type136 - Neighbor Advertisement) or a different OR filter.
27	Neighbor Discovery - 137 (OR)	If set, packets must also match the neighbor discovery filter (type137 - Redirect) or a different OR filter.
28	Reserved	Reserved.
29	MLD (OR)	If set, packets must also match one of the MLD ICMPv6 types or a different OR filter.
31:30	Reserved	Reserved.

11.6.3.7.9.1 Set Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x02, Filter Parameter 0x68)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x68		

If the Extended Decision filter Index is bigger than 5, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

11.6.3.8 Get Intel Filters Formats

11.6.3.8.1 Get Intel Filters Command (Intel Command 0x03)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	Parameter Number		

11.6.3.8.1.1 Get Intel Filters Response (Intel Command 0x03)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x03	Parameter Number	Optional Return Data	

11.6.3.8.2 Get Intel Filters — Manageability Only Command (Intel Command 0x03, Filter Parameter 0x0F)

This command retrieves the MNGONLY register. The MNGONLY register controls whether pass-through packets destined to the BMC are also be forwarded to the Host OS.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	0x0F		

11.6.3.8.2.1 Get Intel Filters — Manageability Only Response (Intel Command 0x03, Filter Parameter 0x0F)

The *MNGONLY* register structure is described in Table 11-4.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x0F	Manageability to Host (3-2)	
28...29	Manageability to Host (1-0)			

11.6.3.8.3 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 11.3.3.5 for details of the values returned by this command.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	0x10		

11.6.3.8.3.1 Get Intel Filters — Flex Filter 0 Enable Mask and Length Response (Intel Command 0x03, Filter Parameter 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x10	Mask Byte 1	Mask Byte 2
28...31	...	...	...	...
32...35	...	...	...	...
36...39	...	...	...	...
40...43	...	Mask Byte 16	Reserved	Reserved
44	Flexible Filter Length			

11.6.3.8.4 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**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
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 11-34. 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

11.6.3.8.4.1 Get Intel Filters — Flex Filter 0 Data Response (Intel Command 0x03, Filter Parameter 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...	0x03	0x11	Filter Group Number	Filter Data 1
	...	Filter Data N		

11.6.3.8.5 Get Intel Filters — Flex TCP/UDP Port Filter Command (Intel Command 0x03, Filter Parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x63	TCP/UDP Filter Index	

Filter index range: 0x0...0xA.

11.6.3.8.5.1 Get Intel Filters — Flex TCP/UDP Port Filter Response (Intel Command 0x03, Filter Parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x63	TCP/UDP Filter Index	TCP/UDP Port (1)
28	TCP/UDP Port (0)			

Filter index range: 0x0...0xA.

11.6.3.8.6 Get Intel Filters — IPv4 Filter Command (Intel Command 0x03, Filter Parameter 0x64)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x64	IPv4 Filter Index	

Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command.

IPv4 Mode: Filter index range: 0x0...0x3.

IPv6 Mode: This command should not be used in IPv6 mode.

11.6.3.8.6.1 Get Intel Filters — IPv4 Filter Response (Intel Command 0x03, Filter Parameter 0x64)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x64	IPv4 Filter Index	IPv4 Address (3)
28...29	IPv4 Address (2-0)			

11.6.3.8.7 Get Intel Filters — IPv6 Filter Command (Intel Command 0x03, Filter Parameter 0x65)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x65	IPv6 Filter Index	

Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command

IPv4 Mode: Filter index range: 0x0...0x2.

IPv6 Mode: Filter index range: 0x0...0x3.

11.6.3.8.7.1 Get Intel Filters — IPv6 Filter Response Intel Command 0x03, Filter Parameter 0x65)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x65	IPv6 Filter Index	IPv6 Address (MSB, Byte 16)
28...31	...	...	...	...
32...35	...	...	...	...
36...39	...	...	...	...
40...42	...	...	IPv6 Address (LSB, Byte 0)	

11.6.3.8.8 Get Intel Filters - EtherType Filter Command (Intel Command 0x03, Filter Parameter 0x67)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x67	EtherType Filter Index	

Valid indices: 0...3

11.6.3.8.8.1 Get Intel Filters - EtherType Filter Response (Intel Command 0x03, Filter Parameter 0x67)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x67	EtherType Filter Index	EtherType Filter MSB
28...30	...	...	EtherType Filter LSB	

If the EtherType filter Index is larger than 3, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

11.6.3.8.9 Get Intel Filters – Packet Addition Extended Decision Filter Command (Intel Command 0x03, Filter Parameter 0x68)

This command allows the BMC to retrieve the Extended Decision Filter.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x68	Extended Decision Filter Index	

11.6.3.8.9.1 Get Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x03, Filter Parameter 0x68)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x68	Decision Filter Index	Decision Filter 1 MSB
28...31	...	...	Decision Filter 1 LSB	Decision Filter 0 MSB
32...34	...	...	Decision Filter 0 LSB	

Where Decision Filter 0 & Decision Filter 1 have the structure as detailed in the respective “Set” commands.

If the Extended Decision Filter Index is bigger than 4, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

11.6.3.9 Set Intel Packet Reduction Filters Formats

The non extended commands (are obsolete. The extended commands (Section 11.6.3.9.2 to Section 11.6.3.9.4.1) should be used instead

11.6.3.9.1 Set Intel Packet Reduction Filters Command (Intel Command 0x04)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x04	Packet Reduction Index	Packet Reduction Data	

Note: It is advised that the BMC only uses the Extended Packet Reduction commands.

The *Packet Reduction Data* field has the following structure:

Table 11-35. Packet Reduction Field Description

Bit(s)	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 11-36. Extended Packet Reduction Field Description

Bit(s)	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	
24	Flex TCO (OR)	If set, packets can pass if match the Flex 128 TCO filter.
28:25	Reserved	

Table 11-36. Extended Packet Reduction Field Description [continued]

Bit(s)	Name	Description
29	MLD (OR)	If set, packets must also match one of the MLD ICMPv6 types or a different OR filter.
31:30	Reserved	Reserved.

The filtering is divided into two decisions:

- Bit 20:13 in Table 11-35 and Bits 3:2 in Table 11-36 works in an AND manner; it must be true for a packet to pass (if was set).
- Bits 28 in Table 11-35 and Bits 24:10 in Table 11-36 work in an OR manner; at least one of them must be true for a packet to pass (if any were set).

11.6.3.9.1.1 Set Intel Packet Reduction Filters Response (Intel Command 0x04)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...	0x04	Packet Reduction Index		

11.6.3.9.2 Set Extended Unicast Packet Reduction Command (Intel Command 0x04, Reduction Filter Index 0x10)

This command has the following format:

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x04	0x10	Extended Unicast Reduction Filter MSB	...
24...27	...	Extended Unicast Reduction Filter LSB	Unicast Reduction Filter MSB	...
28...29	...	Unicast Reduction Filter LSB		

This command overwrites any previously stored value.

Note: See Table 11-35 and Table 11-36 for description of the Unicast Extended Packet Reduction format.

11.6.3.9.2.1 Set Extended Unicast Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x04	0x10		

11.6.3.9.3 Set Extended Multicast Packet Reduction Command (Intel Command 0x04, Reduction Filter Index 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	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 11-35 and Table 11-36 for description of the Multicast Extended Packet Reduction format.

This command overwrites any previously stored value.

11.6.3.9.3.1 Set Extended Multicast Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x04	0x11		

11.6.3.9.4 Set Extended Broadcast Packet Reduction Command (Intel Command 0x04, Reduction Filter Index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	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 11-35 and Table 11-36 for description of the Broadcast Extended Packet Reduction format.

This command overwrites any previously stored value.

11.6.3.9.4.1 Set Extended Broadcast Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x04	0x12		

11.6.3.10 Get Intel Packet Reduction Filters Formats

Note: The non extended commands are not supported anymore. Use the extended commands (Section 11.6.3.10.1 to Section 11.6.3.10.3.1) instead

11.6.3.10.1 Get Extended Unicast Packet Reduction Command (Intel Command 0x05, Reduction Filter Index 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x10		

11.6.3.10.1.1 Get Extended Unicast Packet Reduction Response (Intel Command 0x05, Reduction Filter Index 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	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)			

11.6.3.10.2 Get Extended Multicast Packet Reduction Command (Intel Command 0x05, Reduction Filter Index 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x11		

11.6.3.10.2.1 Get Extended Multicast Packet Reduction Response (Intel Command 0x05, Reduction Filter Index 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x11	Extended Multicast Packet Reduction (3-2)	
28...29	Extended Multicast Packet Reduction (1-0)		Multicast Packet Reduction (3-2)	
30...31	Multicast Packet Reduction (1-0)			

11.6.3.10.3 Get Extended Broadcast Packet Reduction Command (Intel Command 0x05, Reduction Filter Index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x12		

11.6.3.10.3.1 Get Extended Broadcast Packet Reduction Response (Intel Command 0x05, Reduction Filter Index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x12	Extended Broadcast Packet Reduction (3-2)	
28...29	Extended Broadcast Packet Reduction (1-0)		Broadcast Packet Reduction (3-2)	
30...31	Broadcast Packet Reduction (1-0)			

11.6.3.11 System MAC Address

11.6.3.11.1 Get System MAC Address Command (Intel Command 0x06)

To support a system configuration that requires the NC to hold the MAC Address for the BMC (such as shared MAC Address mode), the following command is provided to enable the BMC to query the NC for a valid MAC Address.

The NC must return the system MAC Addresses. The BMC 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.

It is also recommended that the BMC use packet reduction and Manageability-to-Host command to set the proper filtering method.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20	0x06			

11.6.3.11.2 Get System MAC Address Response (Intel Command 0x06)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x06	MAC Address		
28...30	MAC Address			

11.6.3.12 Set Intel Management Control Formats

11.6.3.12.1 Set Intel Management Control Command (Intel Command 0x20)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x20	0x00	Intel Management Control 1	

Where Intel Management Control 1 is as follows:

Bit(s)	Default	Description
0	0b	Enable Critical Session Mode (Keep PHY Link Up and Veto Bit) 0b = Disabled 1b = Enabled When critical session mode is enabled, the following behaviors are disabled: • The PHY is not reset on PE_RST# and PCIe resets (in-band and link drop). Other reset events are not affected — Internal_Power_On_Reset, device disable, Force TCO, and PHY reset by software. • 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.

11.6.3.12.2 Set Intel Management Control Response (Intel Command 0x20)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x20	0x00		

11.6.3.13 Get Intel Management Control Formats

11.6.3.13.1 Get Intel Management Control Command (Intel Command 0x21)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x21	0x00		

Where Intel Management Control 1 is as described in Section 11.6.3.12.2.

11.6.3.13.2 Get Intel Management Control Response (Intel Command 0x21)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...26	0x21	0x00	Intel Management Control 1	

11.6.3.14 TCO Reset

Depending on the bit set in the TCO mode field this command causes the X550 to perform either:

1. **TCO Reset**
   - The Force TCO reset clears the data-path (Rx/Tx) of the X550 to enable the BMC to transmit/ receive packets through the X550.
   - If the BMC has detected that the OS is hung and has blocked the Rx/Tx path The Force TCO reset clears the data-path (Rx/Tx) of the Network Controller to enable the BMC to transmit/ receive packets through the Network Controller.
   - When this command is issued to a channel in a package, it applies only to the specific channel.
   - After successfully performing the command, the Network Controller considers Force TCO command as an indication that the OS is hung, and clears the DRV_LOAD flag (disable the driver). If TCO reset is disabled in NVM, the X550 does not reset the data-path and notifies BMC on successful completion.
2. **TCO isolate**
   - If TCO isolate is enabled in the NVM (See Section 6.2.16). The TCO Isolate command disables PCIe write operations to the LAN port.
   - If TCO Isolate is disabled in NVM, the X550 does not execute the command but sends a response to the BMC with successful completion.
   - Following TCO Isolate management isolates the function related to the port on which the command was received.
3. **Firmware Reset**
   - This command causes re-initialization of all the manageability functions and re-load of manageability related NVM words.
   - When the BMC has loaded new management-related NVM image, the Firmware Reset command loads 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: TCO isolate and force TCO affects only the channel (port) that the command was issued to.

Following firmware reset, BMC must re-initialize all ports. Only one of the fields should be set in a given command. Setting more than one field may yield unexpected results.

11.6.3.14.1 Perform Intel TCO Reset Command (Intel Command 0x22)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20	0x22	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).

Note: For compatibility, the TCO reset command without the TCO Mode parameter is accepted (TCO reset is performed).

1. TCO Isolate Host Write operation enabled in NVM.

11.6.3.14.2 Perform Intel TCO Reset Response (Intel Command 0x22)

When a firmware reset is requested (TCO Mode = RESET_MGMT), there is no response, as the NC goes to Initial State as part of the command execution.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...26	0x22			

11.6.3.15 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 BMC.

11.6.3.15.1 Enable Checksum Offloading Command (Intel Command 0x23)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20	0x23			

11.6.3.15.2 Enable Checksum Offloading Response (Intel Command 0x23)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...26	0x23			

11.6.3.15.3 Disable Checksum Offloading Command (Intel Command 0x24)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20	0x24			

11.6.3.15.4 Disable Checksum Offloading Response (Intel Command 0x24)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...26	0x24			

11.6.3.16 OS2BMC Configuration

These commands control enabling of the OS 2 BMC flow.

11.6.3.16.1 Enable OS2BMC Flow Command (Intel Command 0x40, Index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x01		

11.6.3.16.1.1 EnableOS2BMC Flow Response (Intel Command 0x40, Index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x01		

11.6.3.16.2 Enable Network to BMC Flow Command (Intel Command 0x40, Index 0x2)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x02		

11.6.3.16.2.1 Enable Network to BMC Flow Response (Intel Command 0x40, Index 0x2)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x02		

11.6.3.16.3 Enable Both Host and Network to BMC Flows Command (Intel Command 0x40, Index 0x3)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x03		

11.6.3.16.3.1 Enable Both Host and Network to BMC Flows Response (Intel Command 0x40, Index 0x3)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x03		

11.6.3.16.4 Set BMC IP Address Command (Intel Command 0x40, Index 0x4)

This command is used to expose the BMC IP Address to the host.

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**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x40	0x04	IP type	IPv6 Address (MSB, byte 15)/IPv4 Address (MSB, byte 3)
24...27	IPv6 Address (byte 14)/ IPv4 Address (byte 2)	IPv6 Address (byte 13)/ IPv4 Address (byte 1)	IPv6 Address (byte 12)/ IPv4 Address (LSB, byte 0)	IPv6 Address (byte 11)/ Reserved
28...31	...	...	...	...
32...35	...	...	...	...
36...38	...	...	IPv6 Address (LSB, byte 0)/Reserved	

11.6.3.16.4.1 Set BMC IP Address Response (Intel Command 0x40, Index 0x4)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x04		

11.6.3.16.5 Get OS2BMC Parameters Command (Intel Command 0x41)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20	0x41			

11.6.3.16.5.1 Get OS2BMC Parameters Response (Intel Command 0x41)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x41	Status	IPv6 Address (MSB, byte 15)/IPv4 Address (MSB, byte 3)	IPv6 Address (byte 14)/ IPv4 Address (byte 2)
28...31	IPv6 Address (byte 13)/ IPv4 Address (byte 1)	IPv6 Address (byte 12)/ IPv4 Address (LSB, byte 0)	IPv6 Address (byte 11)/ Reserved	...
32...35	...	...	...	...
36...39	...	...	...	...
40...41	...	IPv6 Address (LSB, byte 0)/Reserved		

Where the Status byte partition is as follows:

Table 11-37. Status Byte Description

Bit(s)	Content
0	Relevant only if the IP Address valid bit is set. 0b = IPv4 1b = IPv6
1	IP Address valid.
2	Network to BMC status. 0b = Network 2 BMC flow is disabled. 1b = Network 2 BMC flow is enabled.

Table 11-37. Status Byte Description [continued]

Bit(s)	Content
3	OS2BMC status. 0b = OS 2 BMC flow is disabled. 1b = OS 2 BMC flow is enabled.
7:4	Reserved.

11.6.3.17 Get Controller Information Command (Intel Command 0x48, Index 0x1)

This command gather the controller identification information and return it back to the BMC.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x48	0x1		

11.6.3.17.1 Get Controller Information Response (Intel Command 0x48, Index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x48	0x01	Reserved	Number of Inventory entries
28...31	Controller Info Item 1 ID	Controller Info Item 1 length	Controller Info Item 1 Data	
...	....			
...	Controller Info Item 2 ID	Controller Info Item 2 length	Controller Info Item 2 Data	
...	....			
...	Controller Info Item n ID	Controller Info Item n length	Controller Info Item n Data	
...	....			

Where the possible inventory items are as described below. Note that not all the inventory items would be present in all the implementations of this command.

Table 11-38. Controller Information Items

ID	Length (bytes)	Data	Notes
0x00	3	Device ID (2 bytes) + RevID	This is the hardware default value, not any value programmed via NVM. Not returned if command set while device is in Dr.
0x0B	2	NVM Image Version	
0x0C	4	EMP ROM Internal Version	
0x0D	4	Flash Firmware Image Internal version	
0x0E	2	PXE firmware version	MajorVersion.MinorVersion.Build
0x0F	2	iSCSI firmware version	
0x10	2	uEFI firmware version	
0x16	2	FCoE Boot firmware version	
0x17	4	Firmware Mini Loader Internal version	

11.6.3.18 Get Thermal Sensor Commands (Intel Command 0x4C)

The following tables are used in the various commands:

Table 11-39. Actions Word Definition

Bit(s)	Action	Notes
0	Measure only	Activate the thermal sensor, but no automatic action.
1	Notify BMC	
2	Power off PHY	
3	Power on PHY	
4	Restore Speed	When the Cancel TS Action command is received, the X550 should set the PHY to its previous state.
5	Set speed to 10 Mb/s Max	N/A for the X550.
6	Set speed to 100 Mb/s Max	These actions set a maximum on the speed and do not force a specific speed. The Set Link command should be used to set a specific link speed.
7	Set speed to 1 Gb/s Max	
8	Set speed to 10 Gb/s Max	
9	Indicate on SDP (set)	The SDP indication is done on SDP_1_1. Indicate on SDP actions are available only if SDP_1_1 is configured as an output in the ESDP register.
10	Indicate on SDP (clear)	
11	HW autonomous algorithm	This is used for the X550 and is equivalent to the NVM based mechanism described in Section 5.7.3. When the Threshold set by the MC is crossed, both PHYs are powered down. This action can be canceled by a Cancel all active actions immediate action
12	Cancel all active actions	
13	Rearm Event	When the Cancel TS Action command is received, the X550 should set the PHY to its previous state.
31:14	Reserved	

Table 11-40. Unit Types Word Definition

Value	Unit Types	Notes
0x0	Generic Number	No specific indication of measured value.
0x1	Celsius	Temperature.
0x2	Volts	
0x3	rpm	Speed.
0x4-0xFF	Reserved	Reserved.

11.6.3.18.1 Threshold and Hysteresis

Each threshold event includes a direction. For example, a thermal event with a threshold of 100 °C and an “up” direction is defined as the temperature crossing from less than 100 °C to more than 100 °C.

For each threshold an hysteresis may be defined. This hysteresis direction is opposite to the threshold direction. If in the previous example, an hysteresis of 10 °C is defined, it is activated when the temperature crosses from more than 90 °C to less than 90 °C.

Figure 11-8 describes the thresholds and hysteresis modes and the actions activated for each of them.

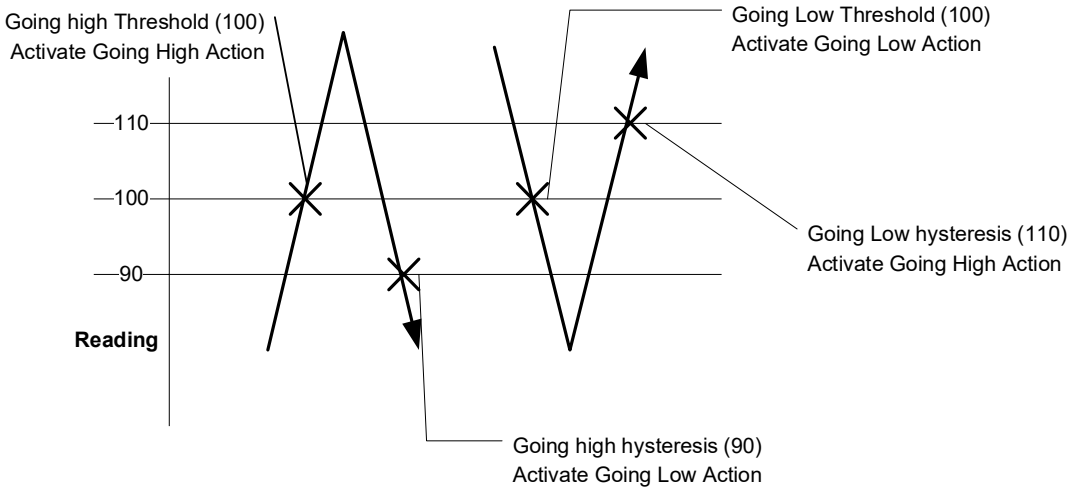


Figure 11-8. Thresholds, Hysteresis and Actions

Note: As there are no low thresholds in the X550, there is no support for thresholds.

11.6.3.18.2 Get Thermal Sensor Capabilities Command (Intel Command 0x4C, Index 0x0)

This command requests the thermal sensor capabilities supported by this device.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x4C	0x00		

11.6.3.18.2.1 Get Thermal Sensor Capabilities Response (Intel Command 0x4C, Index 0x0)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x4C	0x0	Version	Unit Types
28...31	Number of Thresholds		Accuracy	Max hysteresis
32...35	M	B	K1	K2
36...39	Available actions - "High going" thresholds			
40...43	Available actions - "Low going" thresholds			

- Version should always be 1.
- Unit Types = 1 - Temperature reported in Celsius.
- Accuracy describes the accuracy of the reported measurements is as follows:
  - 7:4: Max deviation of actual value above measurement in "unit types" = 3 °C
  - 3:0: Max deviation of actual value below measurement in "unit types" = 3 °C
- Number of thresholds describes the number of up and down thresholds as follows:
  - 15:12: Reserved
  - 11:8: Max Number of mixed thresholds = 0
  - 7:4: Max number of up thresholds = 1
  - 3:0: Max number of down thresholds = 0
- Available Actions "High going" thresholds — Describes the actions that can be activated by the device as described in Table 11-39 when a high going threshold is crossed. The X550 supports Measure Only, Notify BMC, Indicate on SDP (set), Set speed to 100 Mb/s Max, Set speed to 1 GbE Max, and HW autonomous actions.
- Available Actions "Low going" thresholds — Describes the actions that can be activated by the device as described in Table 11-39 when a low going threshold is crossed. The X550 does not support low going thresholds.
- Available Actions "Immediate" — Describes the actions that can be activated by the device as described in Table 11-39. The X550 supports Reset Thermal Sensor, Cancel all active actions, and measure-only actions.
- M = 1; B = 0, K1 = K2 = 0 — The assumption is that the thermal sensor in the X550 is the readable value.
- $T_{junction-Max}$ — The maximal junction temperature supported in X550 is 105 °C.

Note: This formula is compliant with the definition of section 36.3 "Sensor Reading Conversion Formula" in IPMI 2.0

11.6.3.18.3 Get Thermal Sensor Configuration Command (Intel Command 0x4C, Index 0x1)

This command requests the thermal sensor configuration for threshold "Index" in direction "Direction".

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x4C	0x01	Index	

Note: If Index points to a non valid threshold as described in the Get Thermal Sensor Capabilities Response, the command fails with an Invalid Parameter reason.

11.6.3.18.3.1 Get Thermal Sensor Configuration Response (Intel Command 0x4C, Index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x4C	0x1	Index	Threshold (1)
28...31	Threshold(0)	Actions "Going High" (3-1)		
32...35	Actions "Going High" (0)	Actions "Going Low" (3-1)		
36...38	Actions "Going Low" (0)	Direction	Hysteresis	

The threshold and the Hysteresis are measured in "unit types".

The *Actions Going High* field describes the actions to activate upon crossing of the threshold for "Going High" thresholds or when crossing the hysteresis for "Going Low" thresholds according to Table 11-39.

The *Actions Going Low* field describes the actions to activate upon crossing of the threshold for "Going Low" thresholds or when crossing the hysteresis for "Going High" thresholds according to Table 11-39.

Direction is encoded as follows:

0 = High Going
1 = Low Going

11.6.3.18.4 Get Thermal Sensor Status Command (Intel Command 0x4C, Index 0x2)

This command requests the current status of the Thermal sensor.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x4C	0x02		

11.6.3.18.4.1 Get Thermal Sensor Status Response (Intel Command 0x4C, Index 0x2)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x4C	0x2	Measured Value	
28...31	Active Actions			
32...33	Threshold cross events			

Where “Threshold cross events” is a bitmap that describes which events were crossed since the last read of the status or since the activation of the thermal sensor (the latest of the two).

Bit ‘n’ in the bitmap represent threshold index ‘n’ as configured in the Set Thermal Sensor Configuration command (Section 11.6.3.19.1).

Active actions can be set for the following actions (if supported by the device):

- Measure only
- Notify BMC
- Power off PHY

- Set speed to 10 Mb/s Max
- Set speed to 100 Mb/s Max
- Set speed to 1 Gb/s Max
- Set speed to 10 Gb/s Max
- Indicate on SDP (set)
- Indicate on SDP (clear)
- HW autonomous algorithm.

11.6.3.19 Set Thermal Sensor Commands (Intel Command 0x4D)

11.6.3.19.1 Set Thermal Sensor Configuration Command (Intel Command 0x4D, Index 0x1)

This command sets the thermal sensor configuration for threshold "Index"

The threshold and the Hysteresis are measured in "unit types".

Note: If the Max hysteresis reported in the Thermal Sensor capabilities is zero, the Hysteresis value is ignored.

The *Actions Going High* field describes the actions to activate upon crossing of the threshold for "Going High" thresholds or when crossing the hysteresis for "Going Low" thresholds according to Table 11-39.

The *Actions Going Low* field describes the actions to activate upon crossing of the threshold for "Going Low" thresholds or when crossing the hysteresis for "Going High" thresholds according to Table 11-39.

Direction is encoded as follows:

0 = High Going
1 = Low Going

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x4D	0x01	Index	Direction
24...27	Threshold		Actions "Going High" (MSB)	
28...31	Actions "Going High" (LSB)		Actions "Going Low" (MSB)	
32...34	Actions "Going Low" (LSB)		Hysteresis	

Note: If Index and Direction points to a non valid threshold as described in the Get Thermal Sensor Capabilities Response, the command fails with an Invalid Parameter reason.

Note: If the requested action is not supported as defined in Get Thermal Sensor Capabilities Command (Section 11.6.3.18.2), the command fails with an Invalid Parameter reason.

Note: Actions set can not be contradictory - so for a given set of actions, the following combinations are invalid, and result in a command fails with an Invalid Parameter reason:

- Indicate on SDP (set) and Indicate on SDP (clear) both set,
- Power Down PHY and Power up PHY both set.
- Increase and Reduce Speed both set.
- Both a Speed update and a PHY power down action are requested.
- Do Nothing and another option are requested.
- HW independent algorithm and another option are requested.

11.6.3.19.1.1 Set Thermal Sensor Configuration Response (Intel Command 0x4D, Index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x4D	0x1		

11.6.3.19.2 Set Thermal Sensor Action Command (Intel Command 0x4D, Index 0x2)

This command executes actions immediately.

The *Actions* field describes the actions to activate according to Table 11-39.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x4D	0x02	Actions (MSB)	
24...25	Actions (LSB)			

Note: If one of the requested actions is not supported as defined in Get Thermal Sensor Capabilities Command (Section 11.6.3.18.2), the command fails with an Invalid Parameter reason.

Note: Actions set can not be contradictory - so the following combinations are invalid and result in a command fails with an Invalid Parameter reason:

- Indicate on SDP (set) and Indicate on SDP (clear) both set,
- Power Down PHY and Power up PHY both set.
- Restore and one of the Reduce Speed both set.
- Both a Speed update and a PHY power down action are requested.
- Do Nothing and another option are requested.
- HW independent algorithm and another option are requested.

11.6.3.19.2.1 Set Thermal Sensor Action Response (Intel Command 0x4D, Index 0x2)

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x4D	0x2		

11.6.3.20 Intel OEM AENs

11.6.3.20.1 Thermal Sensor AEN (Intel AEN 0x81)

The following is the AEN that may be sent by the NC following a Thermal Sensor event.

This AEN must be enabled using the NC-SI "AEN Enable" command, using bit 17 (0x20000) of the AEN Enable mask.

This AEN is sent on all the channels on which it is enabled.

11.6.4 Asynchronous Event Notifications

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI AEN Header			
4...7				
8...11				
12...15				
20...23	Reserved			0x81
24...27	Index of threshold crossed	Direction 0 = High Going 1 = Low Going	Measured Value	
28...31	Active Actions			

The asynchronous event notifications are unsolicited messages sent from the NC to the BMC to report status changes (such as link change, operating system state change, etc.).

Recommendations:

- The BMC 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 BMC 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.

11.6.5 Querying Active Parameters

The BMC can use the Get Parameters command to query the current status of the operational parameters.

11.6.6 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 BMC firmware designer keep in mind that following any type of reset, all configurations are considered as lost and thus the BMC must re-configure everything.
- As an asynchronous entry into the initial state might not be reported and/or explicitly noticed, the BMC 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 BMC should consider the channel (and most probably the entire NC package) as if it underwent a (possibly unexpected) reset event. Thus, the BMC 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.

11.6.7 Advanced Workflows

11.6.7.1 Multi-NC Arbitration

As described in Section 11.6.1.2, in a multi-NC environment, there is a need to arbitrate the NC-SI lines.

Figure 11-9 shows the system topology of such an environment.

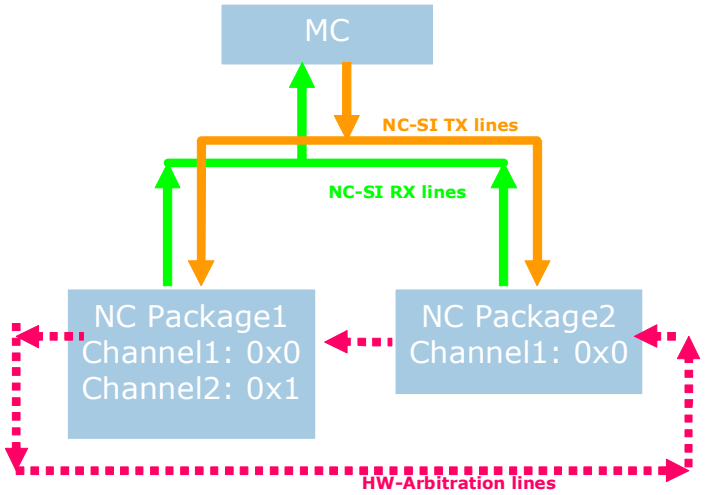


Figure 11-9. Multi-NC Environment

See Figure 11-9. 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 BMC. In this mechanism, the BMC is responsible for arbitrating between the packages by issuing NC-SI commands (Select/De-Select Package). The BMC 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 *NVM Control Word 1*.

For details, refer to the NC-SI specification.

11.6.7.2 Package Selection Sequence Example

Following is an example work flow for a BMC and occurs after the discovery, initialization, and configuration.

Assuming the BMC needs to share the NC-SI bus between packages, the BMC 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 BMC should:
   a. Issue a De-Select Package to the active_package. The BMC 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.

11.6.7.3 Multiple Channels (Fail-Over)

To support a fail-over scenario, it is required from the BMC 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 BMC 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.

11.6.7.3.1 Fail-Over Algorithm Example

The following is a sample workflow for a fail-over scenario for the X550 Quad-port 10 GbE controller (one package and four channels):

1. BMC initializes and configures all channels after power-up. However, the BMC uses the same MAC Address for all of the channels.
2. The BMC queries the link status of all the participating channels. The BMC 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 BMC then only enables channel 0 for network transmission.
4. The BMC 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 BMC begins normal workflow.
6. Should the BMC receive an indication (AEN or polling) that the link status for the active channel (channel 0) has changed, the BMC should:
    a. Disable channel0 for network transmission.
    b. Check if a different channel is available (link is up).
    c. If found:
        1. Enable network Tx for that specific channel.
        2. Issue a gratuitous ARP (or any other packet with its source MAC Address) to the network. This packet informs the switch that this specific MAC Address is registered to channel 0's specific LAN port.
        3. Resume normal workflow.
        4. If not found, report the error and continue polling until a valid channel is found.

The above algorithm can be generalized such that the start-up and normal workflow are the same. In addition, the BMC might need to use a specific channel (such as channel 0). In this case, the BMC 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.

11.6.7.4 Statistics

The BMC might use the statistics commands as defined in NC-SI. These counters are meant mostly 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 pass-through 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.

NC-SI pass-through statistics — These are statistics on the NC-SI pass-through frames. See the NC-SI specification for details.

11.6.8 External Link Control via NC-SI

The BMC can use the NC-SI Set Link command to control the external interface link settings. This command enables the BMC 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 X550 until proven otherwise).
- It is recommended that the BMC first query the link status using the Get Link Status command. The BMC 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.

11.6.8.1 Set Link While LAN PCIe Functionality is Disabled

In cases where the X550 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 link of higher a speed, the BMC should not advertise speeds that are below the desired link speed, as the lowest advertised link speed is chosen.

When the X550 is only used for manageability and the link speed advertisement is configured by the BMC, changes in the power state of the LAN device is not effected and the link speed is not re-negotiated by the LAN device.

11.6.8.2 Set Link Error Codes

The following rules are used to define the error code returned for Set Link command in case an invalid configuration is requested:

1. Host Driver Check: If 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 that, 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, 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, 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, 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).

11.6.8.3 Support for 2.5 and 5 Gb/s Speeds

As the Set Link and Get Link commands do not support the proprietary 2.5 and 5 Gb/s speeds, the following behavior is defined:

- A Get Link response returns a speed indication of 1 Gb/s if a speed of 2.5 or 5 Gb/s was negotiated.
- There is no way to request a specific speed of 2.5 or 5 Gb/s in a Set Link command.
- If both 1 Gb/s and 10 Gb/s are requested in the auto-negotiation, 2.5 and 5 Gb/s are also enabled.

11.7 MCTP

11.7.1 MCTP Overview

The Management Component Transport Protocol (MCTP) defines a communication model intended to facilitate communication between:

- Management controllers and other management controllers.
- Management controllers 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 PCI Express (DMTF's DSP0238) and over SMBus (DMTF's DSP0237). A specification for MCTP over USB is also planned. Management controllers 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.

11.7.1.1 NC-SI Over MCTP

MCTP is a transport layer protocol that do not include the functionality required to control the pass-through traffic required for BMC 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 NC-SI Over MCTP Specification.

The NC-SI over MCTP specification defines two types of MCTP message types: NC-SI (0x2) and Ethernet (0x3). The X550 supports both messages. When used only for control, only the NC-SI (0x2) message type is supported.

In addition to the above message types supported by the X550, 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 11.7.5.

11.7.1.2 MCTP Usage Model

The X550 supports NC-SI over MCTP protocol over the PCI Express and SMBus buses. The X550 can connect through MCTP to a BMC or the ME engine in the chipset as described in Figure 11-10.

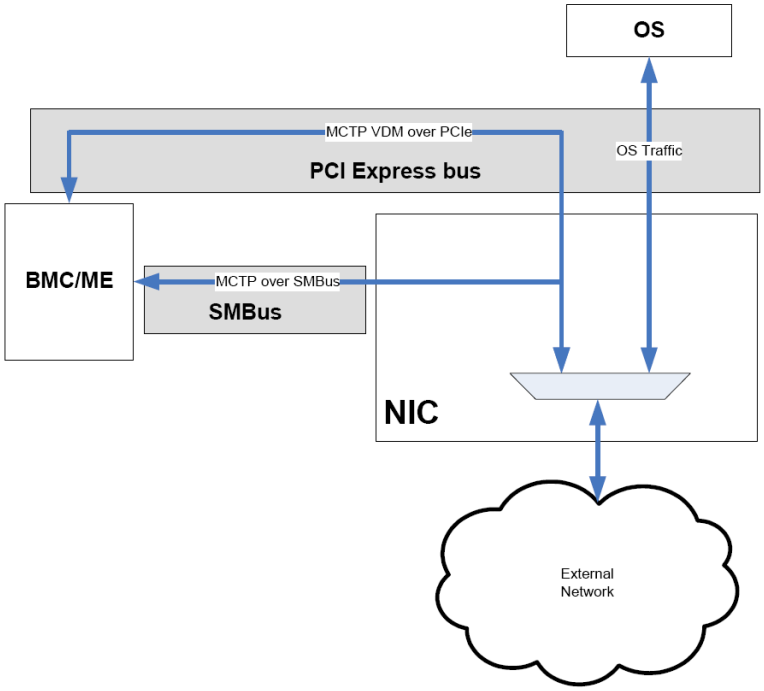


Figure 11-10. MCTP Connections of the X550

11.7.2 NC-SI to MCTP Mapping

The two network ports of the X550 (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 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 11.7.2.1 describes the transition between the two buses.

Both endpoints (SMBus and PCIe) may be active concurrently. However, pass-through traffic may be transferred only through one of them. If the PCIe endpoint is active, it is used for pass-through traffic. Otherwise, the SMBus endpoint is used. The *Set EID* command can be used to force the transition for the PCIe endpoint to the SMBus endpoint if the bus owner determines the PCIe channel is not functional.

For each channel (SMBus or PCIe), the X550 should expect MCTP commands from two sources: the bus owner and the BMC. In addition, it should expect pass-through traffic through one interface only. Thus, it should be able to process up to five interleaved commands/data:

- An MCTP control/OEM command from the PCIe bus owner (single packet message).
- An MCTP control/OEM command from the SMBus bus owner (single packet message).

- An MCTP control/OEM command from the BMC over SMBus (single packet message).
- An MCTP control/OEM command from the BMC over PCIe (single packet message).
- An NC-SI command or Ethernet packet from the BMC over the active channel.

A single source should not interleave packets it sends.

The topology used for MCTP connection is described in Figure 11-11.

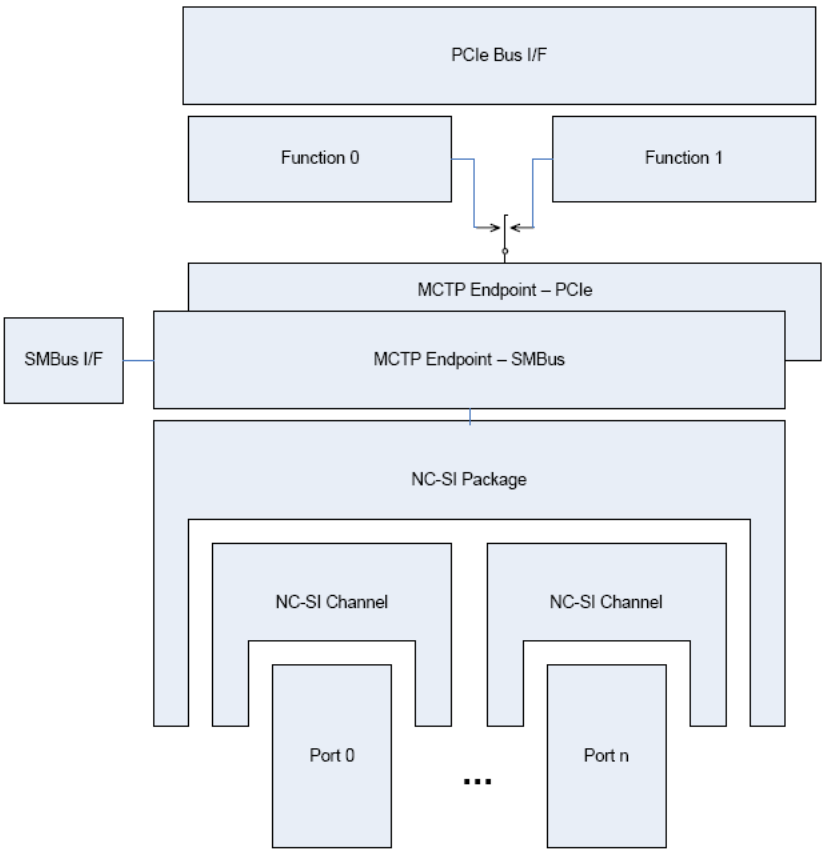


Figure 11-11. MCTP Endpoints Topology

11.7.2.1 Detection of BMC EID and Physical Address

To allow transactions between the BMC and the NIC, the bus physical address (SMBus or PCIe) and the EID of the partner needs to be discovered. The NICs do not try to discover the BMC, and assume the BMC initiates the connection. If the NIC is in NC-SI initial state, the EID and the physical address of the BMC are extracted from the *Clear Initial State* command parameters or any other NC-SI command received later with a channel ID of the X550. Subsequent pass-0through 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 BMC about the change.

11.7.2.2 Bus Transition

This section defines the transition flow between PCIe and SMBus as the bus on which MCTP flows. Figure 11-12 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
- BMC EID on PCIe
- BMC EID on SMBus
- BMC PCIe Target ID
- BMC 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 may be initialized from NVM value.

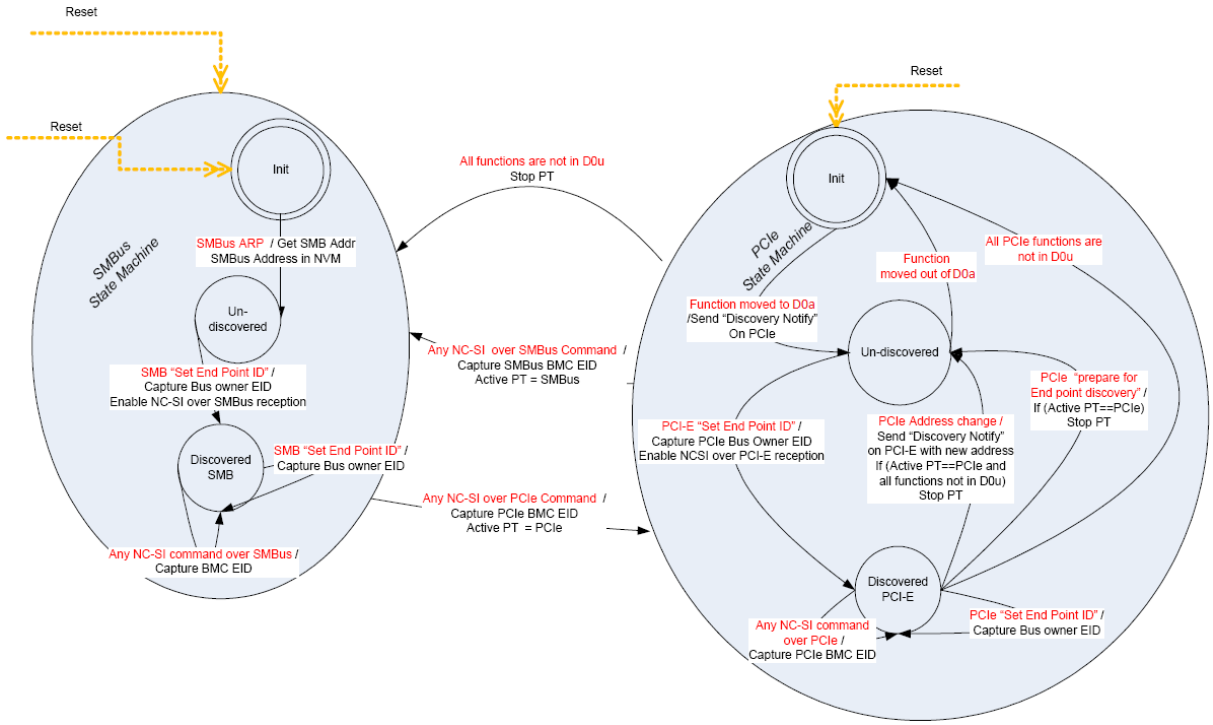


Figure 11-12. MCTP Bus Transition State Machine

11.7.2.2.1 Initial Assignment Flow

1. At power on, the NIC or BMC MCTP channel is connected to the SMBus, is not assigned an EID, and is in "Undiscovered" state.
2. The Bus Owner may preform an SMBus ARP cycle to assign an SMBus address to the NIC or to the BMC. Otherwise, a fixed address may be used. It is assumed that the SMBus address does not change after initialization time.
3. The Bus Owner performs an EID assignment using a *Set Endpoint ID* MCTP command. The NIC or the BMC 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/BMC EID from the *Destination Endpoint ID* field in the MCTP header as described in Section 10.3 of DSP0236. The NIC/BMC is now in "discovered" state.
4. The BMC may detect the NIC EID using one of the two following modes:
   — Static configuration of the NIC SMBus address in the BMC database and *Get Routing Table Entries* command to find the EID matching the SMBus address.
   — Get all endpoints through a *Get Routing Table Entries* command and find endpoints supporting NC-SI using the *Get Message Type Support* command for each endpoint.
5. Once the NIC is found, the BMC may send a *Clear Initial State* command to the NIC to start the NCSI configuration. The NIC captures the BMC SMBus address and BMC EID from any NC-SI command received.
6. After the NC-SI channels are enabled, traffic may be sent using the BMC and NIC addresses previously discovered.
7. The BMC may send a Get UUID command to get a unique identifier of the NIC that may be used later for re-connection upon topology changes.
8. If a firmware reset occurs, a *Discovery Notify MCTP* message should be sent by the device to restart the flow.

11.7.2.2.2 SMBus to PCIe Transition

1. If the NIC or the BMC detects that the PCIe bus is available by detecting a function that moved to D0 state, it may request a transition using a *Discovery Notify* MCTP command on the PCIe bus. This command should be sent with a "Route to Root-complex" addressing as described in DSP0238 Section 6.8. The source EID should be the EID previously assigned on the SMBus.
2. The bus owner should send an *Endpoint Discovery* command using Broadcast from RC routing to allow the NIC to register the Bus Owner EID allowed to set EIDs.
3. Upon reception of 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 may choose to wait for the *Discovery Notify* MCTP command of both the BMC and the NIC to do the transition. The Bus Owner should try to keep the EID previously assigned on the SMBus as the EID on PCIe bus
4. Upon reception of the *Set Endpoint ID* MCTP command, the NIC waits for an NC-SI command from the BMC indicating it is ready to transition the connection to PCIe. Upon reception of such a command, the NIC transitions its pass-through traffic to the PCIe bus using the newly-received addresses.

5. The BMC 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 BMC should not send any pass-through packets from the moment it sent the first NC-SI command on the PCIe and the moment a response is received for this command.
6. The transition of NC-SI traffic (pass-through or commands/responses) from SMBus to PCIe should be done on a packet boundary and should not interrupt a packet fragmentation or reassembly.

11.7.2.2.3 PCIe Target ID Change

The target ID of one of the endpoints may change, either due to a new enumeration of the PCIe bus or due to the disabling of one of the functions in the device (move to a non D0 state). In this case the following flow should be used:

1. The endpoint should send a *Discovery Notify* MCTP command on the PCIe bus using the new Requester ID.
2. Upon reception of the *Discovery Notify* MCTP command with the new Requester ID, the Bus Owner sends a *Set Endpoint ID* MCTP command on the PCIe bus, and updates the routing table. The Bus Owner should try to keep the EID previously assigned on the SMBus as the EID on the previous Requester ID.
3. The Bus Owner sends a *Routing Information Update* command to all supporting endpoints that may then update the parameters of their counterpart they use.
4. If a firmware reset occurs, the same flow is used to re-establish a connection.

11.7.2.2.4 PCIe to SMBus Transition

1. If the NIC or the BMC detects that the PCIe bus is not available by detecting a transition of all functions to a non D0 state, it stops using the PCIe for pass-through traffic or NC-SI traffic.
2. The BMC, upon detection of the unavailability of the PCIe bus, transitions the NC-SI channel to the MCTP over SMBus as described above.

Note: The transition of NC-SI traffic (pass-through or commands/responses) from PCIe to SMBus may done at any stage and may interrupt a packet fragmentation or reassembly, as it is assumed that such a transition occurs only when the PCIe bus is not available anymore.

11.7.3 MCTP Over PCIe

11.7.3.1 Message Format

The message format used for NC-SI over MCTP over PCIe is as follows:

+0								+1								+2								+3							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
FMT 011			Type 10r2r1r0[1]					R	TC 000			R	Attr[2]	R	TH[2]	TD[2]	EP[2]	Attr[1:0][2]		AT 00		Length 00_000x_xxxx											
PCI Requester ID																PCI Tag Field								Message Code Vendor Defined = 0111_1111b							
																R		Pad Len[3]		MCTP VDM code - 0000b											
PCI Target ID (For Route by ID messages, otherwise = Reserved)																Vendor ID = 0x1AB4 (DMTF)															
MCTP Reserved				Header version = 1				Destination endpoint ID								Source endpoint ID								SOM	EOM	SEQ#			TO	Tag			
IC[2]	Message Type = 0x02							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
3. Defines the number of 0x00 pad bytes that have been added to the end of the packet to make the packet DWord-aligned with respect to PCIe. Expected only in the last packet of a message (EOM = 1).

PCIe TLP header
MCTP header
NC-SI header and payload

11.7.3.2 PCIe Discovery Process

The X550 follows the discovery process described in Section 5.9 of the *MCTP PCIe VDM Transport Binding Specification* (DSP0238).

Upon reception of an Endpoint Discovery message (while in undiscovered stage), the X550 exposes the endpoint on the selected function as described above.

If the *Bus Master Enable* bit of the selected function is cleared after the endpoint was discovered, or if the bus number of the X550 changes due to a re-enumeration of the bus, the X550 sends a Discovery Notify message to indicate to the BMC that it should do a re-enumeration of the device to discover the new endpoint.

11.7.3.3 MCTP Over PCIe Special Features

The X550 supports the following optional features of MCTP when running over PCIe:

- Rate Limiting
- ACLs

11.7.3.3.1 MCTP Uplink Rate Limiting

As the PCIe link can carry a traffic bandwidth much higher than what the BMC can sustain, to avoid drop of packets, the X550 allows rate limiting of the MCTP pass-through traffic. The X550 supports rate limiting between 1 Mb/s and 1 Gb/s. The following parameters defines 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).

11.7.3.3.2 Service Provider MCTP Endpoint ACLs

The X550 supports a set of ACLs that allows reception of sensitive commands only from 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 bit 5 in *NC-SI over MCTP Configuration* NVM word - Section 6.2.17.13) the following flow is used decide which packets are accepted.

Commands can be divided to 3 types:

- ACL programming commands: Such commands can be received only from the address that sent the *Prepare For Endpoint Discovery* command via broadcast routing.
- Sensitive commands including all the NC-SI commands and pass-through traffic. These commands can be received only from requesters whose bus number is set in the ACL list. If an MCTP packet is dropped, the SPMEACLD counter is increased. This counter can be read by the MCTP bus owner using the *Get ACL Violation Counters* command.
- Regular MCTP commands are received from any requester. However, the *Set EID* command is processed only if received from the address that sent the *Prepare For Endpoint Discovery* command via broadcast routing.

11.7.4 MCTP Over SMBus

The message format used for NC-SI over MCTP over SMBus is as follows:

+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							NC-SI Command/Pass-Through data																							
.....																															
NC-SI Command/Pass-Through data																															
PEC																															

1. IC = 0

SMBus header/PEC
MCTP header
NC-SI header and payload

11.7.4.1 SMBus Discovery Process

The X550 follows the discovery process described in Section 6.5 of the *MCTP SMBus/I^{2}C Transport Binding Specification* (DSP0237). It indicates support for ASF in the SMBus getUID command (see Section 11.5.5.4). It responds to any SMBus command using the MCTP command code - so that the bus owner knows that the X550 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.

11.7.4.2 MCTP over SMBus Special Features

The X550 supports the following optional features of MCTP when running over SMBus:

- Simplified MCTP mode
- Fairness arbitration

11.7.4.2.1 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* & *EOM* bits. This bit is set according to the *Simplified MCTP* bit in the *NC-SI over MCTP Configuration* word in the NVM.

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).

11.7.4.2.2 Fairness Arbitration

When sending MCTP messages over SMBus, the X550 should respect the fairness arbitration as defined in Section 5.13 of DSP0237 when sending MCTP messages.

11.7.5 NC-SI Over MCTP

X550 support for NC-SI over MCTP is similar to the support for NC-SI over RMII with the following exceptions:

- A set of new NC-SI OEM commands used to expose the NC-SI over MCTP capabilities.
- The format of the packets is modified to account for the new transport layer as described below.

11.7.5.1 NC-SI Packets Format

NC-SI over MCTP defines two different message types 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), while 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

11.7.5.1.1 Control Packets

The format used for Control packets (Commands, Responses, and AENs) is as follows:

+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 11.2.2.2

SMBus/PCIe header
MCTP header
NC-SI header
NC-SI Data

Note that the MAC header and MAC FCS present when working over NC-SI are not part of the packet in MCTP mode.

11.7.5.1.2 Pass-Through Packets

The format used for pass-through packets is 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 X550. In transmit, the CRC is added by the X550.

+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																															
....																															
....																															

11.7.6 MCTP Programming

The MCTP programming model is based on:

- A set of MCTP commands used for the discovery process and for the link management. The list of supported commands is described in Section 11.7.6.1.
- A subset of the NC-SI commands used in the regular NC-SI interface, including all the OEM commands as described in Section 11.6.2 (NC-SI programming I/F). The specific commands supported are listed in Table 11-27 and Table 11-30.

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.

11.7.6.1 MCTP Commands Support

Table 11-41 lists the MCTP commands supported by the X550.

Table 11-41. MCTP Commands Support

Command Code	Command Name	General Description	X550 Support as Initiator	X550 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 11.7.6.1.1 for details.	No	Yes
0x03	Get Endpoint UUID	Retrieves a per-device unique UUID associated with the endpoint. See Section 11.7.6.1.2 for details.	No	Yes
0x04	Get MCTP Version Support	Lists which versions of the MCTP control protocol are supported on an endpoint. See Section 11.7.6.1.3 for details.	No	Yes
0x05	Get Message Type Support	Lists the message types that an endpoint supports. See Section 11.7.6.1.3.1 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 11.7.6.1.4 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 pass for a given message type when routing to the target endpoint.	No	No

1. These commands are supported only for MCTP over PCIe.

11.7.6.1.1 Get Endpoint ID

The Get Endpoint ID response of the X550 is described in Table 11-42.

Table 11-42. Get Endpoint ID Response

Byte	Description	Value
1	Completion Code	
2	Endpoint ID	0x00 = EID not yet assigned.. Otherwise = Returns EID assigned using Set Endpoint ID command
3	Endpoint Type	0x00 (Dynamic EID, Simple Endpoint).
4	Medium Specific	0x00 = PCIe 0x01 = SMBus — Fairness arbitration protocol supported.

11.7.6.1.2 Get Endpoint UUID

The UUID returned is calculated according to the following functions:

Time Low = Read from MCTP UUID - Time Low LSB/MSB NVM words of Sideband Configuration Structure.

Time mid = Read from Read from MCTP UUID - Time Mid NVM word of Sideband Configuration Structure

Time High and version = Read from MCTP UUID - Time High and version NVM word of Sideband Configuration Structure

Clock Sec and Reserved = Read from MCTP UUID - Clock Seq NVM word of Sideband Configuration Structure

Node = MAC Address as taken from the PCI_SERL and PCI_SERH registers.

11.7.6.1.3 Get MCTP Version Support

Table 11-43 describes the returned value according to the requested message type. The list of supported message types is based on the protocols enabled in the NVM and should be the same as the list reported in the *Get Message Type Support* command in Section 11.7.6.1.3.1.

Table 11-43. Get MCTP Version Support Returned Value

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)	0xF1F0F000 (1.0)	0xF1F0F000 (1.0)	0xF1F0FF00 (1.0)	0
10:7	Version number entry 2	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)	
14:11	Version number entry 3	0xF1F2F000 (1.2.0)	0xF1F2F000 (1.2.0)				

11.7.6.1.3.1 Get Message Type Support Command

The Get Message Type Support response of the X550 is described in Table 11-44.

Table 11-44. Get Message Type Support Response

Byte	Description	Value
1	Completion Code	0x00
2	MCTP Message Type Count	0x01/0x02/0x03 - The X550 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 pass-through is supported.
		0x7E (PCIe based VDM messages). Over PCIe. Also over SMBus if OEM commands are supported.

11.7.6.1.4 Get Vendor Defined Message Support Command

This command is supported only for MCTP over PCIe.

The Get Vendor Defined Message Support response of the X550 is described in Table 11-45 if the Vendor ID Set Selector equals 0x00.

Table 11-45. Get Vendor Defined Message Support Response

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)

11.7.6.1.5 Set Endpoint ID Command

The X550 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 X550 can be set only through their own interface, Set EID and Force EID are equivalent. The Reset EID operation is not supported by the X550.

The Set Endpoint ID response of the X550 is described in Table 11-46.

Table 11-46. Set Endpoint ID Response

Byte	Description	Value
1	Completion Code	0x00
2	Completion Status	[1:0] = 00b - Device does not use an EID pool. [3:2] = 00b - Reserved. [5:4] = 00b - EID assignment accepted. [7:6] = 00b - Reserved.
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.

11.8 Manageability Host Interface

This section details host interaction with the manageability portion of the X550. The information within this section is only available to the host driver, the BMC does not have access.

11.8.1 HOST CSR Interface (Function 1/0)

The software device driver of all functions communicates with the manageability block through CSR access. The manageability is mapped to address space 0x15800 to 0x15FFF on the slave bus of each function.

Note: Writing to address 0x15800 from any function is targeted to the same address in the RAM.

11.8.2 Host Slave Command Interface to Manageability

This interface is used by the software device driver for several of the commands and for delivering various types of data in both directions (Manageability-to-Host and Host-to-Manageability).

The address space is separated into two areas:

- Direct access to the internal data RAM: The internal shared (between firmware and software) RAM is mapped to address space 0x15800 to 0x15EFF. Writing/reading to this address space goes directly to the RAM.
- Control register is located at address 0x15F00.

11.8.2.1 Host Slave Command Interface Low Level Flow

This interface is used for the external host software to access the manageability subsystem. Host software writes a command block or read data structure directly from the data RAM. Host software controls these transactions through a slave access to the *HICR* control register (see Section 8.2.2.20.5).

The following flow shows the process of initiating a command to the manageability block:

1. The software clears the FWSTS.*FWRI* flag (clear by write one) to clear any previous firmware reset indications.
2. The software device driver takes ownership of the Management Host interface using the flow described in Section 4.7.
3. The software device driver reads the HOST Interface Control register (see Section 8.2.2.20.5) and checks that the *Enable* (HICR.*EN*) bit is set.
4. The software device driver writes the relevant command block into the RAM area that is mapped to addresses 0x15800-0x15EFF.
5. The software device driver sets the *Command* (HICR.*C*) bit in the HOST Interface Control register (see Section 8.2.2.20.5). Setting this bit causes an interrupt to the ARC.
6. The software checks the FWSTS.*FWRI* flag to make sure a firmware reset didn't occur during the command processing. If this bit is set, the command may have failed.
7. The software device driver polls the HOST Interface Control register for the *Command* (HICR.*C*) bit to be cleared by firmware. The command should complete within half a second.

8. When firmware finishes with the command, it clears the *Command* (HICR.*C*) bit (if firmware replies with data, it should clear the bit only after the data is placed in the shared RAM area where the software device driver can read it).

If the software device driver reads the HOST Interface Control register and the HICR.*SV* bit is set to 1b, there is a valid status of the last command in the shared RAM. If the HICR.*SV* bit is not set, the command has failed with no status in the RAM.

On completion of access to the shared RAM software device driver should release ownership of the shared RAM using the flow described in Section 4.7.

11.8.2.2 Host Interface Structure

11.8.2.2.1 Host Interface Command Structure

Table 11-47 describes the structure used by the software device driver to send a command to firmware using the Host slave command interface (shared RAM mapped to addresses 0x15800-0x15EFF).

Table 11-47. Host Driver Command Structure

Byte(s)	Description	Bit(s)	Value	Description
0	Command	7:0	Command Dependent	Specifies which host command to process.
1	Buffer Length	7:0	Command Length	Command Data Buffer length: 0 to 252, not including 32 bits of header.
2	Reserved	7:0		Reserved.
3	Checksum	7:0	Defined Below	Checksum signature. If the value is 0xFF, the checksum is not checked by the firmware.
255:4	Data Buffer	7:0	Command Dependent	Command Specific Data Minimum buffer size = 0 Maximum buffer size = 252

11.8.2.2.2 Host Interface Status Structure

Table 11-48 lists the structure used by firmware to return a status to the software device driver via the Host slave command interface. A status is returned after a command has been executed.

Table 11-48. Status Structure Returned to Host Driver

Byte(s)	Description	Bit(s)	Value	Description
0	Command	7:0	Command Dependent	Command ID.
1	Buffer Length	7:0	Status Dependent	Status buffer length: 252:0
2	Return Status	7:0	Depends on Command Executing Results	0x1 = Status OK 0x2 = Illegal command ID 0x3 = Unsupported command 0x4 = Illegal payload length 0x5 = Checksum failed 0x6 = Data Error 0x7 = Invalid parameter 0x8–0x7F = Reserved 0x80–0xFF = Command Specific Errors. May be used by individual commands for command specific errors

Table 11-48. Status Structure Returned to Host Driver [continued]

Byte(s)	Description	Bit(s)	Value	Description
3	Checksum	7:0	Defined Below	Checksum signature.
255:4	Data Buffer		Status Dependent	Status configuration parameters Minimum Buffer Size = 0 Maximal Buffer Size = 252 If Return Status is not "Status OK" the Data Buffer is empty.

11.8.2.2.3 Checksum Calculation Algorithm

The Host Command/Status structure is summed with this field cleared to 0b. The calculation is done using 8-bit unsigned math with no carry. The inverse of this sum is stored in this field (0b minus the result). Result: The current sum of this buffer (8-bit unsigned math) is 0b.

11.8.3 Host Interface Commands

11.8.3.1 Driver Info Host Command

This command is used to provide the driver information in NC-SI mode.

Table 11-49. Driver Info Host Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0xDD	Driver info command.
1	Buffer Length	7:0	0x5 + driver string length	Port Number + 4 bytes of the Driver info + driver string length (total length can be up to 255 bytes).
2	Reserved	7:0	0x0	Reserved.
3	Checksum	7:0		Checksum signature of the Host command.
4	Function Number	7:0	Function Number	Indicates the function currently reporting its driver info.
8:5	Driver Version	7:0	Driver Version	Numerical for driver version - should be: Byte 8 = Major Byte 7 = Minor Byte 6 = Build Byte 5 = SubBuild ***Note:*** The old version is kept to support CEM version 1 commands.
9:9 + driver string length	Driver String		Driver String	Driver string (not null terminated) as reported by driver. For example, in Linux the DRV_VERSION #define, or in NDIS, the OID_GEN_VENDOR_DRIVER_VERSION. Can be up to 250 bytes.

Following is the status returned on this command:

Table 11-50. Driver Info Host Status

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0xDD	Driver Info command.
1	Buffer Length	7:0	0x0	No data in return status.
2	Return Status	7:0	0x1	See Table 11-48.
3	Checksum	7:0		Checksum signature.

11.8.3.2 Disable RXEN Command

This command is used to enable the driver to request a safe disable of Receive Enable.

Table 11-51. Disable RXEN Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0xDE	Driver info command.
1	Buffer Length	7:0	0x1	1 data byte attached to this command (the port number).
2	Reserved	7:0	0x0	Reserved.
3	Checksum	7:0		Checksum signature of the Host command.
4	Port Number	7:0	Port Number	Indicates the port for which the operation is requested.

Following is the status returned on this command:

Table 11-52. OS2BMC Control Status

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0xDE	Driver Info command.
1	Buffer Length	7:0	0x0	No data in return status.
2	Return Status	7:0	0x1	See Table 11-48.
3	Checksum	7:0		Checksum signature.

The firmware should execute the following flow when receiving this command:

- Store OS2BMC enable bit.
- Clear OS2BMC enable bit.
- Clear RXEN
- Restore OS2BMC enable bit to original value.
- Return from command.

11.8.3.3 Flash I/F interface

These commands allow buffers of up to 1Kbyte of data. To do this, the *Buffer Length* field is expanded to 2 bytes.

Note: This *Buffer Length* field in the command is in big endian order. For example, byte 1 contains the MSB part of the buffer length and byte 2 contains the LSB part of the buffer length. The buffer length in the response is in the reverse order (byte 1 contains the LSB and byte 2 [7:5] contains the MS bits).

11.8.3.3.1 Flash Read

This command is used to request a read from the flash. This command allows access to the region of the flash owned by the device (in case of non-shared SPI, it is the entire flash).

This command always returns the value from the flash, even if read within the Shadow RAM boundaries.

Table 11-53. Flash Read Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x30	Flash Read.
2:1	Buffer Length	15:0	0x6	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to read from the flash.
9:8	Length to Read	15:0		How many bytes to read. Can be up to 1024.

Table 11-54. Flash Read Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x30	Flash Read.
1	Buffer Length (LS Byte)	7:0		Buffer length[7:0].
2	Buffer Length (MS Byte)	7:5		Buffer length[10:8].
2	Return Status	4:0		See Table 11-48.
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to read from the flash.
9:8	Length Read	15:0		How many bytes where actually read.
11:10	Reserved			Reserved.
12: Length Read+12	Read Data			The Requested Data read from flash.

11.8.3.3.2 Shadow RAM Read

This command is used to request a read from the Shadow RAM. Requesting addresses above Shadow RAM boundaries causes an error.

Table 11-55. Shadow RAM Read Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x31	Shadow RAM Read.
2:1	Buffer Length	15:0	0x6	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to read from the Shadow RAM.
9:8	Length to Read	15:0		How many bytes to read. Can be up to 1024.

Table 11-56. Shadow RAM Read Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x31	Shadow RAM Read.
1	Buffer Length (LS Byte)	7:0		Buffer length[7:0].
2	Buffer Length (MS bits)	7:5		Buffer Length[10:8].
2	Return Status	4:0		See Table 11-48.
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to read from the Shadow RAM.
9:8	Length Read	15:0		How many bytes where actually read.
11:10	Reserved			Reserved.
12: Length Read+12	Read Data			The Requested Data read from Shadow RAM.

11.8.3.3.3 Flash Write

This command is used to update the flash sections out of the Shadow RAM. This command allows access to the writable region of the flash owned by the device that is not part of the Shadow RAM. If not all the area to erase is writable, the command returns an error.

Table 11-57. Flash Write Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x32	Flash Write.
2:1	Buffer Length	15:0	0x8 + Length to Write	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to write to the Shadow RAM.
9:8	Length to Write	15:0		How many bytes to write. Can be up to 1024.
11:10	Reserved			Reserved.
12: Length to Write +12	Write Data			The data to write to the flash.

Table 11-58. Flash Write Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x32	Flash Write.
1	Buffer Length	7:0	0x6	
2	Return Status	7:0		See Table 11-48.
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to write to the Shadow RAM.
9:8	Data Written	15:0		How many bytes where actually written.

11.8.3.3.4 Shadow RAM Write

This command is used to update the Shadow RAM. It allows write to the writable parts of the Shadow RAM. If not all the area to write is writable, or not all the area is in the Shadow RAM range, the command returns an error.

Table 11-59. Shadow RAM Write Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x33	Shadow RAM write.
2:1	Buffer Length	15:0	0x8 + Length to Write	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to write to the flash.
9:8	Length to Write	15:0		How many bytes to write. Can be up to 1024.
11:10	Reserved			
12: Length to Write+12	Write Data			The data to write to the Shadow RAM

Table 11-60. Shadow RAM Write Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x33	Shadow RAM Write.
1	Buffer Length	7:0	0x6	
2	Return Status	7:0		See Table 11-48.
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to write to the Shadow RAM.
9:8	Data Written	15:0		How many bytes where actually written.

11.8.3.3.5 Flash Module Update

This command is used to update a secured module. After a successful response to this command, for a firmware code update, an *Apply Update* command (Section 11.8.3.3.6) is sent to activate the new firmware.

Table 11-61. Flash Module Update Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x34	Flash Module Update.
2:1	Buffer Length	15:0	0x1	
3	Checksum	7:0		Checksum signature.
4	Module ID	7:0		Which module to update: Firmware code = 0x1 PHY Firmware Image = 0x5 Option ROM = 0xFE

Table 11-62. Flash Module Update Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x34	Flash Module Update.
1	Buffer Length	7:0	0x0	
2	Return Status	7:0		See Table 11-48. If the Authentication failed, an Authentication Error (0x80) is returned
3	Checksum	7:0		Checksum signature.

11.8.3.3.6 Apply Update

This command is used to request the firmware to switch to the new uploaded code. This command involves a firmware reset, so no response should be expected after this command is given. The software device driver may read the *FWRESETCNT* register before and after the command is given to check if the reset took place.

Note: An Apply Update command sent not after a successful Flash Module Update Command is ignored.

If after 100 ms the *FWRESETCNT* register has not increase, the software should assume the command failed.

Table 11-63. Apply Update Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x38	Apply Update.
2:1	Buffer Length	15:0	0x0	
3	Checksum	7:0		Checksum signature.

11.8.3.3.7 Flash Block Erase

This command is used to erase some of the flash sections. This command allows access to the writable region of the flash owned by the device that is not part of the Shadow RAM. If not all the area to erase is writable, the command returns an error.

Table 11-64. Flash Block Erase Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x35	Flash Block Erase.
2:1	Buffer Length	15:0	0x5	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address of block to erase – must be 4K bytes aligned.
8	Number of sectors to erase	7:0		How many 4KB sectors to erase. To avoid a too long command, this number should be no larger than 3.

Table 11-65. Flash Block Erase Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x35	Flash Block Erase.
1	Buffer Length	7:0	0x5	
2	Return Status	7:0		See Table 11-48.
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address of block to erase.
8	Erased Blocks	7:0		How many blocks where actually erased.

11.8.3.3.8 Shadow RAM Dump

This command is used to trigger a Shadow RAM dump. It should be used after updating a module in Shadow RAM.

Table 11-66. Shadow RAM Dump Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x36	Shadow RAM dump.
2:1	Buffer Length	15:0	0x0	
3	Checksum	7:0		Checksum signature.

Table 11-67. Shadow RAM Dump Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x36	Shadow RAM dump.
1	Buffer Length	7:0	0x0	
2	Return Status	7:0		See Table 11-48.
3	Checksum	7:0		Checksum signature.

11.8.3.3.9 Flash Info

This command is used to provide information about the Flash.

Note: If the connected Flash does not support SFDP, the JEDEC ID information – *Valid* field is not set.

Table 11-68. Flash Info Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x37	Flash Info.
2:1	Buffer Length	15:0	0x0	
3	Checksum	7:0		Checksum signature.

Table 11-69. Flash Info Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x37	Flash Info.
1	Buffer Length	7:0	0x8	
2	Return Status	7:0		See Table 11-48.
3	Checksum	7:0		Checksum signature.
7:4	Flash Size	31:0		Size of flash available to this device in bytes. In case on non-shared SPI, it is the size of the flash. In case of SFDP flash, the size from the SFDP data is reported, otherwise the value from NVM is reported.
8	JEDEC ID information – Bank	6:0		Defines the bank number of the manufacturer ID (number of 0x7F read).
	JEDEC ID information – Valid	7		The JEDEC ID information is valid (is not set in shared SPI modes).
9	JEDEC ID information – Manufacturer ID	7:0		Returns the manufacturer ID read by RDID command.
10	JEDEC ID information – Device ID 1st byte	7:0		First byte of the device ID read by RDID command (*Family* and *Density* fields).
11	JEDEC ID information – Device ID 2nd byte	7:0		Second byte of the device ID read by RDID command (Sub version and Revision).

11.8.4 Software and Firmware Synchronization

Software and firmware synchronize accesses to shared resources in the X550 through a semaphore mechanism and a shared configuration register between the software device driver of the two ports and the firmware. This semaphore enables synchronized accesses to the following shared resources:

- NVM
- PHY 0 and PHY 1 registers
- MAC (LAN controller) shared registers. This semaphore protects the MAC registers described in Section 8.2.2.16.
- I^2C

The SW_FW_SYNC.*REGSMP* bit is used as a semaphore mechanism between software and firmware. Once software or firmware takes control over this semaphore flag, it can access the SW_FW_SYNC Register and claim ownership over specific resources. The SW_FW_SYNC includes pairs of bits (one owned by software and the other by firmware), where each pair of bits controls a different resource. A resource is owned by software or firmware when its respective bit is set. It is illegal to have both pair bits set at the same time. Described below are the required sequences for gaining and releasing control over shared resources.

11.8.4.1 Gaining Control of Shared Resource by Software

1. The software device driver checks that the software device driver of the other LAN function does not use the software/firmware semaphore.
   - The software device driver polls the SWSM.*SMBI* bit till it is read as 0b, or time expires (recommended expiration is ~10 ms + expiration time used for the SW_FW_SYNC.*REGSMP*).
   - If the SWSM.*SMBI* is found at 0b, the semaphore is taken. Note that following this read cycle the hardware auto sets the bit to 1b.
   - If time expired, it is assumed that the software device driver of the other function malfunctioned. The software proceeds to the next step.
2. The software device driver checks that the firmware does not use the software/firmware semaphore and then takes its control.
   - Software polls the SW_FW_SYNC.*REGSMP* bit till it is read as 0b, or time expires (recommended expiration is ~50 ms). If time has expired the software assumes that the firmware malfunctioned and proceeds to the next step, while ignoring the firmware bits in the SW_FW_SYNC register.
3. The software takes control of the requested resource(s).
   - The software device driver reads the firmware and software bit(s) of the requested resource(s) in the SW_FW_SYNC register. If the bit(s) is cleared, the resource(s) is accessible (i.e. no other entity owns the resource(s)). In this case the software device driver sets the software bit(s) of the requested resource(s) in the SW_FW_SYNC register. The software then clears the SW_FW_SYNC.*REGSMP* and SWSM.*SMBI* bits (releasing the software/firmware semaphore register), and can use the specific resource(s).
   - Otherwise (i.e. either firmware or software of the other LAN function owns the resource), the software clears the SW_FW_SYNC.*REGSMP* and SWSM.*SMBI* bits and then repeats the entire process after some delay (recommended 5-10 ms).
     - If the resources are not released by the software device driver of the other LAN function in a timely manner (recommended expiration time is ~1 sec), the software device driver can assume that the other software device driver malfunctioned. In that case the software device driver should clear all software flags that it does not own (including SW_FW_SYNC.*REGSMP* bit) and then repeat the entire process once again.
     - If the resource is not released by the firmware (recommended expiration time for firmware is ~50 ms) the software device driver can assume that the firmware malfunctioned. In that case the software device driver should sets the software bit(s) of the requested resource(s), while ignoring the corresponding firmware bits in the SW_FW_SYNC register.

Note: The firmware initializes its semaphore flags as part of its init flow. The software semaphores are not reset.

11.8.4.2 Releasing a Shared Resource by Software

1. The software device driver takes control over the software/firmware semaphore as described above for gaining shared resources.
2. The software device driver clears the bit(s) of the released resource(s) in the SW_FW_SYNC register.
3. The software device driver releases the software/firmware semaphore by clearing the SW_FW_SYNC.*REGSMP* and SWSM.*SMBI* bits
4. The software device driver should delay (recommended 5-10 ms) before trying to gain the semaphore immediately following its release.

11.8.4.3 Gaining Control of Shared Resource by Firmware

1. The firmware takes control over the software/firmware semaphore (SW_FW_SYNC register)
   — The firmware polls the SW_FW_SYNC.*REGSMP* bit till it is read as 0b, or timeout expires (recommended expiration time is ~10 ms).
   — If timeout has expired the firmware clears the SW_FW_SYNC.*REGSMP* bit (it is assumed the software device driver is not functional).
2. The firmware takes ownership of the requested resources
   — The firmware reads the software bit(s) corresponding to the requested firmware resource(s) in the SW_FW_SYNC register.
   — If the software bit is cleared (i.e. the software device driver does not own the resource), the firmware sets the firmware bit(s) of the requested resource(s). The firmware then clears the SW_FW_SYNC.*REGSMP* bit (releasing the software/firmware semaphore) and can use the specific resource(s).
   — Otherwise (i.e. software owns the resource), the firmware clears the SW_FW_SYNC.*REGSMP* bit and then repeats the above process after some delay (recommended delay of 5-10 ms).
      - If the resources owned by software are not released in a timely manner (~1 sec), the firmware "forces" its ownership over the requested resources. The firmware clears the software flags of the requested resources in the SW_FW_SYNC register (assuming the software that set those flags is not functional).

11.8.4.4 Releasing a Shared Resource by the Firmware

1. The firmware takes control over the software/firmware semaphore as described above for gaining shared resources.
2. The firmware clears the bit(s) of the selected resource(s) in the SW_FW_SYNC register.
3. The firmware releases the software/firmware semaphore by clearing the SW_FW_SYNC.*REGSMP* bit.
4. The firmware should delay before trying to gain the selected resource semaphore immediately following its release (recommended 5-10 ms).

11.9 Host Isolate Support

If a BMC decides that a malicious software prevents its usage of the LAN, it may decide to isolate the NIC from its driver. This is done using the TCO reset command (Section 11.6.3.14).

If TCO isolate is enabled in the NVM (Section 6.2.16), The TCO Isolate command disables PCIe write operations to the LAN port. As the driver needs to access the CSR space to provide descriptors to the NIC, this operation also stops the network traffic including OS-to-BMC and BMC-to-OS traffic as soon as the existing transmit and receive descriptor queues are exhausted.

Chapter 12 Electrical/Mechanical Specification

12.1 Introduction

This section describes the X550 DC and AC (timing) electrical characteristics and the X550 package specification. This includes absolute maximum rating, recommended operating conditions, power sequencing requirements, DC and AC timing specifications. The DC and AC characteristics include generic digital IO specification as well as other specifications of interfaces supported by the X550.

12.2 Operating Conditions

12.2.1 Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Units
T_{case}	Case Temperature Under Bias	0	107	°C
$T_{storage}$	Storage Temperature Range	-50	150	°C
VCC3P3	3.3 Voltage	-500	3630	mV
VCC2P1	2.1 Voltage	-500	2320	mV
VCC1P2	1.2 Voltage	-500	1320	mV
VCC0P83	0.83 Voltage	-500	1190	mV
V_{ESD}	ESD On All Pins Except Line		2	kV
V_{CD}	Cable Discharge On Line		3	kV

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.

12.2.2 Recommended Operating Conditions

Symbol	Parameter	Min	Typ	Max	Units
T_a	Operating Temperature Range Commercial (Ambient; 0 CFS airflow)	0		55	°C
T_j	Junction Temperature			105	°C

Notes:

1. 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.
2. Recommended operation conditions require accuracy of power supply of ±5% relative to the nominal voltage.
3. External Heat Sink (EHS) is needed.
4. Refer to Chapter 14 for a description of the allowable thermal environment.

12.3 Power Delivery

12.3.1 Power Delivery Definitions

- **DC Voltage Regulation** — DC accuracy with respect to the nominal voltage specification. Includes controller feedback accuracy and resistive losses in the power distribution network.
- **AC Voltage Regulation** — Minimum/maximum noise voltage based on mid-to-high frequency (~1-20 MHz) AC load currents (excluding ripple voltage). Note that this specification is met by following the system design recommendations/considerations related to the recommended decoupling design (see Chapter 13, "Design Considerations and Guidelines").
- **Total Line Regulation** — The sum of DC and AC voltage regulation.

12.3.2 Power Supply Specifications

Description	Parameter
VCC3P3V	
Nominal Voltage	3.3 V
DC Voltage Regulation	± 5% DC Regulation
AC Voltage Regulation	± 4% AC Regulation (132 mV Pk-Pk)
Total Line Regulation	± 9%
VCC2P1	
Nominal Voltage	2.1 V
DC Voltage Regulation	± 3% DC Regulation
AC Voltage Regulation	± 0.6% AC Regulation (13 mV Pk-Pk)
Total DC/AC Voltage Regulation	± 3.6%
Step Load Size	64 mA
Step Load Slew Rate di/dt	200 mA/1 µs
VCC1P2	
Nominal Voltage	1.2 V
DC Voltage Regulation	± 3% DC Regulation
AC Voltage Regulation	± 2% AC Regulation (24 mV Pk-Pk)
Total DC/AC Voltage Regulation	± 5%
Step Load Size	125 mA
Step Load Slew Rate di/dt	125 mA/1 µs
VCC0P83	
Nominal Voltage	0.83 V
DC Voltage Regulation	± 3% DC Regulation
AC Voltage Regulation	± 2% AC Regulation (16.6 mV Pk-Pk)

Description	Parameter
Total DC/AC Voltage Regulation	± 5%
Step Load Size	2 A
Step Load Slew Rate di/dt	587 mA/1 µs

12.3.3 VCC3P3 External Power Supply Specification (3.3 V)

Parameter	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark.	0.1	50	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)	48	28800	V/S
Operational Range	Voltage range for normal operating conditions.	3	3.6	V
Overshoot	Maximum overshoot allowed.	N/A	100	mV
Overshoot Settling Time	Maximum overshoot allowed duration. ***Note:*** At that time delta voltage should be lower than 5 mV from steady state voltage.	N/A	0.05	ms
Suggested Decoupling Capacitance	Capacitance range.	25	N/A	µF

12.3.4 VCC2P1 External Power Supply Specification (2.1 V)

Title	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark.	0.1	50	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)	32	17600	V/S
Operational Range	Voltage range for normal operating conditions.	2.0	2.2	V
Overshoot	Maximum overshoot allowed.	N/A	100	mV
Overshoot Settling Time	Maximum overshoot allowed duration. ***Note:*** At that time delta voltage should be lower than 5 mV from steady state voltage.	N/A	0.1	ms
Suggested Decoupling Capacitance	Capacitance range.	200	N/A	µF

12.3.5 VCC1P2 External Power Supply Specification (1.2 V)

Title	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark.	0.1	50	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)	18	10080	V/S
Operational Range	Voltage range for normal operating conditions.	1.14	1.26	V
Overshoot	Maximum overshoot allowed.	N/A	60	mV
Overshoot Duration	Maximum overshoot allowed duration. ***Note:*** At that time delta voltage should be lower than 5 mV from steady state voltage.	0.0	0.05	ms
Suggested Decoupling Capacitance	Capacitance range.	200	N/A	µF

12.3.6 VCC0P83 External Power Supply Specification (0.83 V)

Title	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark.	0.1	20	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)	13	7136	V/S
Operational Range	Voltage range for normal operating conditions.	0.788	0.871	V
Overshoot	Maximum overshoot allowed.	N/A	40	mV
Overshoot Duration	Maximum overshoot allowed duration. ***Note:*** At that time delta voltage should be lower than 5 mV from steady state voltage.	N/A	0.05	ms
Suggested Decoupling Capacitance	Capacitance range.	500	N/A	µF

12.3.7 Power On/Off Sequence

3.3 V should be powered on at 90% before any other rails.

The following power-up sequence is recommended for the X550.

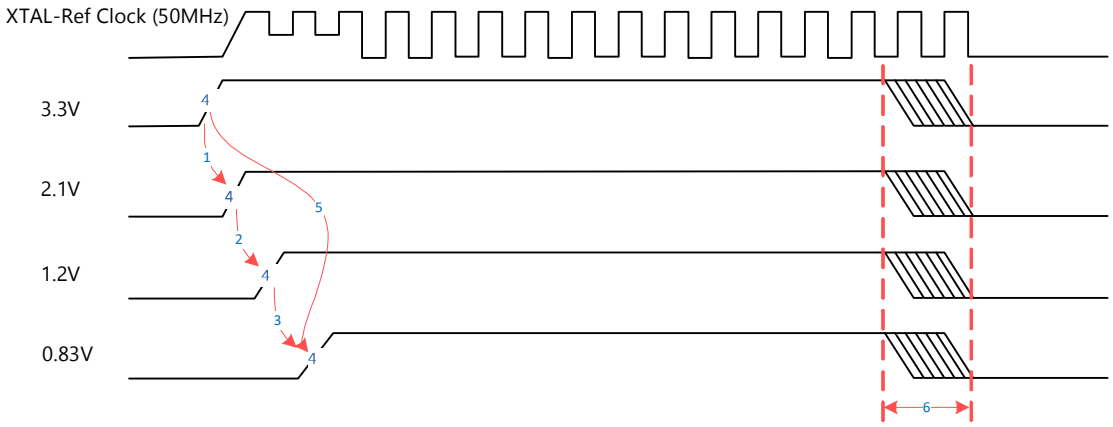


Figure 12-1. Power On/Off Sequence Flow

Table 12-1. Notes for Power On/Off Sequence Diagram

Note	Description
1	The 2.1 V rail does not start to ramp before the 3.3 V rail is 80% of its final value.
2	The 1.2 V rail does not start to ramp before the 2.1 V rail is 80% of its final value. Tmax(2.1 V to 1.2 V) should be less than 5 ms.
3	The 0.83 V rail does not start to ramp before the 1.2 V rail is 80% of its final value.
4	Tramp for each power rail. as defined in Section 12.3.3 through Section 12.3.6.
5	Total power-up time from the start of the 3.3 V rail raising until the 0.83 V rail gets to its final level is < 170 ms.
6	When powering off, all power rails must get to a 0 V level within 20 ms from the point the first power rail starts powering off.

12.3.8 Power On Reset

The X550 internal power-on reset circuitry initiates a full chip reset when voltage levels of power supplies reach certain thresholds at power-up.

Symbol	Parameter	Specification			Units
		Min	Typ	Max	
2.1 V	Threshold for 2.1 Vdc supply in power-up	1.7		1.8	Vdc
1.2 V	Threshold for 1.2 Vdc supply in power-up	0.931		0.989	Vdc
0.83 V	Threshold for 0.8 Vdc supply in power-up	0.563		0.597	Vdc
LAN_PWR_GOOD	High-threshold (VIH)	0.83		3.6	Vdc
LAN_PWR_GOOD	Low-threshold (VIL)	0.3		0.36	Vdc

12.3.9 Current Consumption

- Power numbers are based on measurement data.
- Maximum power mode: 3 sigma FAST material, Vnom, Tj-max (105 °C).
- Other operational modes: Typical material, Vnom.
- Unless stated otherwise, the power consumption below is based on a PCIe Gen3 x4 connection for the X550-AT and the X550-AT2, and PCIe Gen2 x8 for the X550-BT2.

Table 12-2. X550-AT Power Consumption

Operating Mode[1]	3.3 V		2.1 V		1.2 V		0.83 V		Device Total
	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Power (W)
10 GbE Max	194	640.2	623	1308.3	928	1113.6	6347	5268.01	8.33
10 GbE Max 30 m Reach	192	633.6	639	1341.9	968	1161.6	5180	4299.4	7.44
10 GbE Idle	193	636.9	622	1306.2	844	1012.8	5018	4164.94	7.12
10 GbE 3 m reach	192	633.6	604.31	1269.06	838.9	1006.68	4765.62	3955.46	6.86
5 GbE	192	633.6	598.86	1257.6	763.7	916.44	3511.2	2914.3	5.72
2.5 GbE	194	640.2	590.18	1239.37	808.31	969.98	2215.24	1838.65	4.69
1 GbE	192	633.6	622	1306.2	802	962.4	1500	1245	4.15
100 Mb/s	193	636.9	435	913.5	680	816	1294	1074.02	3.44
1 GbE WoL	192	633.6	616	1293.6	639	766.8	976	810.08	3.50
100 Mb/s WoL	191	630.3	436	915.6	531	637.2	767	636.61	2.82

1. Max defined as 105 °C Tj and 3 sigma FAST silicon. All others measured at 80 °C and typical silicon.

Table 12-3. X550-AT2 Power Consumption

Operating Mode[1]	3.3 V		2.1 V		1.2 V		0.83 V		Device Total
	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Power (W)
10 GbE Max (dual)	41	135.3	849.2	1783.32	1102	1322.4	9849	8174.67	11.4
10 GbE Max (dual) 30 m Reach	40	132	785	1648.5	1107	1328.4	7322	6077.26	9.2
10 GbE (dual idle)	46	151.8	860	1806	946	1135.2	6474	5373.42	8.5
10 GbE (single)	50	165	614	1289.4	793	951.6	4443	3687.69	6.1
10 GbE (dual) 30 m reach	45	148.5	809	1698.9	993	1191.6	5070	4208.1	7.2
10 GbE (dual) 3 m reach	41	135.3	786	1650.6	987	1184.4	4815	3996.45	7.0
5 GbE (dual)	46	151.8	828	1738.8	856	1027.2	4530	3759.9	6.7
2.5 GbE (dual)	46	151.8	816	1713.6	906	1087.2	2858	2372.14	5.3
1 GbE (dual)	45	148.5	855	1795.5	908	1089.6	1677	1391.91	4.4
1 GbE (single)	53	174.9	622	1306.2	802	962.4	1500	1245	3.7
100 Mb/s (dual)	46	151.8	477	1001.7	681	817.2	1292	1072.36	3.0
100 Mb/s (single)	53	174.9	435	913.5	680	816	1294	1074.02	3.0
1 GbE WoL (dual)	65	214.5	844	1772.4	758	909.6	1181	980.23	3.9
1 GbE WoL (single)	71	234.3	616	1293.6	639	766.8	976	810.08	3.1
100 Mb/s WoL (dual)	66	217.8	475	997.5	526	631.2	799	663.17	2.5
100 Mb/s WoL (single)	71	234.3	436	915.6	531	637.2	767	636.61	2.4

1. Max defined as 105 °C Tj and 3 sigma FAST silicon. All others measured at 80 °C and typical silicon.

Table 12-4. X550-BT2 Power Consumption

Operating Mode[1]	3.3 V		2.1 V		1.2 V		0.83 V		Device Total
	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Power (W)
10 GbE Max (dual)	104	343.2	860	1806	1113	1335.6	9817	8248.11	11.6
10 GbE Max (dual) 30 m Reach	77	254.1	797	1673.7	1209	1450.8	7113	5903.79	9.3
10 GbE (dual idle)	106	349.8	853	1791.3	994	1192.8	7079	5875.57	9.2
10 GbE (single)	84	277.2	628	1318.8	878	1053.6	4443	3687.69	6.3
10 GbE (dual) 30 m reach	79	260.7	806	1692.6	1064	1276.8	5025	4170.75	7.4
10 GbE (dual) 3 m reach	75	247.5	805	1690.5	1057	1268.4	4787	3973.21	7.2
5 GbE (dual)	81	267.3	844	1712.4	942	1130.4	4647	3857.01	7.0
2.5 GbE (dual)	80	264	820	1722	978	1173.6	2854	2368.82	5.5
1 GbE (dual)	75	247.5	863	1812.3	974	1168.8	1696	1407.68	4.6
1 GbE (single)	83	273.9	630	1323	866	1039.2	1513	1255.79	3.9
100 Mb/s (dual)	80	264	486	1020.6	753	903.6	1318	1093.94	3.3
100 Mb/s (single)	85	280.5	444	932.4	753	903.6	1331	1104.73	3.2
1 GbE WoL (dual)	95	313.5	851	1787.1	745	894	1069	887.27	3.9
1 GbE WoL (single)	102	336.6	630	1323	635	762	917	761.11	3.2
100 Mb/s WoL (dual)	103	339.9	484	1016.4	516	619.2	728	604.24	2.6
100 Mb/s WoL (single)	105	346.5	444	932.4	522	626.4	723	600.09	2.5

1. Max defined as 105 °C Tj and 3 sigma FAST silicon. All others measured at 80 °C and typical silicon.

12.3.10 Peak Current Consumption

12.3.10.1 Aux Power on Peak Current Consumption

Conditions:

- 3 sigma FAST material
- Vnom
- T_j = 80 °C
- 100 Mb/s link

3.3 V	2.1 V	1.2 V	0.83 V
400 mA	1.4 A	1.6 A	6 A

12.3.10.2 Main Power on Peak Current Consumption During 10GBASE-T Training

Conditions:

- 3 sigma FAST material
- Vnom
- T_j = 105 °C (max)
- 10 Gb/s link

3.3 V	2.1 V	1.2 V	0.83 V
400 mA	1.5 A	2 A	13 A

12.4 DC/AC Specifications

12.4.1 Digital Functional 3.3 V I/O DC Electrical Characteristics

Table 12-5. Digital I/O LAN_PWR_GOOD

Symbol	Parameter	Description	Min	Typ	Max	Units
VIH	Input High Voltage		0.84		3.60	V
VIL	Input Low Voltage		-0.30		0.36	V
VT	Threshold Point			0.60		V
Iil	Input Leakage Current at VI=3.3 V or 0 V					µA
IOZ	Tri-state Output Leakage Current at VO=3.3 V or 0 V					µA
PD	Internal Pull Down		50		70	KΩ
PU	Internal Pull Up		50		70	KΩ

Table 12-6. Digital I/O LED, SDP

Symbol	Parameter	Conditions	Min	Max	Units	Note
VOH	Output High Voltage	LED IOH = -16 mA SDP IOH = -12 mA VCC3P3 = minimum	2.4		V	
VOL	Output Low Voltage	LED IOL = 16 mA SDP IOL = 12 mA VCC3P3 = minimum		0.4	V	
VIH	Input High Voltage		2.0	3.45	V	
VIL	Input Low Voltage		-0.3	0.8	V	
Iil	Input Leakage Current			10	µA	
IOFF	Current at IDDQ Mode			10	µA	
PD	Internal Pull Down		50	70	KΩ	
Cin	Pin Capacitance			5	pF	
Cload	Pin Capacitance			5	pF	

Notes:

1. Table 12-6 applies to ENCRYPTION_EN, LED0_[3:0], LED1_[3:0], SDP0_[3:0], and SDP1_[3:0].
2. Digital I/O is 3.3 V input tolerance.

Table 12-7. Digital I/O Flash

Symbol	Parameter[1]	Conditions	Min	Max	Units	Note
VOH	Output High Voltage	IOH = -12 mA VCC3P3 = minimum	2.4		V	
VOL	Output Low Voltage	IOL = 12 mA VCC3P3 = minimum		0.7	V	
VIH	Input High Voltage		2.0	3.6	V	
VIL	Input Low Voltage		-0.3	0.7	V	
lil	Input Leakage Current			10	µA	
IOFF	Current at IDDQ Mode			10	µA	
PU	Internal Pull Up		50	70	KΩ	
Cin	Pin Capacitance			5	pF	
Cload	Pin Capacitance			5	pF	

Note: Table 12-7 applies to FLSH_SO, FLSH_SI, FLSH_CE_N, FLSH_SCK.

1. Parameters listed are 3.3 V tolerant.

Table 12-8. Digital I/O JTAG

Symbol	Parameter	Conditions	Min	Max	Units	Note
VOH	Output High Voltage	IOH = -12 mA VCC3P3 = minimum	2.4		V	
VOL	Output Low Voltage	IOL = 12 mA VCC3P3 = minimum		0.7	V	
VIH	Input High Voltage		2.0	3.6	V	
VIL	Input Low Voltage		-0.3	0.7	V	
lil	Input Leakage Current			10	µA	
IOFF	Current at IDDQ Mode			10	µA	
PU	Internal Pull Up		50	70	KΩ	
PD	Internal Pull Down		34	101	KΩ	
Cin	Pin Capacitance			5	pF	
Cload	Pin Capacitance			5	pF	

Note: Table 12-8 applies to TDO, TDI, TMS, TCK. TRST_N.

Table 12-9. Digital I/O Miscellaneous, PE_RST_N

Symbol	Parameter	Conditions	Min	Max	Units	Note
VOH	Output High Voltage	IOH = -12 mA VCC3P3 = minimum	2.4		V	
VOL	Output Low Voltage	IOL = 12 mA VCC3P3 = minimum		0.7	V	
VIH	Input High Voltage		2.0	3.6	V	
VIL	Input Low Voltage		-0.3	0.7	V	
Iil	Input Leakage Current			10	µA	
IOFF	Current at IDDQ Mode			10	µA	
PU	Internal Pull Up		50	70	KΩ	
Cin	Pin Capacitance			5	pF	
Cload	Pin Capacitance			5	pF	

Note: Table 12-9 applies to AUX_PWR, MAIN_PWR_OK, LAN1_DIS_N, LAN0_DIS_N, BYPASS_POR, ENCRYPTION_EN, and PE_RST_N.

12.4.2 Open Drain I/Os

Table 12-10. Open Drain I/Os

Symbol	Parameter	Condition	Min	Max	Units	Note
Vih	Input High Voltage		2.1		V	
Vil	Input Low Voltage			0.8	V	
Ileakage	Output Leakage Current	0 ≤ Vin ≤ VCC3P3 maximum		±10	µA	2
Vol	Output Low Voltage	@ Ipullup = 4 mA		0.4	V	5
Ipullup	Current Sinking	Vol = 0.4 V	4		mA	
Cin	Input Pin Capacitance			7	pF	3
Cload	Maximum Load Pin Capacitance			30	pF	4
Ioffsmb	Input leakage Current	VCC3P3 off or floating		±10	µA	2

Notes:
1. Section 12.4.2 applies to SMBD, SMBCLK, SMBALRT_N, PE_WAKE_N.
2. Device must meet this specification whether powered or un-powered.
3. Characterized, not tested.
4. Cload should be calculated according to the external pull-up resistor and the frequency.
5. OD no high output drive. VOL max=0.4 V at 16 mA, VOL max=0.2 V at 0.1 mA.

The buffer specification meets the SMBus specification requirements defined at: www.smbus.org.

12.4.3 NC-SI I/O DC Specification

Table 12-11. NC-SI I/O DC Specification

Symbol	Parameter	Conditions	Min.	Typ.	Max	Units
Vref[1]	Bus High Reference		3.0	3.3	3.46	V
Vabs	Signal Voltage Range		-0.3		3.765	V
ViL	Input Low Voltage				0.8	V
ViH	Input High Voltage		2.0			V
VoL	Output Low Voltage	IoL = 4 mA Vref= $Vref_{min}$	0		0.4	V
VoH	Output High Voltage	IoL = -4 mA Vref= $Vref_{min}$	2.4		Vref	V
IiH	Input High Current	Vin = 3.6 V Vref = 3.6 V	0		200	µA
IiL	Input Low Current	Vin = 0 V $Vref_{min}$ to $Vref_{max}$	-20		0	µA
Vckm	Clock Midpoint Reference Level				1.4	V
Iz	Leakage Current for Output Signals in High-Impedance State	$0 \leq Vin \leq Vih_{max}$ @Vref = $Vref_{max}$	-20		20	µA

Notes:

1. Vref = Bus high reference level. This parameter replaces the term supply voltage since actual devices can have internal mechanisms that determine the operating reference for the sideband interface that are different from the device's 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. To facilitate system implementations, a device must provide a mechanism (such as a power supply pin, internal programmable reference, or reference level pin) to enable 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 specification define the Vrefmax up to 3.6 V, the X550 supports the Vrefmax up to 3.46 V (3.3 V +5%).
2. Section 12.4.3 applies to NCSI_CLK_IN, NCSI_CRS_DV, NCSI_RXD[1:0], NCSI_TX_EN and NCSI_TXD[1:0], NCSI_ARB_IN, NCSI_ARB_OUT.
3. Refer to the *Network Controller Sideband Interface (NC-SI) Specification for more details*.

12.4.4 Digital I/F AC Specifications

12.4.4.1 Digital I/O AC Electrical and Timing Characteristics

Table 12-12. Digital 3.3 V I/O AC Specifications — SDP

Parameters	Description	Min	Max	Condition	Notes
f_{symbol}	Symbol Rate		25 MS/s		

12.4.4.2 Digital 3.3 V I/O AC Specifications — JTAG AC Specifications

Table 12-13. JTAG I/F Timing Parameters

Symbol	Parameter	Min	Typ	Max	Unit
t_{jclk}	JTCK clock frequency			10	MHz
t_{jh}	JTMS and JTDI hold time	10			ns
t_{jsu}	JTMS and JTDI setup time	10			ns
t_{jpr}	JTDO propagation delay			15	ns

Notes:
1. Table 12-13 applies to JTCK, JTMS, JTDI and JTDO.
2. Timing measured relative to JTCK reference voltage of VCC3P3/2.

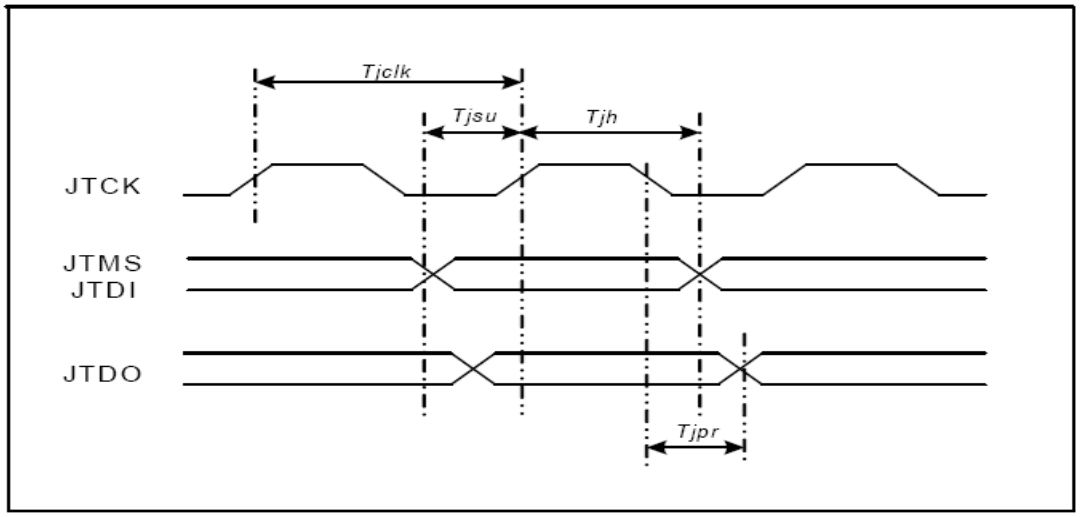


Figure 12-2. JTAG AC Timing Diagram

12.4.4.3 SMBus AC Specifications

The X550 meets the SMBus AC specification of 100 KHz as defined in SMBus specification, Version 2, Section 3.1.1 (http://www.smbus.org/specs/).

The X550 also supports a 400 KHz SMBus (as an input clock from the MC and as a slave). The X550 meets the 100 KHz, 400 KHz, and 100 MHz specifications listed in Table 12-14.

Table 12-14. SMBus Timing Parameters (Master Mode)

Symbol	Parameter	Min	Typ 100 KHz[1]	Typ 400 KHz[2]	Typ 1 MHz[3,4]	Max	Units
F_{SMB}	SMBus Frequency		100	400	1000	1000	KHz
T_{BUF}	Time between STOP and START condition driven by the X550	0.5	4.7	1.3	0.5		µs
$T_{HD:STA}$	Hold time after Start Condition. After this period, the first clock is generated.	0.26	4	0.6	0.26		µs
$T_{SU:STA}$	Start Condition setup time	0.14	2	0.3	0.14		µs
$T_{SU:STO}$	Stop Condition setup time	0.26	4	0.6	0.26		µs
$T_{HD:DAT}$	Data hold time	0	0.3	0	0		µs
$T_{SU:DAT}$	Data setup time	0.05	0.25	0.1	0.05		µs
$T_{TIMEOUT}$	Detect SMBClk low timeout	35	35	35	35	35	ms
T_{LOW}	SMBClk low time	0.5	4.7	1.3	0.5		µs
T_{HIGH}	SMBClk high time	0.26	4	0.6	0.26		µs

1. Specifications based on SMBus specification.
2. Specifications based on I^2C specification for Fast-mode (400 KHz).
3. Specifications based on I^2C specification rev 03 for Fast-mode Plus (1 MHz).
4. 1 MHz speed may vary and is impacted by the topology.

Table 12-15. SMBus Timing Parameters (Slave Mode)

Symbol	Parameter	Min 100 KHz[1]	Min 400 KHz[2]	Min 1 MHz[3]	Max	Units
F_{SMB}	SMBus Frequency	10	10	10	1000	KHz
T_{BUF}	Time Between STOP and START	4.7	1.3	0.5		µs
$T_{HD,STA}$	Hold Time After Start Condition. After This Period, the First Clock is Generated.	4	0.6	0.26		µs
$T_{SU,STA}$	Start Condition Setup Time	4.7	0.6	0.26		µs
$T_{SU,STO}$	Stop Condition Setup Time	4	0.6	0.26		µs
$T_{HD,DAT}$	Data Hold Time	300	100	0		µs
T_{LOW}	SMBCLK Low Time	250	100	50		µs
T_{HIGH}	SMBCLK High Time	4.7	1.3	0.5		µs

1. Specifications based on SMBus specification.
2. Specifications based on I^2C specification for Fast-mode (400 KHz).
3. Specifications based on I^2C specification rev 03 for Fast-mode Plus (1 MHz).

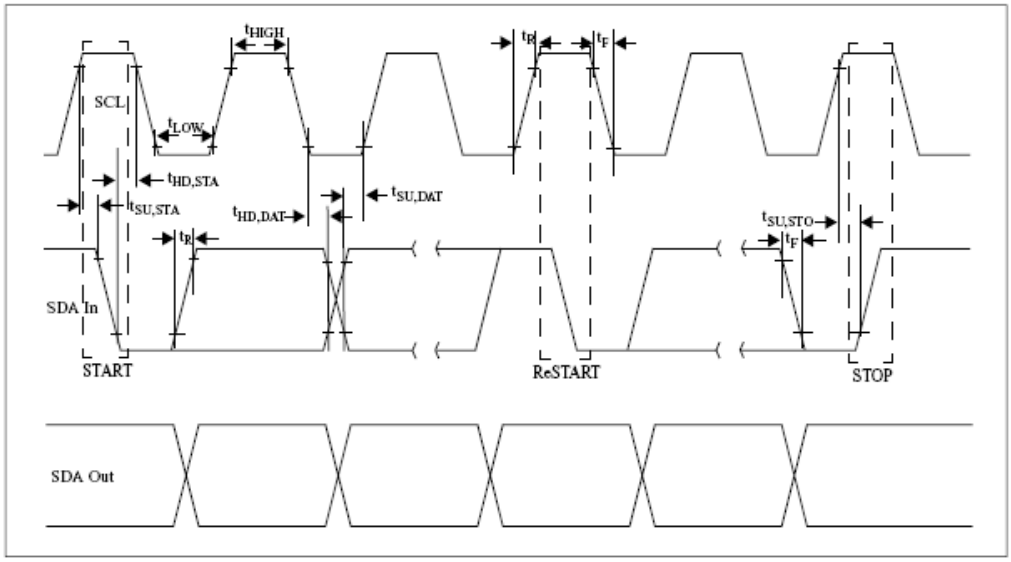


Figure 12-3. SMBus I/F Timing Diagram

12.4.4.4 Flash AC Specification

The X550 is designed to support a serial Flash.For Flash I/F timing specifications, see Table 12-16 and Figure 12-4.

Table 12-16. Flash I/F Timing Parameters from Supported Flash Devices

Symbol	Parameter	Min	Max	Units
fSCK	Serial Clock (SCK) Frequency		25	MHz
fRDLF	SCK Frequency for Read Array (Low Frequency - 0x03 Opcode)		25	MHz
tSCKH	SCK High Time	16		ns
tSCKL	SCK Low Time	16		ns
tSCKR(1)	SCK Rise Time - Peak-to-Peak (Slew Rate)	0.1		V/ns
SCKF(1)	SCK Fall Time - Peak-to-Peak (Slew Rate)	0.1		V/ns
tCSH	Chip Select High Time	25		ns
tCSLS	Chip Select Low Setup Time (Relative to SCK)	10		ns
tCSLH	Chip Select Low Hold Time (Relative to SCK)	10		ns
tCSHS	Chip Select High Setup Time (Relative to SCK)	10		ns
tCSHH	Chip Select High Hold Time (Relative to SCK)	10		ns
tDS	Data In Setup Time	5		ns
tDH	Data In Hold Time	5		ns
tDIS(1)	Output Disable Time		15	ns
tV(2)	Output Valid Time		15	ns
tOH	Output Hold Time	1		ns

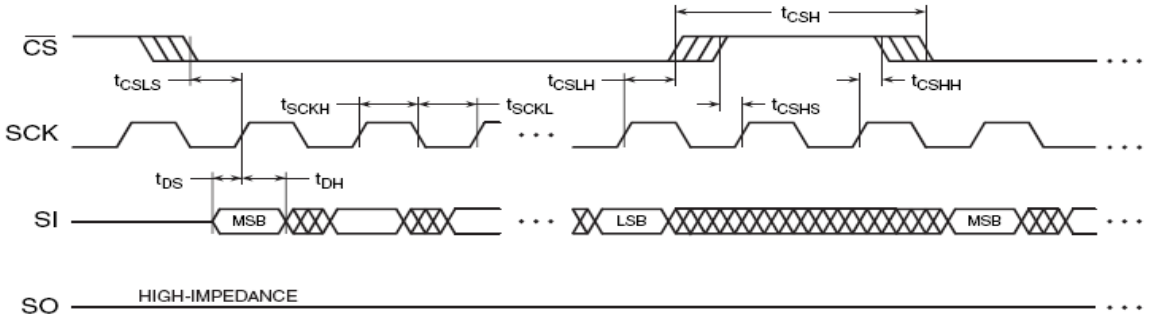


Figure 12-4. Serial Input Timing

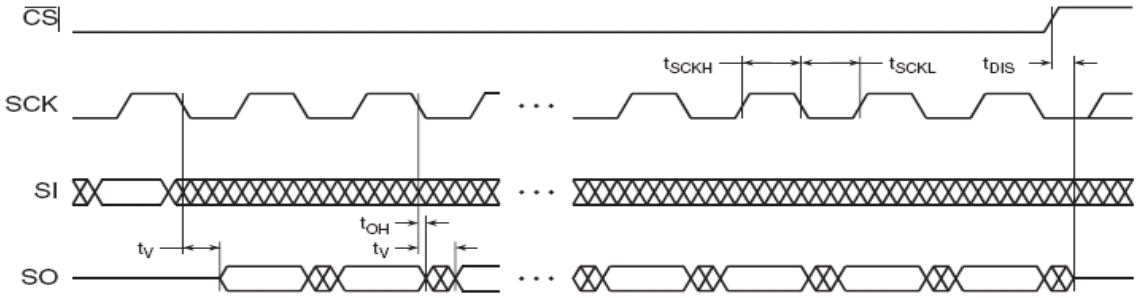


Figure 12-5. Serial Output Timing

12.4.4.5 NC-SI AC Specifications

The X550 supports the NC-SI standard as defined in the Network Controller Sideband Interface (NC-SI) specification. The NC-SI timing specifications can be found in Table 12-17 and Figure 12-6.

Table 12-17. 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[1] (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, ARB_IN, ARB_OUT Data Setup to REF_CLK Rising Edge	Tsu		3			ns	3
TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER, ARB_IN, ARB_OUT 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	

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; middle point (50% of Vref in Section 12.4.3). 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; middle point (50% of Vref in Section 12.4.3). 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 Figure 12-6). The middle points (50% of Vref) are marked as Vckm and Vm for clock and data, respectively.

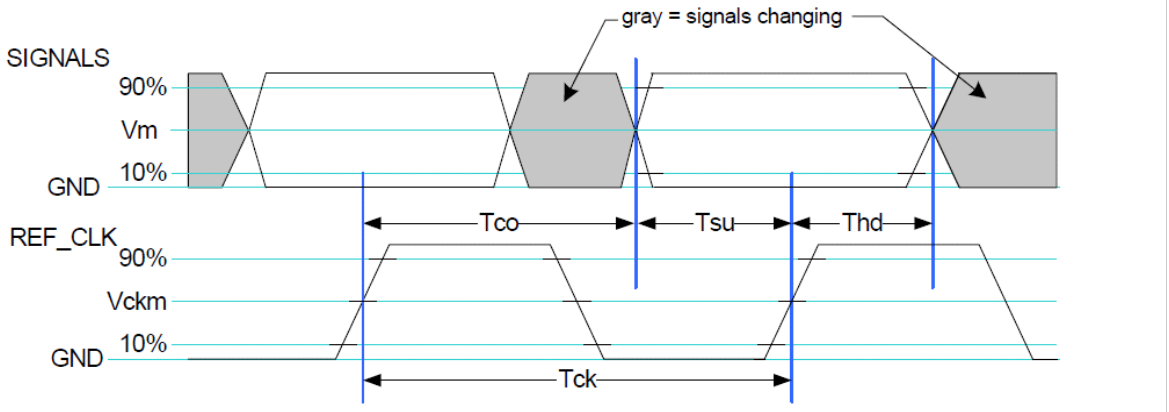


Figure 12-6. NC-SI AC Specification

12.4.4.6 Reset Signals

For power-on indication, the X550 can either use an internal power-on circuit, which monitors the 2.1 V, 1.2 V, and 0.83 V power supply, or external reset using the LAN_PWR_GOOD pad.

Table 12-18. POR Configuration

Parameter	LAN_PWR_GOOD
Internal POR	High
External POR	Reset Signal

The timing between the power-up sequence and the different reset signals when using the internal power indication is described in Section 12.3.8.

12.4.4.6.1 External Power-On Reset

The X550 may use the LAN_PWR_GOOD pad as power-on indication.Table 12-19 lists the timing for the external power-on signal.

Table 12-19. External Reset Specification

Symbol	Title	Description	Min	Max	Units
Tlpgw	LAN_PWR_GOOD Minimum Width	Minimum width for LAN_PWR_GOOD.	5	N/A	µs
Tlpg	LAN_PWR_GOOD High Hold	Hold time following power-up (power supplies in acceptable operating range).	40	80	ms

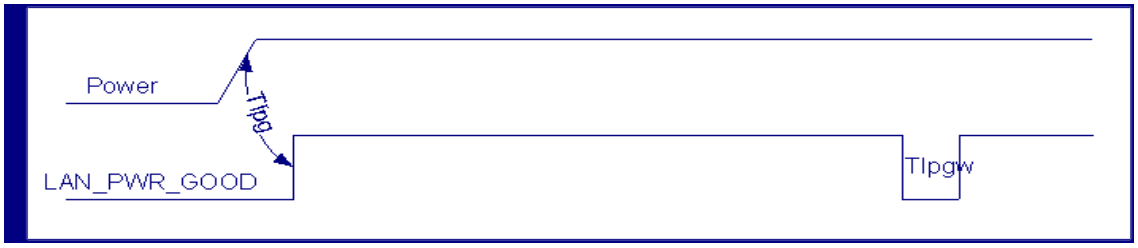


Figure 12-7. External Reset Timing Diagram

12.4.5 PCIe Interface AC/DC Specification

The X550 PCIe interface supports the electrical specifications defined in:

- PCI Express* 2.0 Card Electromechanical Specification.
- PCI Express* 3.0 Base Specification, Chapter 4.

12.4.6 Network Interface AC/DC Specification

The AC/DC specification of the network interface is according to the 10GBASE-T, 1000BASE-T, and 100BASE-TX Standards (802.3, 802.3an, and 802.3u, respectively), and the and NBASE-T specification. The 100BASE-T parameters are also described in standard ANSI X3.263.

12.5 Thermal Diode

The X550 incorporates an on-die diode that can be used to monitor the die temperature (junction temperature). An external thermal sensor located on the motherboard or a stand-alone measurement device can also be used to monitor the die temperature of the X550 for thermal management or characterization. Table 12-20 lists the parameters that pertain to the X550's thermal diode pins (THERM_D1_P, THERM_D1_N).

Table 12-20. Thermal Diode Parameters

Symbol	Min.	Typical	Max	Unit	Notes
I forward bias	50	100	1000	µA	1
n_ideality	1.03	1.05	1.08		3
ESR		2.77		Ω	2

Notes:

1. Intel does not support or recommend operation of the thermal diode under reverse bias.
2. ESR: Effective Series Resistance - needed for various TD measurement tools.
3. n_ideality is the diode ideality factor parameter, as represented by the diode equation:

$$I = I_0\left(e^{\frac{eV_D}{nkT}} - 1\right)$$

Note: The X550's thermal diode has a -10 °C offset from the Tj at the center of the die. An external thermal sensor should add this offset for accurate measurement.

12.6 Crystal Specification

Parameter Name	Symbol	Recommended Value	Conditions
Frequency	f_o	50.000 MHz	@25 °C
Vibration Mode		Fundamental	
Cut		AT	
Operating /Calibration Mode		Parallel	
Frequency Tolerance @25 °C	Df/f_o @25 °C	±10 ppm / ±15 ppm (see note)	@25 °C
Temperature Tolerance	Df/f_o	±10 ppm ±15 ppm (see note)	
Operating Temperature	T_{opr}	0 to +70 °C	
Non Operating Temperature Range	T_{opr}	-40 to +90 °C	
Equivalent Series Resistance (ESR)	R_s	50 Ω maximum	@50 MHz
Load Capacitance	C_{load}	16 pF	
Shunt Capacitance	C_o	<5 pF maximum	
Max Drive Level	D_L	300 µW	
Insulation Resistance	IR	500 MΩ minimum	@ 100 Vdc
Aging	Df/f_o	(±5 ppm/year) / (±2 ppm/year) (see note)	
CL1/CL2		18 pF	

Note: To provide some flexibility in the Bill Of Materials (BOM), two options are supported:

1. Crystal parameters option 1:
   a. Frequency tolerance @ 25 °C ±15 ppm
   b. Temperature tolerance ±15 ppm
   c. Aging ±2 ppm/year
2. Crystal parameters option 2:
   a. Frequency tolerance @ 25 °C ±10 ppm
   b. Temperature tolerance ±10 ppm
   c. Aging ±5 ppm/year

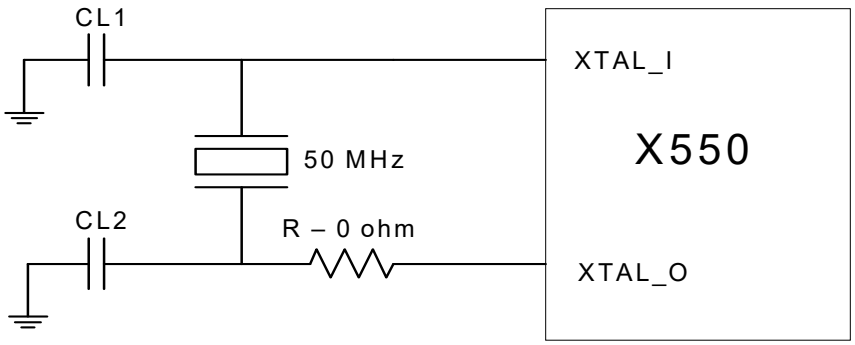


Figure 12-8. Crystal Specification Schematic

12.7 Package

12.7.1 Mechanical

The X550 is assembled into a 25x25 mm X540 layout-compatible FCBG package with a 10-layer substrate, or to a 17x17 mm package with a 6-layer substrate.

Body Size	Ball Count	Ball Pitch	Substrate
25x25 mm	576	1.0 mm	10 layers
17x17 mm	256	1.0 mm	6 layers

12.7.2 Thermal

For the X550's package thermals, refer to Chapter 14, “Thermal Design Recommendations”.

12.7.3 Electrical

Package electrical models are part of the IBIS files.

12.7.4 Mechanical Package Diagram

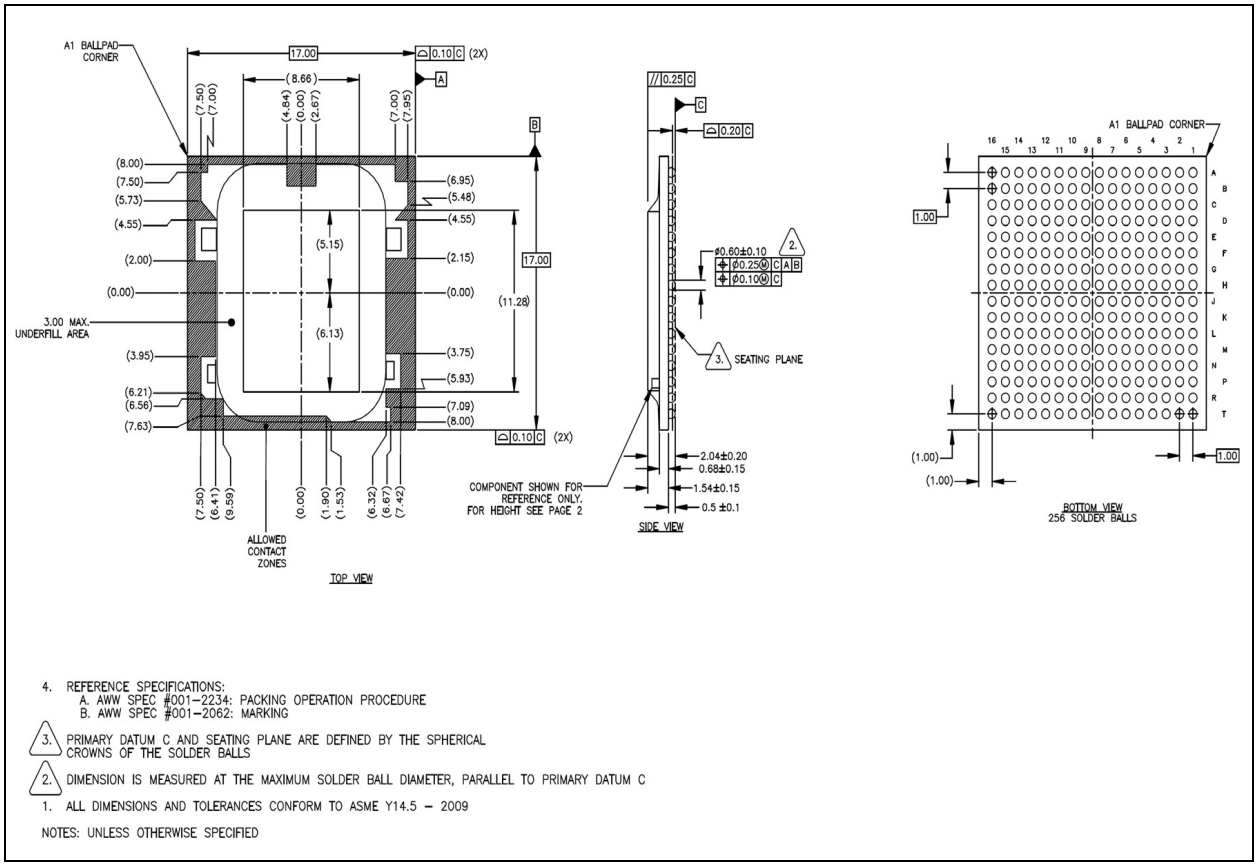


Figure 12-9. X550-AT2

Table 12-21. X550-AT2 - Discrete Components

Item No.	X	Y	Z (Height)	Component Type
CO1	-6.70	-5.00	0.56	0402
CO2	6.62	-4.80	0.56	0402
CO3	-6.90	3.515	0.90	0603
CO4	6.90	3.515	0.90	0603

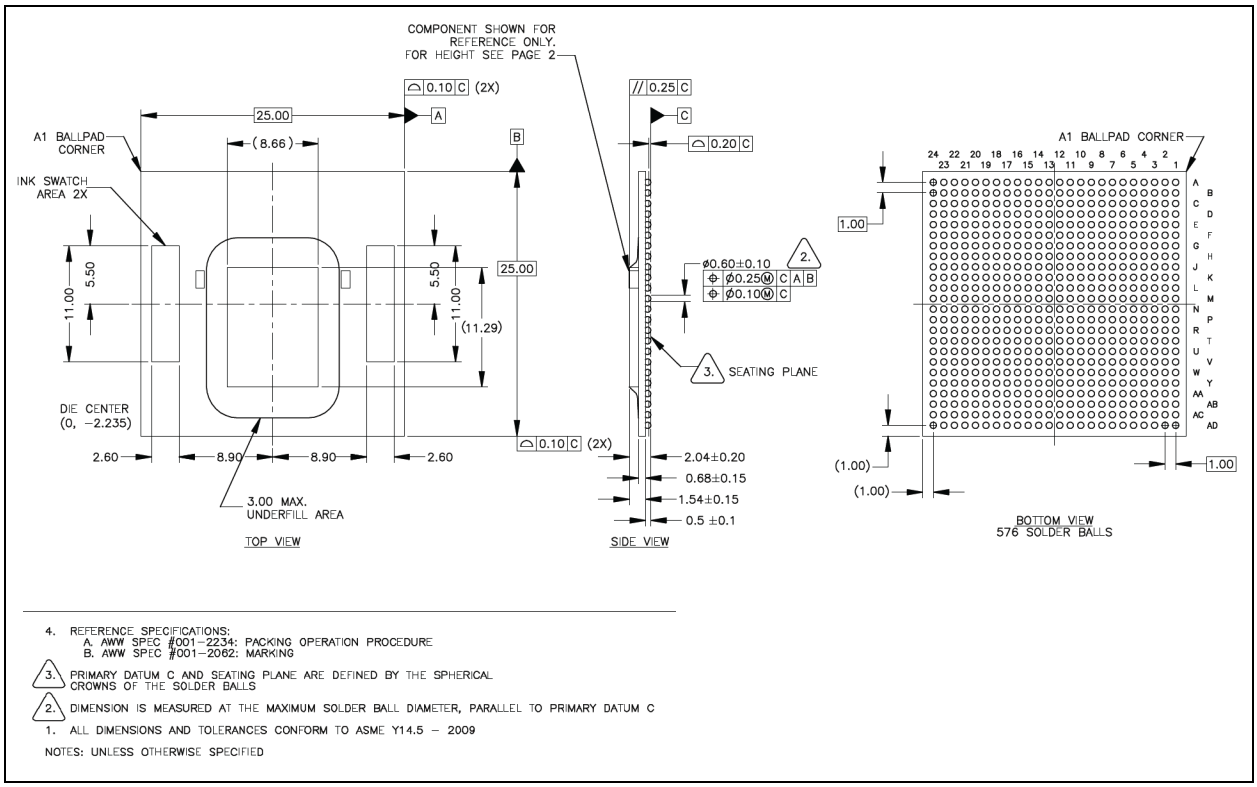


Figure 12-10. X550-BT2

Table 12-22. X550-BT2 - Discrete Components

Item No.	X	Y	Z (Height)	Component Type
CO1	-6.90	2.485	0.90	0603
CO2	6.90	2.485	0.90	0603

12.8 Devices Supported

12.8.1 Flash

The X550 supports the following 3.3 V Flash devices with an SPI interface.

To provide some flexibility in the Bill of Materials (BOM), a few options are supported:

Flash Size	Manufacturer	Part Number
4 MB	EON Silicon	EN25QH32
	ESMT	F25L32PA / F25L32QA
	Fidelix	FM25Q32A
	Macronix	MX25L3233F
	Micron	N25Q032A
	Spansion	S25FL132K
	Winbond	W25Q32FV
8 MB	Adesto Tech	AT25SF161-SHD-T
	Macronix	MX25V1635FM2I
	Winbond	W25Q16JVSNIQ
16 MB	Adesto Tech	AT25SF321-SHD-T
	Macronix	MX25L3233FM2I-08G
	Winbond	W25Q32JVSSIQ
32 MB	Adesto Tech	AT25SF641-SUT-T
	Macronix	MX25L6433FM2I-08G
	Winbond	W25Q64JVSSIQ

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

Chapter 13 Design Considerations and Guidelines

This section provides guidelines for selecting components, connecting interfaces, using special pins, and layout guidance.

Unused interfaces should be terminated with pull-up or pull-down resistors. These are indicated in Section 2 or reference schematics. There are reserved pins, identified as RSVD_NC and RSVD_VSS. The X550 might enter special test modes unless these strapping resistors are in place.

Some unused interfaces must be left open. Do not attach pull-up or pull-down resistors to any balls identified as No Connect or Reserved No Connect.

13.1 Connecting the PCIe Interface

The X550 connects to the host system using a PCIe interface. The interface can be configured to operate in several link modes as detailed in Section 3. A link between the ports of two devices is a collection of lanes. Each lane 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). Each end of the link is terminated on the die into nominal 100 differential DC impedance. Board termination is not required.

Refer to the *PCI Express* Base Specification, Revision 3.0* and *PCI Express* Card Electromechanical Specification, Revision 2.0.*

13.1.1 Link Width Configuration

The X550 supports link widths of x8, x4, or x1 as determined by the Flash *Lane_Width* field in the PCIe initialization configuration. The configuration is loaded using bits 9:4 in the *Max Link Width* field of the Link Capabilities register (0xAC); default setting is 0x08. The X550-AT2 default link width is x4. The X550-BT2 default link width is x8.

During link configuration, the platform and the X550 negotiate a common link width. For proper operation, the selected maximum number of PCIe lanes must be connected to the host system.

13.1.2 Polarity Inversion and Lane Reversal

To ease routing, designers have the flexibility to the lane reversal modes supported by the X550. Polarity inversion can also be used, since the polarity of each differential pair is detected during the link training sequence.

When lane reversal is used, some of the down-shift options are not available. For a description of available combinations, see Section 3.

13.1.3 PCIe Reference Clock

The X550 requires a 100 MHz differential reference clock called PE_CLK_p and PE_CLK_n. These signals are typically generated on the system board and routed to the PCIe connector. For add-in cards, the clock is furnished at the PCIe connector.

Note: The frequency tolerance for the PCIe reference clock is +/- 300 ppm.

13.1.4 Bias Resistor

For proper biasing of the PCIe analog interface, a 4.75 KΩ 0.1% resistor needs to be connected between PE_RSENSE and PE_RBIAS. To avoid noise coupled onto this reference signal, place the bias resistor close to the X550 and keep traces as short as possible.

13.1.5 Miscellaneous PCIe Signals

The X550 signals power management events to the system by pulling low the PE_WAKE_N signal. This signal operates like the PCI PME# signal. Note that somewhere in the system, this signal has to be pulled high to the auxiliary 3.3 V supply rail.

The PE_RST# signal, which serves as the familiar reset function for the X550, needs to be connected to the host system's corresponding signal.

13.1.6 PCIe Layout Recommendations

For information regarding the PCIe signal routing, refer to the *Intel PCIe Design Guide*.

13.2 Connecting the 10GBASE-T MDI Interfaces

In 10GBASE-T mode, the line interface on the X550 is capable of driving up to 100 meters of CAT-6a unshielded twisted pair or 100 meters of CAT-7 shield cable (100 Ω differential impedance). It can also drive 55 meters of CAT-6 cable. In 100BASE-TX and 100MBASE-TX modes, it can drive 130 meters of CAT-5e (or better) cable. It is designed to drive this via a quad, 50 Ω center tapped 1:1 transformer connected to an RJ-45 PCB-mount jack. Solutions that combine the transformer and RJ-45 jack into a single device are also supported.

The line interface on the X550 supports automatic A/B and C/D pair swaps and inversions (MDI-X). It also supports provisioned ABCD to DCBA pair reversal for ease of routing via an NVM setting, which sets this configuration at power up.

Note: This reversal does not swap polarities thus A+ maps to D+, etc.

13.2.1 MDI Circuit Guidelines

The MDI discrete design and integrated magnetic components were chosen for inclusion in the reference design and Bill of Material (BOM). Refer to Section 13.13 for more details. These components are capable of delivering the performance required for this demanding application.

13.2.2 Magnetics Module

The magnetics module has a critical effect on overall IEEE and emissions conformance. The X550 should meet the performance required for a design with reasonable margin to allow for manufacturing variation. Carefully qualifying new magnetics modules prevents problems that might arise because of interactions with other components or the Printed Circuit Board (PCB) itself. The magnetics specified should comply with the specifications listed in the Intel® 10-GBASE-T Magnetic Specification Electrical/ Mechanical Requirements for 10GBASE-T Magnetic Components, which includes separate specifications for discrete and integrated magnetics modules.

These have five channels of 3-wire choke / transformer pairs in them, with four facing the choke towards the line (RJ45), and one facing the choke towards the X550.

The steps involved in magnetics module qualification are:

1. Verify that the vendor's published specifications in the component datasheet meet or exceed the required IEEE 802.3an specifications and the internal specifications listed in the Intel® 10-GBASE-T Magnetic Specification Electrical/Mechanical Requirements for 10GBASE-T Magnetic Components.
2. Independently measure the component's electrical parameters on a test bench, checking samples from multiple lots. Check that the measured behavior is consistent from sample-to-sample and that measurements meet the published specifications.
3. Perform physical layer conformance testing and EMC (FCC and EN) testing in real systems. Vary temperature and voltage while performing system level tests.

Magnetics modules for 10GBASE-T Ethernet as used by the X550 are similar to those designed for 1GBASE-T, except that the electrical requirements for the board layout and magnetics are more stringent. Refer to Section 12 for specific electrical requirements that the magnetics need to meet.

13.2.3 5th Channel

To sense and cancel common-mode noise, the X550 provides a 5th channel designed for this purpose. It is very similar to the receivers on Pairs A - D, with the exception that it does not have a driver and only receives. Its input impedance is 100 Ω.

When discrete magnetics are in use, this channel should be connected to a common-mode sense point. In this design, the common-mode sense point is the Bob Smith termination of Pair D. This sense point is run through a transformer to convert the signal to differential for pick-up by the 5th channel receiver. To match the 100 Ω impedance of the receiver input with the required 75 Ω Bob Smith termination, a 300 Ω parallel resistor is used.

Integrated magnetics perform the required termination internally. As a result, the Bob Smith termination used in the discrete case is not necessary.

If a different choice of magnetics is required they should comply with the specifications listed in Section 13.13.

13.2.4 Board Noise Cancellation

An advantage of the 5th channel is that in addition to canceling received common-mode interference from the line, it also cancels noise picked up on the PCB. This is especially important if the RJ-45 is located any significant distance from the X550. Consequently, every effort should be made to route the 5th channel traces along the same path as the other four MDI traces.

13.2.5 MDI Layout Guidance

The MDI that was chosen for X550 designs consist of an RJ-45 right-angle PCB jack, Bob Smith termination and discrete magnetics (see Figure 13-1). The magnetics can be implemented either as a discrete module or an integrated solution combining the magnetics and the RJ-45 jack.

Minimizing the amount of space needed for the PHY is important because other interfaces compete for physical space on a Network Interface Card (NIC) or LAN on Motherboard (LOM) near the connector. The PHY circuits need to be as close as possible to the connector.

Figure 13-1 illustrates some basic placement distance guidelines. It shows four differential pairs and the layout can be generalized for a 10 GbE system with four analog pairs. The ideal placement for the X550 is approximately two inches behind the magnetics module for both the discrete and integrated solutions.

13.2.5.1 Discrete Magnetics

The X550 uses a common-mode sensing circuit on the MDI of each channel to cancel in-band common-mode interference that couples into the differential receive signals. This common-mode sense circuitry is referred to as the 5th channel. This is where the sense circuitry sits in series with the Bob Smith termination for Channel D. The signal runs through a transformer to perform the common-to-differential conversion where it is followed by the X550-facing common-mode choke. This provides isolation from board noise being radiated on Channel D. In addition, the impedance of the Bob Smith termination is 75 Ω Channel A, B, C and 300 Ω Channel D, whereas the PHY input impedance is 100 Ω.

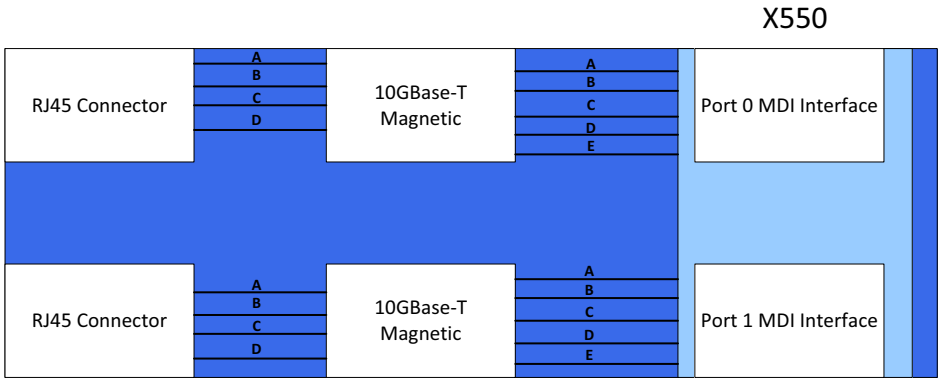


Figure 13-1. Basic MDI Placement Guidelines

The X550, referred as a LAN silicon in Figure 13-2 and Figure 13-3, must be at least 1.5 inches from the I/O back panel. To help reduce EMI, the following recommendations should be followed:

- Minimize the length of the MDI interface.
- Place the MDI traces no closer than 0.5 inch (1.3 cm) from the board edge.

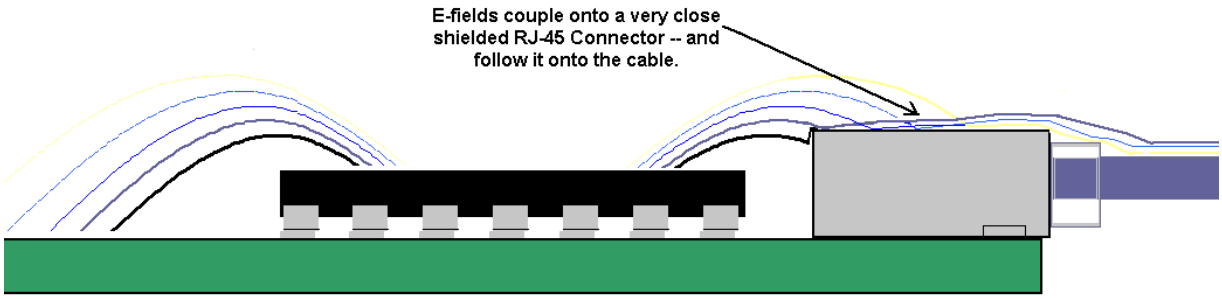


Figure 13-2. Effect of Device Placed Less Than One Inch from the RJ-45 Connector

The associated grounding scheme (blue) with the front-end of the chip is shown in the microstrip traces between the transformer and the RJ-45 (see Figure 13-3).

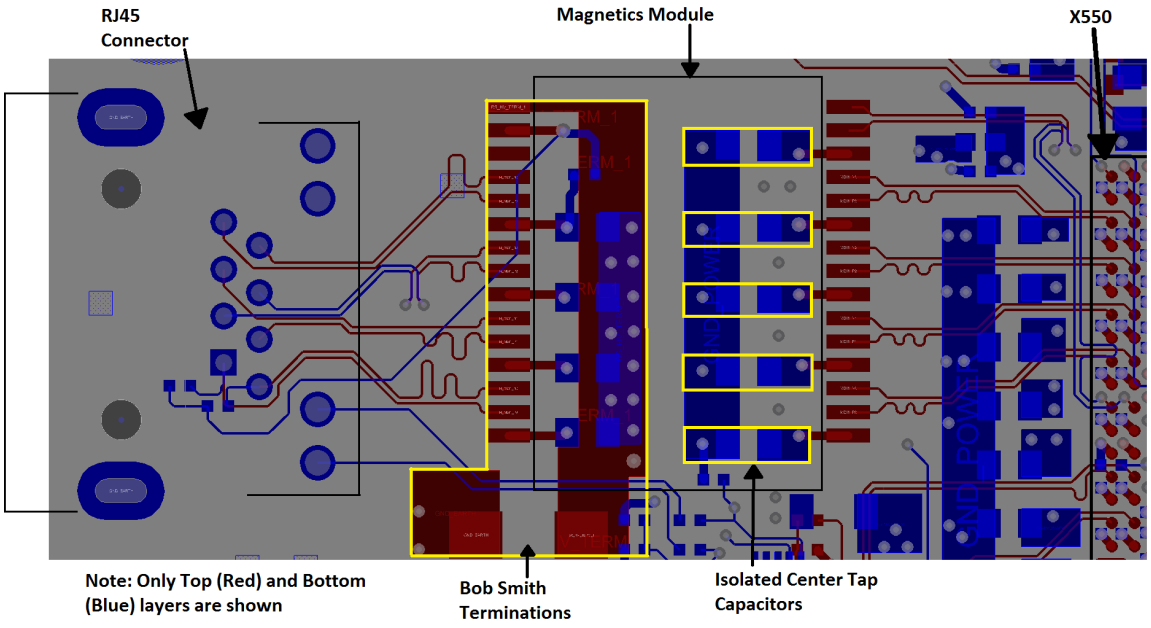


Figure 13-3. Transformer Bypass Components

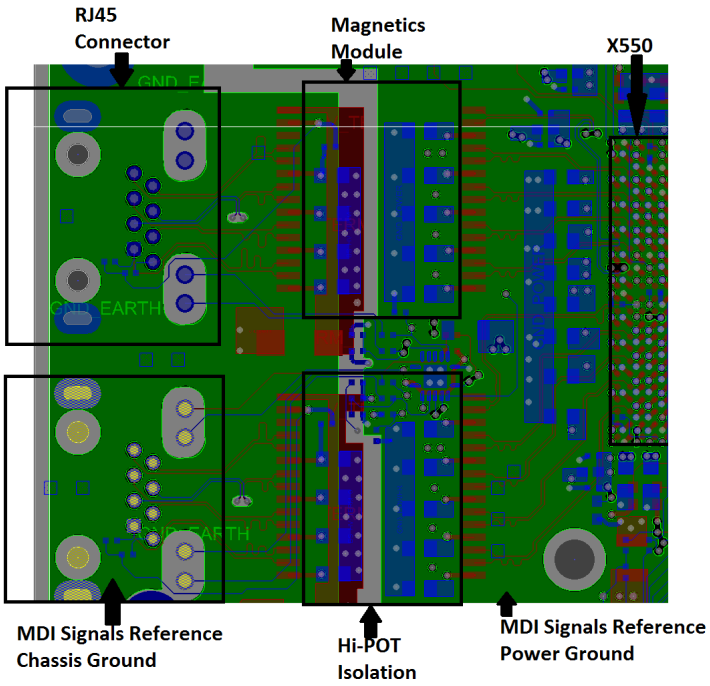


Figure 13-4. MDI Grounds

From these figures (Figure 13-3 and Figure 13-4), the following should be noted:

- The ground is split underneath the magnetics, with the chassis ground being present under the front-end (Figure 13-3), and the circuit ground under the X550 side of the magnetics. Thus, the MDI traces on the line side are referenced to chassis ground.
- The magnetic module is used with the common mode choke facing the X550. This enables a Bob Smith termination to be implemented from the line side transformer center taps to chassis ground.
- Bob Smith termination and the shield of the RJ-45 jack are both connected to chassis ground, which are the two large circular metalizations in the MDI (see Figure 13-4).
- The resistors in the Bob Smith termination are required to be 0805s to handle the cable discharge voltages.
- Similarly, the magnetics should be placed as close as possible to the RJ-45 jack.
- In Figure 13-3, the Hi-POT clearance for cable discharge is shown and designed with ≥80 mils of clearance from the ground. Note that the breakdown voltage in FR-4 is lowest in the x-y axes planar to the PCB and highest in the z-axis between layers.

13.2.5.2 Integrated Magnetics

The 5th MDI channel previously described in discrete magnetics implementation is still realized when using integrated magnetics, with some notable exceptions (see Figure 13-5). The required termination, reference ground-plane division, and 5th channel wiring to the 4th channel center-tap occur within the integrated magnetics module. The guidance on MDI trace implementation between the X550 and the integrated module remains the same as with discrete magnetics, including length requirements and target impedance.

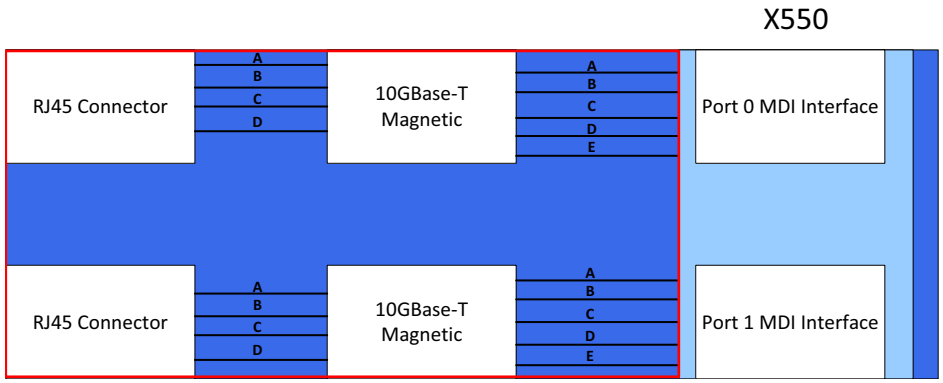


Figure 13-5. Basic MDI Placement Guidelines

Note: The RJ-45 jack and 10GBase-T Magnetics (red outline) are included in the integrated magnetics module.

The differences between the discrete and integrated magnetics implementations are as follows:

- The ground split that occurs underneath discrete magnetics modules is performed internally when using integrated magnetics. As a result, this ground split does not need to be addressed.
- The required Bob Smith termination is implemented internally when using integrated magnetics.

13.2.5.3 MDI Differential Pair Trace Routing for LAN Design

Trace routing considerations are important to minimize the effects of crosstalk and propagation delays on sections of the board where high-speed signals exist. Signal traces should be kept as short as possible to decrease interference from other signals, including those propagated through power and ground planes.

13.2.5.4 Signal Trace Geometry

One of the key factors in controlling trace EMI radiation are the trace length and the ratio of trace-width to trace-height above the reference plane. To minimize trace inductance, high-speed signals and signal layers that are close to a reference or power plane should be as short and wide as practical. Ideally, the trace-width to trace-height above the ground plane ratio is between 1:1 and 3:1. To maintain trace impedance, the width of the trace should be modified when changing from one board layer to another if the two layers are not equidistant from the neighboring planes.

Each pair of signals should target a differential impedance of 100 Ω. ±15%.

A set of trace length calculation tools are made available from Intel to aid with MDI topology design. Contact your Intel representative for tool availability.

When designing a board layout, the automatic router feature of the CAD tool must not route the differential pairs without intervention. In most cases, the differential pairs require manual routing.

Note: Measuring trace impedance for layout designs targeting 100 Ω often results in lower actual impedance due to over-etching. Designers should verify actual trace impedance and adjust the layout accordingly. If the actual impedance is consistently low, a target of 105 Ω to 110 Ω should compensate for over-etching.

13.2.5.4.1 Matching Traces within a Pair (P and N)

P and N for each MDI pair should be matched to within 5 mils on the PCB to prevent common-to-differential and differential-to-common conversion due to the length mismatch.

If in-pair length matching is not possible using bends or small loops, serpentine routing (zig-zag of a shorter trace) is acceptable if only one to three meanders are routed within 200 mils of the source of the skew or the end(s) of any otherwise unmatched lengths of a differential trace segment. For example, near device pins and/or at or near the connector pins and possibly at differential signal vias. Refer to the Intel® Ethernet Controller X550 Checklists for more details.

Table 13-1. MDI Routing Summary

Parameter	Main Route Guidelines	Breakout Guidelines[1]	Notes
Signal group	MDI_P[3:0] MDI_N[3:0]		
Microstrip*/Stripline* uncoupled single-ended impedance specification	50 Ω ±10%		
Microstrip/Stripline uncoupled differential impedance specification	100 Ω ±15%		[2,3]
Microstrip nominal trace width	Design dependent	Design dependent	
Microstrip nominal trace space	Design dependent	Design dependent	[3]
Microstrip/Stripline trace length	< 8 inches		Table 13-2
Microstrip/Stripline pair-to-pair space (edge-to-edge)	≥ 7 times the dielectric thickness		
Microstrip/Stripline bus-to-bus spacing	≥ 7 times the dielectric thickness		
Matching traces within a pair (P and N)	< 5 mils		
Keep pair-to-pair length differences	< 2 inches		

1. Pair-to-pair spacing ≥ 7 times the dielectric thickness for a maximum distance of 500 mils from the pin. The phase tolerance between MDI_P and MDI_N is <5mils.
2. Board designers should ideally target 100 Ω ±10%. If it's not feasible (due to board stackup) it is recommended that board designers use a 95 Ω ±10% target differential impedance for MDI with the expectation that the center of the impedance is always targeted at 95 Ω. The ±10% tolerance is provided to allow for board manufacturing process variations and not lower target impedances. The minimum value of impedance cannot be lower than 90 Ω.
3. Simulation shows 80 Ω differential trace impedances degrade MDI return loss measurements by approximately 1 dB from that of 90 Ω.

Table 13-2. Maximum Trace Lengths Based on Trace Geometry and Board Stackup

Dielectric Thickness (mils)	Dielectric Constant (DK) at 1 MHz	Width / Space/ Width (mils)	Pair-to-Pair Space (mils)	Nominal Impedance (Ω)	Impedance Tolerance (+/-%)	Maximum Trace Length (inches)[1]
2.7	4.05	4 / 10 / 4	19	95[2]	17[2]	3.5
2.7	4.05	4 / 10 / 4	19	95[2]	15[2]	4
2.7	4.05	4 / 10 / 4	19	95	10	5
3.3	4.1	4.2 / 9 / 4.2	23	100[2]	17[2]	4
3.3	4.1	4.2 / 9 / 4.2	23	100	15	4.6
3.3	4.1	4.2 / 9 / 4.2	23	100	10	6
4	4.2	5 / 9 / 5	28	100[2]	17[2]	4.5
4	4.2	5 / 9 / 5	28	100	15	5.3
4	4.2	5 / 9 / 5	28	100	10	7

1. Longer MDI trace lengths can be achievable, but might make it more difficult to achieve IEEE conformance. Simulations have shown deviations are possible if traces are kept short. Longer traces are possible; use cost considerations and stackup tolerance for differential pairs to determine length requirements.
2. Deviations from 100 Ω nominal and/or tolerances greater than 15% decrease the maximum length for IEEE conformance.

Note: Use the MDI Differential Trace Calculator to determine the maximum MDI trace length for your trace geometry and board stackup. Contact your Intel representative for access.

The following factors can limit the maximum MDI differential trace lengths for IEEE conformance:

- Dielectric thickness
- Dielectric constant
- Nominal differential trace impedance
- Trace impedance tolerance
- Copper trace losses
- Additional devices, such as switches, in the MDI path might impact IEEE conformance.

Board geometry should also be factored in when setting trace length.

13.2.5.5 Ground Planes Under a Magnetics Module

The magnetics module chassis or output ground (secondary side of transformer) should be separated from the digital or input ground (primary side) by a physical separation of 80 mils minimum. Splitting the ground planes beneath the transformer minimizes noise coupling between the primary and secondary sides of the transformer and between the adjacent coils in the magnetics. This arrangement also improves the common mode choke functionality of magnetics module.

Integrated magnetics perform this ground plane separation internally. As a result, the ground plane split previously described is not needed when using integrated magnetics.

13.2.6 PHY MDI Lane Swap Configuration

The X550 provides flexible MDI LAN swaps for MDI board routing (see Figure 13-6) via an NVM setting.

- Default configuration - 0, 1, 2, 3 <-> A, B, C, D
- MDI swap configuration - 3, 2, 1, 0 <-> A, B, C, D
- 5th channel swap not supported (MDIx_4_P and MDIx_4_N)

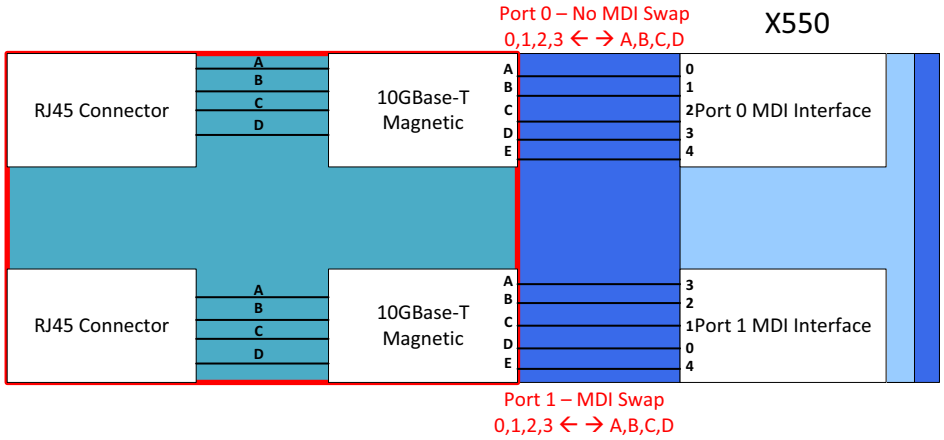


Figure 13-6. MDI Lane Swap Configuration

13.2.7 Center Tap Connection Via Capacitors to Ground

The X550 has a voltage-mode driver. When using it in 10GBASE-T applications, it is required that a center tap be de-coupled to ground. When using an integrated magnetic, a 0.1 μF capacitor should be connected from each center tap pin to ground. Similarly, when using a discrete magnetic, add a 0.1 μF capacitor to each center tap pin for a total of five capacitors, as previously described.

13.3 Connecting the Power Supply Delivery Network

For the X550, it is necessary to provide an adequate power supply delivery network such that the performance of the device meets expected performance requirements. To achieve this goal, both high speed and bulk decoupling capacitor networks are required in the design to ensure that frequency performance of the power supply delivery network is flat across the frequency spectrum. For details on how many high frequency and bulk decoupling capacitors are necessary to achieve this requirement along with the associated values of these capacitors, refer to the *X550 10GBASE-T Dual Port Ethernet Controller Reference Design* schematics.

To further ensure that these specific high frequency and bulk decoupling capacitors are placed in locations around and under the X550, Figure 13-7 breaks out the five required voltage supplies and color codes them according to power supply domain as a potential reference placement. Around the outside of the X550, highlighted bulk decoupling capacitors can be seen and because of their capacitance value and relative size, should be placed outside the BGA landing pattern. These decoupling capacitors still provide efficient bulk decoupling performance, as their capacitance value, along with the inductance of copper connecting them, still perform optimally.

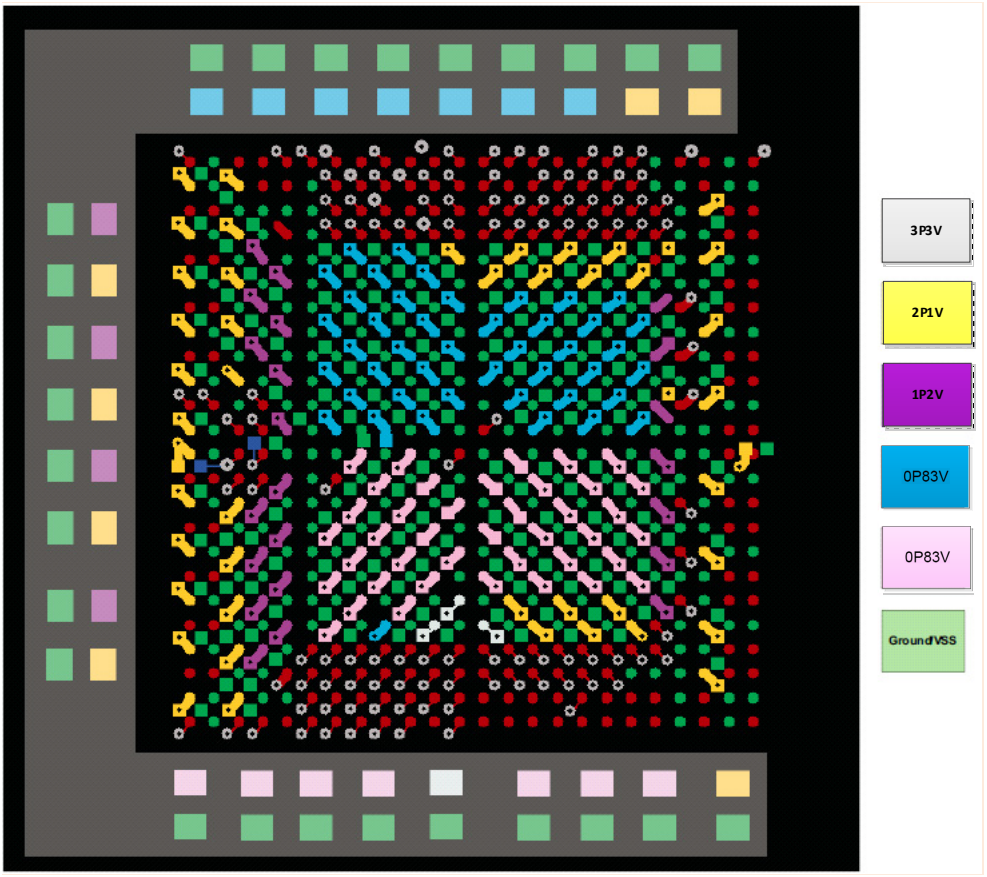


Figure 13-7. Power Supply Delivery Network

Within the BGA landing pattern it is recommended that the high speed decoupling capacitors be placed directly below the X550 on the secondary side of the PCB to ensure a low inductance path connection. For each power supply domain, the associated high speed decoupling capacitors should be placed in a way to provide coverage for all BGA locations. It is not Intel's recommendation that each power supply domain ball have a unique high-speed decoupling capacitor, but rather that the high speed decoupling capacitors required from the reference schematic for each power supply domain provide consistent coverage across the targeted BGA connection locations.

13.4 Connecting the Flash Interface

13.4.1 Connecting the Flash

The X550 provides support for a Serial Peripheral Interface (SPI) Flash device that is made accessible to the system through the following:

- Flash Base Address register (PCIe Control register at offset 0x14 or 0x18).
- An address range of the IOADDR register, defined by the I/O Base Address register (PCIe Control register at offset 0x18 or 0x20).
- Expansion ROM Base Address register (PCIe Control register at offset 0x30)

13.4.2 Supported Flash Devices

The X550 uses a SPI Flash. Several words of the Flash are accessed automatically by the X550 after reset to provide pre-boot configuration data before it is accessed by host software. The remainder of the Flash space is available to software for storing the MAC address, serial numbers, and additional information. For a complete description of the content stored in the Flash refer to Section 6.

Supported Op Code:

- Write Enable (06)
- Read Status Register (05)
- Write Status Register (01)
- Read Data (03)
- Byte/Page Program (02) (program 1 to 256 data bytes)
- 4 KB Sector Erase (20)
- Chip Erase (C7)

Note: For the Write Status register op code, the written data is 0x00 (to cancel the default protection).

The X550 contains a 3.3V Flash I/O.

13.5 Connecting Manageability Interfaces

SMBus and NC-SI are optional interfaces for pass-through and/or configuration traffic between the MC and the X550. For a description of the operation mode of these interfaces, refer to Section 7 and Section 12.

13.5.1 Connecting the SMBus Interface

To connect the SMBus interface to the chipset or the MC; connect the SMBD, SMBCLK and SMBALRT_N signals to the corresponding pins of the chipset/MC. These pins require pull-up resistors to the 3.3V supply rail.

Note: If the interface is not used, the previously mentioned pull-up resistors on the SMBD, SMBCLK and SMBALRT_N signals still have to be in place.

It is recommended that the SMBus be connected to the chipset or MC for the Flash recovery solution. If the connection is to a MC, it is able to send the Flash release command.

13.5.2 Connecting the NC-SI Interface

The NC-SI interface is a connection to an external MC. It operates as a single interface with an external MC, where all traffic (other than header redirection) between the X550 and the MC flows through the interface.

13.5.2.1 External MC

The external MC is required to meet the electrical specifications called out in the NC-SI specification.

13.5.2.2 Single and Multi-Drop Applications

Figure 13-8 shows the connectivity requirements and provides information about a single-drop application. This configuration only has a single package connected to the MC. The X550 does support a multi-drop NC-SI configuration architecture, including hardware arbitration support from the MC. refer to the NC-SI specification for requirements for multi-drop applications.

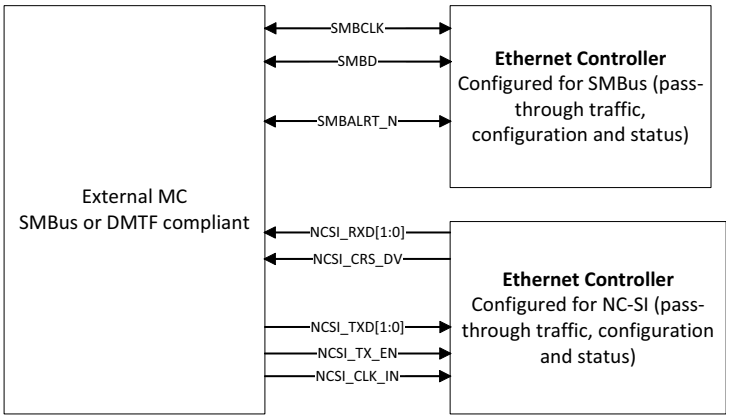


Figure 13-8. External MC Connections with NC-SI and SMBus

13.5.2.3 Pull-Ups and Pull-Downs

Depending on whether or not the NC-SI interface is used, different pull-up and pull-down resistors are required. Figure 13-8 shows the connectivity requirements necessary for interfacing a NC-SI compliant MC.

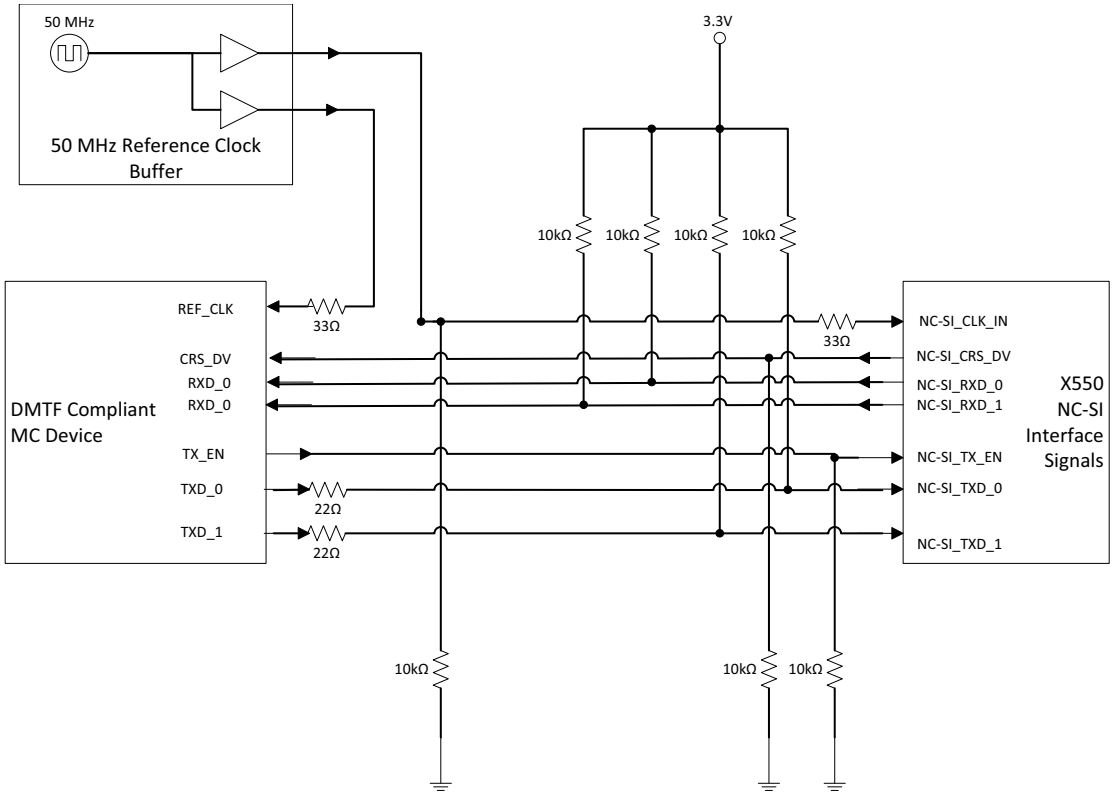


Figure 13-9. NC-SI Connections to an External MC

13.5.3 Layout Requirements

13.5.3.1 Board Impedance

The NC-SI manageability interface is a single ended signaling environment. It is recommended that a target board and trace impedance of 50 plus 20% and minus 10%. This ensures optimal signal integrity.

13.5.3.2 Trace Length Restrictions

It is recommended that a trace length maximum value from a board placement and routing topology perspective of eight inches for direct connect applications. This ensures signal integrity and quality is preserved from a design perspective and that compliance is met for NC-SI electrical requirements.

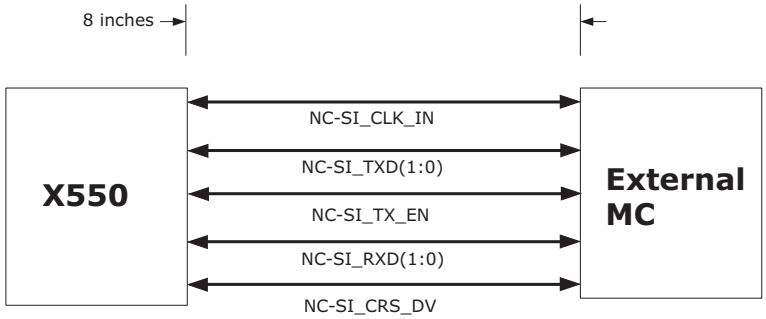


Figure 13-10. NC-SI Maximum Trace Length Requirement for Direct Connect Applications

For multi-drop applications, a spacing recommendation of a maximum of four inches between the two packages connected to the same MC. This is done to keep the overall length between the MC and the X550 within specification. See Figure 13-11

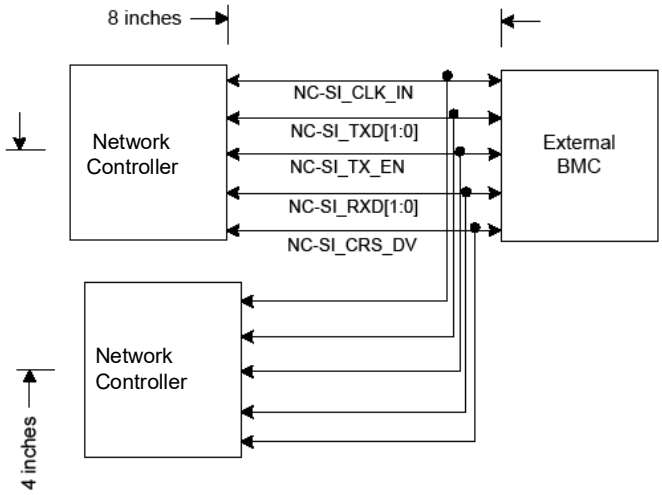


Figure 13-11. Spacing Recommendation for Multi-Drop Applications

13.5.3.3 Special Delay Requirements

To ensure that the X550 is able to communicate with a specification-compliant MC, the following guidelines must be followed:

- Total skew from the clock source to all devices should be less than 1 ns (less than or equal to an approximate 5.5 inch trace length skew).
- Each inch of trace has about 160 ps to 180 ps of delay, depending on the effective dielectric constant.

13.6 Connecting the Software-Definable Pins (SDPs)

The X550 has four SDPs per port that can be used for miscellaneous hardware or software-controllable purposes. These pins and their function are bound to a specific LAN device. The pins can each be individually configured to act as either input or output pins via Flash. The initial value in case of an output can also be configured in the same way. However, the X550 default for any of these pins is to be configured as outputs.

To avoid signal contention, all four pins (per port) are set as input pins until the Flash configuration has been loaded.

Choose the correct SDPs for specific applications. All pins are tri-state buffers. Consider that four of these pins (SDPx_0 – SDPx_3) can be used to support IEEE1588 auxiliary devices, input for external interrupts, indication for thermal-sensor event or as general purpose interrupt (GPI) inputs. To act as GPI pins, the desired pins must be configured as inputs. A separate GPI interrupt-detection enable is then used to enable rising-edge detection of the input pin (rising-edge detection occurs by comparing values sampled at 62.5 MHz, as opposed to an edge-detection circuit). When detected, a corresponding GPI interrupt is indicated in the Interrupt Cause register.

When connecting the SDPs to different digital signals please keep in mind that these are 3.3V signals and use level shifting if necessary.

The use, direction, and values of SDPs are controlled and accessed using fields in the Extended SDP Control (ESDP) and Extended OD SDP Control (EODSDP) registers.

13.7 Connecting the Light Emitting Diodes (LEDs)

The X550 provides four programmable high-current sink (active high) outputs per port to directly drive LEDs for link activity and speed indication. Each LAN device provides an independent set of LED outputs; these pins and their function are bound to a specific LAN device. Each of the four LED outputs can be individually configured to select the particular event, state, or activity, which is indicated on that output. In addition, each LED can be individually configured for output polarity, as well as for blinking versus non-blinking (steady-state) indication.

The LED ports are fully programmable through the Flash interface (LEDCTL register). In addition, the hardware-default configuration for all LED outputs can be specified via a Flash field, thus supporting LED displays configurable to a particular OEM preference.

Provide separate current limiting resistors for each LED connected.

Since the LEDs are likely to be placed close to the board edge and to external interconnect, take care to route the LED traces away from potential sources of EMI noise. In some cases, it might be desirable to attach filter capacitors.

13.8 Connecting Miscellaneous Signals

13.8.1 LAN Disable

The X550 has two signals that can be used for disabling Ethernet functions from system BIOS. LAN0_DIS_N and LAN1_DIS_N are the separated port disable signals. Each signal can be driven from a system output port. Choose outputs from devices that retain their values during reset. For example, some PCH GPIO outputs transition high during reset. It is important not to use these signals to drive LAN0_DIS_N or LAN1_DIS_N because these inputs are latched upon the rising edge of PE_RST_N or an in-band reset end.

A LAN port can also be disabled through Flash settings. See Section 4.4 for details. Note that setting the Flash *LAN_PCI_DIS* bit does not bring the PHY into power down.

Table 13-3. PCI Functions Mapping (Legacy Mode)

PCI Function #	LAN Function Select	Function 0	Function 1
Both LAN functions are enabled	0	LAN 0	LAN 1
	1	LAN 1	LAN 0
LAN 0 is disabled	x	LAN 1	Disable
LAN 1 is disabled	x	LAN 0	Disable
Both LAN functions are disabled	Both PCI functions are disabled. The X550 is in low power mode.		

Table 13-4. PCI Functions Mapping (Dummy Function Mode)

PCI Function #	LAN Function Select	Function 0	Function 1
Both LAN functions are enabled	0	LAN 0	LAN 1
	1	LAN 1	LAN 0
LAN 0 is disabled	0	Dummy	LAN 1
	1	LAN 1	Disable
LAN 1 is disabled	0	LAN 0	Disable
	1	Dummy	LAN 0
Both LAN functions are disabled	Both PCI functions are disabled. The X550 is in low power mode.		

When both LAN ports are disabled following a power on reset / LAN_PWR_GOOD / PE_RST_N / in-band reset, the LAN_DIS_N signals should be tied statically to low. At this state, the X550 is disabled, LAN ports are powered down, all internal clocks are shut and the PCIe connection is powered down (similar to L2 state).

13.8.2 BIOS Handling of Device Disable

Assume that in the following power up sequence the LANx_DIS_N signals are driven high (or is already disabled):

1. PCIe link is established following the PE_RST_N.
2. BIOS recognizes that the X550 should be disabled.
3. BIOS drives the LANx_DIS_N signals to the low level.
4. BIOS issues PE_RST_N or an in-band PCIe reset.
5. As a result, the X550 samples the LANx_DIS_N signals and enters the desired device-disable mode.
6. Re-enable could be done by driving high one of the LANx_DIS_N signals and then issuing a PE_RST_N to restart the X550.

13.9 Connecting the JTAG Port

The X550 contains a test access port (3.3 V only) conforming to the IEEE 1149.1-2001 Edition (JTAG) specification. To use the test access port, connect these balls to pads accessible by specific test equipment.

For proper operation, a pull-down resistor should be connected to the JTCK and JRST_N signals and pull up resistors to the JTDO, JTMS and JTDI signals.

A Boundary Scan Definition Language (BSDL) file describing the X550 is available for use in specific test environments.

13.10 Power On Reset (POR)

After power is applied, the X550 must be reset. There are two ways to do this:

1. Using the internal Power on Reset (POR) circuit with external LAN_PWR_GOOD signal pulled up.
2. Using the internal POR circuit with the external reset as LAN_PWR_GOOD signal generated by the system.

The power supply sequencing timing requirement between the 3.3V and 2.1V, 1.2V, and 0.83V power rails has to be met (see Section 12.3.8).

There are two options for implementing POR:

1. Internal POR (recommended as default).
    a. BYPASS_POR signal (pull down).
    b. LAN_PWR_GOOD signal (pull up).
2. External reset with LAN_PWR_GOOD signal:
    a. BYPASS_POR signal (pull down).
    b. LAN_PWR_GOOD signal generated by the system following device power-up timing.

13.11 Crystal Design Considerations

This section provides information regarding crystals for use with the X550.

All designs require an external clock. The only option for this clock source is a 50 MHz external crystal. The X550 uses the crystal to generate clocks for the high speed interfaces.

The chosen crystal vendor should be consulted early in the design cycle. Crystal manufacturers familiar with networking equipment clock requirements might provide assistance in selecting an optimum, low-cost solution.

13.11.1 Quartz Crystal

Quartz crystals are the mainstay of frequency components due to low cost and ease of implementation. They are available from numerous vendors in many package types with various specification options.

13.11.2 Vibrational Mode

Crystals are available in both fundamental and third overtone. Unless there is a special need for third overtone, use fundamental mode crystals. At any operating frequency, third overtone crystals are thicker and more rugged than fundamental mode crystals. Third overtone crystals are more suitable for use in military or harsh industrial environments. Third overtone crystals require a trap circuit (extra capacitor and inductor) in the load circuitry to suppress fundamental mode oscillation as the circuit powers up. Selecting values for these components is beyond the scope of this document.

13.11.3 Frequency Tolerance

The frequency tolerance for an Ethernet Platform LAN Connect is dictated by the IEEE 802.3 specification as ±50 parts per million (ppm). This measurement is referenced to a standard temperature of 25° C. It is recommended that a frequency tolerance of ±10 ppm be used.

13.11.4 Temperature Stability and Environmental Requirements

Temperature stability is a standard measure of how the oscillation frequency varies over the full operational temperature range (and beyond). Several optional temperature ranges are currently available, including -10 °C to +70 °C for industrial environments. Some vendors separate operating temperatures from temperature stability. Manufacturers can also list temperature stability as 50 ppm in their data sheets.

Note: Crystals also carry other specifications for storage temperature, shock resistance, and reflow solder conditions. Crystal vendors should be consulted early in the design cycle to discuss the application and its environmental requirements.

13.11.5 Calibration Mode

The terms series-resonant and parallel-resonant are used to describe crystal oscillator circuits. Specifying parallel mode is critical to determining how the crystal frequency is calibrated at the factory. A crystal specified and tested as series resonant oscillates without problem in a parallel-resonant circuit, but the frequency is higher than nominal by several hundred parts per million. The purpose of adding load capacitors to a crystal oscillator circuit is to establish resonance at a frequency higher than the crystal's inherent series resonant frequency.

13.11.6 Reference Crystal Circuit

Figure 13-12 illustrates a simplified schematic of the internal oscillator circuit. Pin X1 and X2 refers to XTAL1 and XTAL2 in the X550, respectively. The crystal and the capacitors form a feedback element for the internal inverting amplifier. This combination is called parallel-resonant, because it has positive reactance at the selected frequency. In other words, the crystal behaves like an inductor in a parallel LC circuit. Oscillators with piezoelectric feedback elements are also known as Pierce oscillators.

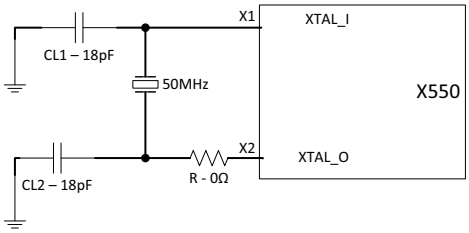


Figure 13-12. Oscillator Circuit

13.11.7 Crystal Load Capacitance

The formula for crystal load capacitance is as follows:

$$C_L = \frac{(C1 \cdot C2)}{(C1 + C2)} + C_{stray}$$

where C1 = C2 = 18 pF and C_{stray} = allowance for additional capacitance in pads, traces and the chip carrier within the Ethernet device package An allowance of 3 pF to 7 pF accounts for lumped stray capacitance. The calculated load capacitance is 16 pF with an estimated stray capacitance of about 5 pF. Individual stray capacitance components can be estimated and added. For example, surface mount pads for the load capacitors add approximately 2.5 pF in parallel to each capacitor. This technique is especially useful if Y1, C1 and C2 must be placed farther than approximately one-half (0.5) inch from the device. Thin circuit boards generally have higher stray capacitance than thick circuit boards. Oscillator frequency should be measured with a precision frequency counter where possible. The load specification or values of C1 and C2 should be fine-tuned for the design. As the actual capacitance load increases, the oscillator frequency decreases.

Note: C1 and C2 can vary by as much as 5% (approximately 1 pF) from their nominal values.

13.11.8 Shunt Capacitance

The shunt capacitance parameter is relatively unimportant compared to load capacitance. Shunt capacitance represents the effect of the crystal's mechanical holder and contacts. The shunt capacitance should equal a maximum of 5 pF.

13.11.9 Equivalent Series Resistance (ESR)

ESR is the actual component of the crystal's impedance at the calibration frequency, which the inverting amplifier's loop gain must overcome. ESR varies inversely with frequency for a given crystal family. The lower the ESR, the faster the crystal starts up. Use crystals with an ESR value of 50 Ω or better.

13.11.10 Driver Level

Drive level refers to power dissipation in use. The allowable drive level for a Surface Mounted Technology (SMT) crystal is less than its through-hole counterpart, because surface mount crystals are typically made from narrow, rectangular AT strips, rather than circular AT quartz blanks. Some crystal data sheets list crystals with a maximum drive level of 1 mW. However, Intel Ethernet controllers drive crystals to a level less than the suggested 0.3 mW value. This parameter does not have much value for on-chip oscillator use.

13.11.11 Aging

Aging is a permanent change in frequency (and resistance) occurring over time. This parameter is most important in its first year because new crystals age faster than old crystals. Use crystals with a maximum of ±5 ppm per year aging.

13.11.12 Reference Crystal

The normal tolerances of the discrete crystal components can contribute to small frequency offsets with respect to the target center frequency. To minimize the risk of tolerance-caused frequency offsets causing a small percentage of production line units to be outside of the acceptable frequency range, it is important to account for those shifts while empirically determining the proper values for the discrete loading capacitors, C1 and C2. Even with a perfect support circuit, most crystals oscillate slightly higher or slightly lower than the exact center of the target frequency. Therefore, frequency measurements (which determine the correct value for C1 and C2) should be performed with an ideal reference crystal. When the capacitive load is exactly equal to the crystal's load rating, an ideal reference crystal is perfectly centered at the desired target frequency.

13.11.13 Reference Crystal Selection

There are several methods available for choosing the appropriate reference crystal. These are listed below.

- If a Saunders and Associates (S&A) crystal network analyzer is available, discrete crystal components can be tested until one is found with zero or nearly zero ppm deviation (with the appropriate capacitive load). A crystal with zero or near zero ppm deviation is a good reference crystal to use in subsequent frequency tests to determine the best values for C1 and C2.
- If a crystal analyzer is not available, the selection of a reference crystal can be done by measuring a statistically valid sample population of crystals, which has units from multiple lots and approved vendors. The crystal, which has an oscillation frequency closest to the center of the distribution, should be the reference crystal used during testing to determine the best values for C1 and C2.
- It might also be possible to ask the approved crystal vendors or manufacturers to provide a reference crystal with zero or nearly zero deviation from the specified frequency when it has the specified CLoad capacitance.

When choosing a crystal, keep in mind that to comply with IEEE specifications for 100/ 1000/10GBase-T Ethernet LAN, the transmitter reference frequency must be precise within ±50 ppm. Intel® recommends using a transmitter reference frequency that is accurate to within ±10 ppm to account for variations in crystal accuracy due to crystal manufacturing tolerance.

13.11.14 Circuit Board

Since dielectric layers of the circuit board are allowed some reasonable variation in thickness, stray capacitance from the printed board (to the crystal circuit) also vary. If the thickness tolerance for the outer layers of dielectric are controlled within ±17 percent of nominal, the circuit board should not cause more than ±2 pF variation to the stray capacitance at the crystal. When tuning crystal frequency, it is recommended that at least three circuit boards are tested for frequency. These boards should be from different production lots of bare circuit boards. Alternatively, a larger sample population of circuit boards can be used. A larger population increases the probability of obtaining the full range of possible variations in dielectric thickness and the full range of variation in stray capacitance. Next, the exact same crystal and discrete load capacitors (C1 and C2) must be soldered onto each board and the LAN reference frequency should be measured on each circuit board. The circuit board, which has a LAN reference frequency closest to the center of the frequency distribution, should be used while performing the frequency measurements to select the appropriate value for C1 and C2.

13.11.15 Temperature Changes

Temperature changes can cause crystal frequency to shift. Frequency measurements should be done in the final system chassis across the system’s rated operating temperature range.

13.12 PCB Guidelines

This section describes the general PCB design guidance targeted as supplementary information in addition to the specific requirements and recommendations for individual interfaces. For items not directly covered in the specific interface sections, these guidance recommendations apply to the design.

13.12.1 Board Stack-Up Example

PCBs for these designs typically have six, eight, or more layers. Although the X550 does not dictate stackup, the following examples show typical stack-up options.

Microstrip Example:

- Layer 1 is a signal layer.
- Layer 2 is a ground layer.
- Layer 3 is used for power planes.
- Layer 4 is a signal layer. Careful routing is necessary to prevent crosstalk with layer 5.
- Layer 5 is a signal layer. Careful routing is necessary to prevent crosstalk with layer 4.
- Layer 6 is used for power planes.
- Layer 7 is a signal ground layer.
- Layer 8 is a signal layer.

Note: Layers 4 and 5 should be used mostly for low-speed signals because they are referenced to potentially noisy power planes that might also be slotted.

Stripline Example:

- Layer 1 is a signal layer.
- Layer 2 is a ground layer.
- Layer 3 is a signal layer.
- Layer 4 is used for power planes.
- Layer 5 is used for power planes.
- Layer 6 is a signal layer.
- Layer 7 is a signal ground layer.
- Layer 8 is a signal layer.

Note: To avoid the effect of the potentially noisy power planes on the high-speed signals, use offset stripline topology. The dielectric distance between the power plane and signal layer should be three times the distance between ground and signal layer.

This board stack-up configuration can be adjusted to conform to company-specific design rules.

13.12.2 Customer Reference Board Stack-Up Example

Layer Name	Plane Description	Layer Thickness (mil)	Copper Weight (oz)	Dielectric eR
	solder mask	**0.5**		**3.8**
Signal 1	SIGNAL	1.9	**1.5**	
	1080-prepreg	**2.7**		**4.0**
Plane 2	GND DDR PWR GND	1.3	**1.0**	
	core	**4.0**		**4.1**
Signal 3	SIGNAL	1.3	**1.0**	
	prepreg	12.0		**4.0**
Plane 4	POWER (Vddq, GND, Vtt)	2.6	**2.0**	
	core	4.0		**4.1**
Plane 5	POWER (12V input)	2.6	2.0	
	prepreg	6.0		**4.0**
Plane 6	POWER (PLL, VSA,	2.6	**2.0**	
	core	4.0		4.1
Plane 7	GND	2.6	2.0	
	prepreg	12.0		4.0
Signal 8	SIGNAL	1.3	1.0	
	core	4.0		4.1
Plane 9	GND DDR PWR GND	1.3	1.0	
	1080-prepreg	2.7		4.0
Signal 10	SIGNAL	1.9	1.5	
	solder mask	0.5		3.8
		71.8		

13.12.3 Intel Reference Board Stack-Up Example

Form Rev 04/27/2009

INTEL TWIN LAKES 04/05/2010

Material: PANASONIC R1755V/R1650V

Layer	Cu Clad (oz)	Thickness (in)	% Cu	Prepreg Style	Prepreg Qty	Ref	50 ohms +- 10% SE W	85 ohms +- 10% Differential W	85 ohms +- 10% Differential S	100 ohms +- 10% Differential W	100 ohms +- 10% Differential S
1	0.5	0.00182	F			L2		0.0075	0.006	0.0048	0.0052
				106-74%							
				1080-67%							
		0.00468 OK		3313-56%							
				2116-54%	1						
				7628-44%							
2	1	0.00135	80%								
		0.004									
3	1	0.00135	80%			L2,4	0.0049				
				106-74%			.005 @ 49 OHMS				
				1080-67%							
		0.00701 OK		3313-56%							
				2116-54%							
				7628-44%	1						
4	1	0.00135	80%								
		0.005									
5	1	0.00135	40%			L4,6	0.0053				
				106-74%			.005 @ 52 OHMS				
				1080-67%							
		0.00593 OK		3313-56%							
				2116-54%							
				7628-44%	1		.005 @ 52 OHMS				
6	1	0.00135	40%			L5,7	0.0053				
		0.005									
7	1	0.00135	80%								
				106-74%							
				1080-67%							
		0.00701 OK		3313-56%							
				2116-54%							
				7628-44%	1						
8	1	0.00135	80%								
		0.004									
9	1	0.00135	80%								
				106-74%							
				1080-67%							
		0.00468 OK		3313-56%							
				2116-54%	1						
				7628-44%							
10	0.5	0.00182	F			L9		0.0075	0.006	0.0048	0.0052

PLATED THICKNESS = 0.0618

INSTRUCTIONS - EACH LAYER MUST HAVE CU THICKNESS AS "0" OR .5, 1, 2, 3 ETC
EACH DIELECTRIC SPACE MUST HAVE # PLYS EACH STYLE AT "0" OR 1,2,3, ETC
EACH CORE LAYER MUST HAVE CORE NOMINAL WITHOUT CU - .010 , .0102. .0052 ETC
CORE TOLERANCE IS NOT CALCULATED AND NEEDS TO BE CONSIDERED

13.12.4 Via Usage

Use vias to optimize signal integrity. Figure 13-13 shows correct via usage. Figure 13-14 shows the type of topology that should be avoided.

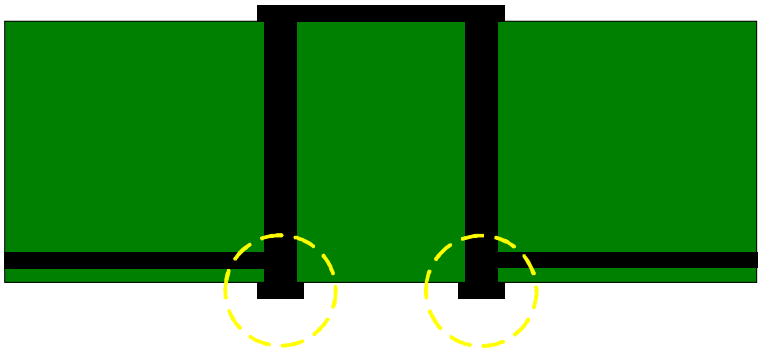


Figure 13-13. Correct Via Usage

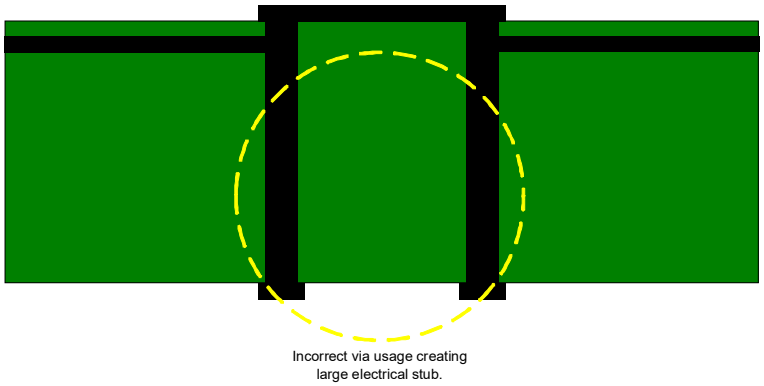


Figure 13-14. Incorrect Via Usage

Any via stubs on the MDI differential signal traces must be less than 35 mils in length. Keeping MDI signal via stubs less than or equal to 20 mils is preferable.

Place ground vias adjacent to signal vias used for the MDI interface. DO NOT embed vias between the high-speed signals, but place them adjacent to the signal vias. This helps to create a better ground path for the return current of the AC signals, which also helps address impedance mismatches and EMC performance.

It is recommend that, in the breakout region between the via and the capacitor pad, target a Z0 for the via to capacitor trace equal to 50 Ω. This minimizes impedance imbalance.

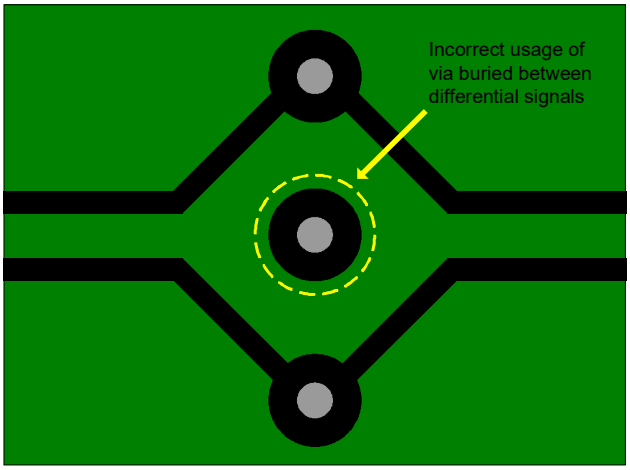


Figure 13-15. Incorrect Via Usage for Differential Pair

13.12.5 Reference Planes

Do not cross plane splits with the MDI high-speed differential signals. This causes impedance mismatches and negatively affects the return current paths for the board design and layout. See Figure 13-16.

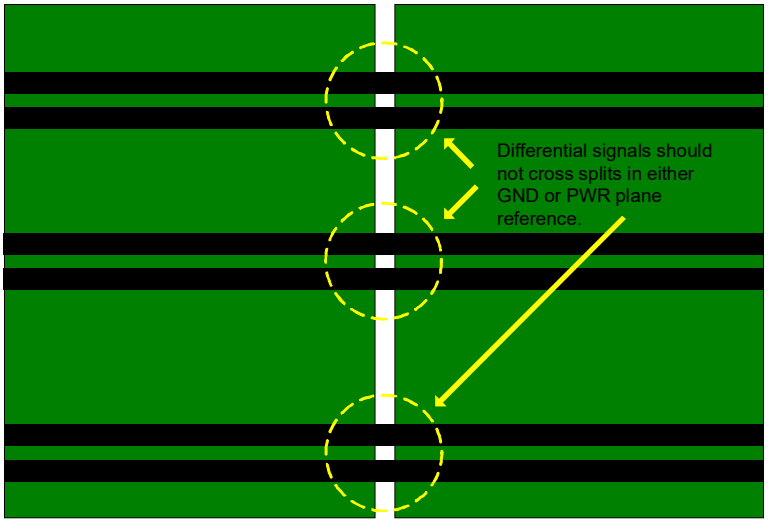


Figure 13-16. Improper Differential Signal Routing - Plane Split

Traces should not cross power or ground plane splits if at all possible. Traces should stay seven times the dielectric height away from plane splits or voids. If traces must cross splits, capacitive coupling should be added to stitch the two planes together to provide a better AC return path for the high-speed signals. To be effective, the capacitors should be have low ESR and low equivalent series inductance.

Note: Even with plane split stitching capacitors, crossing plane splits is extremely high risk for 10GBASE-T designs.

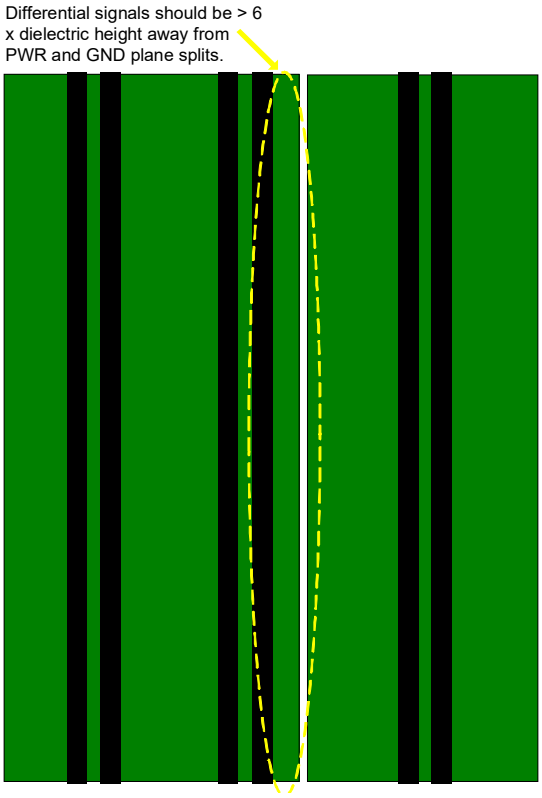


Figure 13-17. Differential Signal Routing - Plane split and Void Proximity

It is recommended that the MDI signals stay at least seven times the dielectric height away from any power or ground plane split. This improves impedance balance and return current paths.

If a high-speed signal needs to reference a power plane, ensure that the height of the secondary (power) reference plane is at least 3 x the height of the primary (ground) reference plane.

13.12.6 Reducing Circuit Inductance

Traces should be routed over a continuous reference plane with no interruptions. If there are vacant areas on a reference or power plane, the signal conductors should not cross the vacant area. Routing over a void in the reference plane causes impedance mismatches and usually increases radiated noise levels. Noisy logic grounds should NOT be located near or under high-speed signals or near sensitive analog pin regions of the LAN silicon. If a noisy ground area must be near these sensitive signals or IC pins, ensure sufficient decoupling and bulk capacitance in these areas. Noisy logic and switching power supply grounds can sometimes affect sensitive DC subsystems such as analog to digital conversion, operational amplifiers, etc.

All ground vias should be connected to every ground plane; and similarly, every power via should be equally potential power planes. This helps reduce circuit inductance. Another recommendation is to physically locate grounds to minimize the loop area between a signal path and its return path. Rise and fall times should be as slow as possible while still meeting the relevant electrical requirements because signals with fast rise and fall times contain many high frequency harmonics, which can radiate significantly. The most sensitive signal returns closest to the chassis ground should be connected together. This results in a smaller loop area and reduces the likelihood of crosstalk. The effect of different configurations on the amount of crosstalk can be studied using electronics modeling and simulation software.

13.12.7 Signal Isolation

To maintain the best signal integrity, keep digital signals far away from the analog traces. A good rule to follow is no digital signal should be within 7x to 10x dielectric height of the differential pairs. If digital signals on other board layers cannot be separated by a ground plane, they should be routed at a right angle (90 degrees) to the differential signal traces. If there is another Ethernet controller on the board, take care to keep the differential pairs away from that circuit. The same thing applies to switching regulator traces.

Rules to follow for signal isolation:

- Separate and group signals by function on separate board layers if possible. Maintain a separation that is at least seven times the thinnest adjacent dielectric height between all differential pairs (Ethernet) and other nets, but group associated differential pairs together.
- Over the length of the trace run, each differential pair should be at least seven times the thinnest adjacent dielectric height away from any parallel signal traces.
- Physically group together all components associated with one clock trace to reduce trace length and radiation.
- Isolate other I/O signals from high-speed signals to minimize crosstalk. Crosstalk can increase radiated EMI and can also increase susceptibility to EMI from other signals.
- Avoid routing high-speed LAN traces near other high-frequency signals associated with a video controller, cache controller, processor, or other similar devices.

13.12.8 Traces for Decoupling Capacitors

Traces between decoupling and I/O filter capacitors should be as short and wide as practical. Long and thin traces are more inductive and reduce the intended effect of decoupling capacitors. Also, for similar reasons, traces to I/O signals and signal terminations should be as short as possible. Vias to the decoupling capacitors should be sufficiently large in diameter to decrease series inductance.

13.12.9 Power and Ground Planes

Good grounding requires minimizing inductance levels in the interconnections and keeping ground returns short, signal loop areas small, and locating decoupling capacitors at or near power inputs to bypass to the signal return. This significantly reduces EMI radiation.

These guidelines reduce circuit inductance in NICs and LOMs:

- Route traces over a continuous plane with no interruptions. Do not route over a split power or ground plane. If there are vacant areas on a ground or power plane, avoid routing signals over the vacant area. Routing signals over power or ground voids increases inductance and increases radiated EMI levels.
- Use distance and/or extra decoupling capacitors to separate noisy digital grounds from analog grounds to reduce coupling. Noisy digital grounds can affect sensitive DC subsystems.
- All ground vias should be connected to every ground plane, and every power via should be connected to all power planes at equal potential. This helps reduce circuit inductance.
- Physically locate grounds between a signal path and its return. This minimizes the loop area.
- Avoid fast rise/fall times as much as possible. Signals with fast rise and fall times contain many high frequency harmonics, which can radiate EMI.
- Do not route high-speed signals near switching regulator circuits.
- It’s acceptable to put ground fill or thieving on the trace layers, but preferably not closer than 50 mils to the differential traces and the connector pins.
- If differential traces must be routed on another layer, the signal vias should carry the signal to the opposite side of the PCB (to be near the top of the PCB), AND if the high-speed signals are being routed between two connectors on the same board, before the signal traces reach the second connector, they must return to the original signal layer (before reaching the connector pin). This strategy keeps via stubs short without requiring back drilling.
- Each time differential traces make a layer transition (pass through a pair of signal vias), there must be at least one ground via located near each signal via. Two ground vias near each signal via is better. See Figure 13-18 and Figure 13-19.

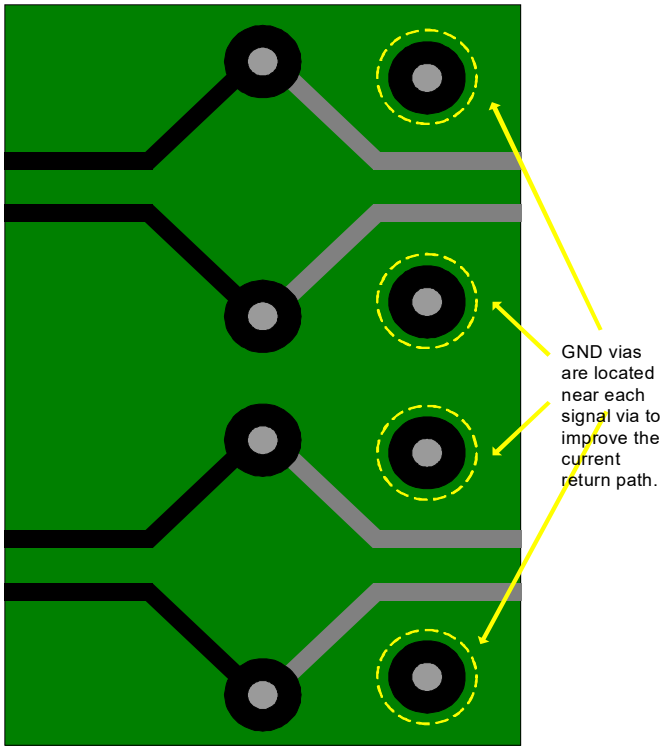


Figure 13-18. Return Path Vias for Differential Signals - Acceptable Example

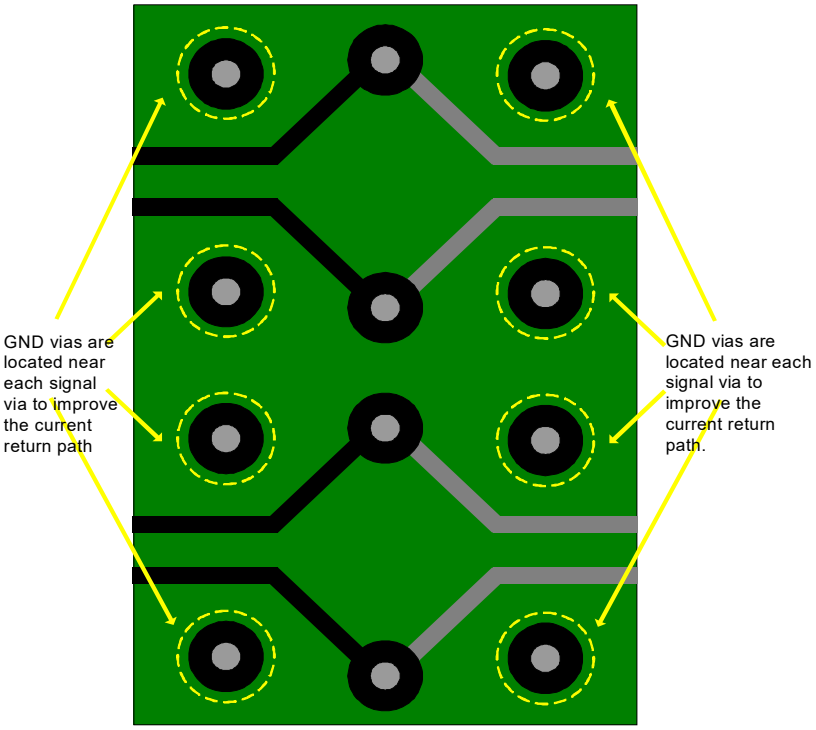


Figure 13-19. Return Path Vias for Differential Signals - Optimal Example

If the PCB fabrication process permits it, it's best to remove signal via pads on unconnected metal layers. See Figure 13-20 and Figure 13-21.

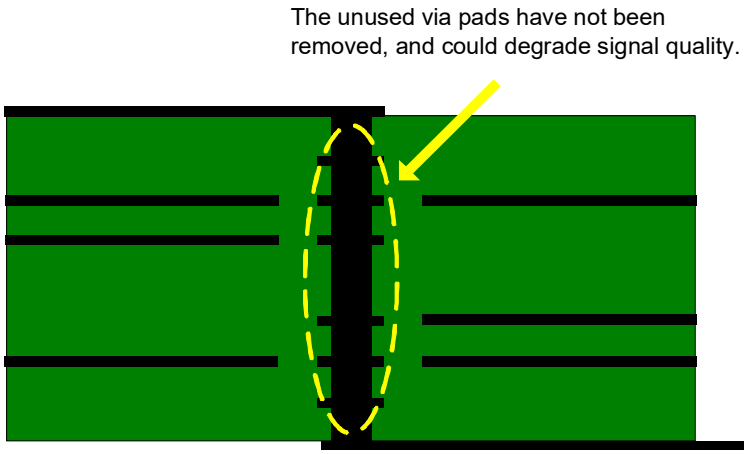


Figure 13-20. Signal Vias: Improper Padstack Example

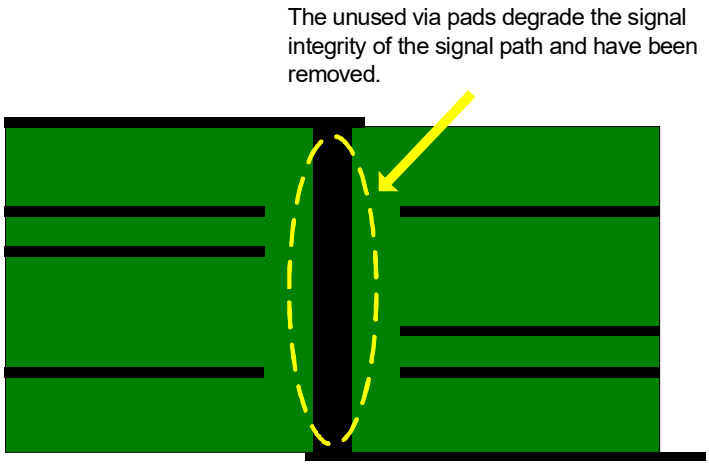


Figure 13-21. Signal Vias: Optimal Padstack

On metal layers where signal vias need to have via pads, it is desirable to reduce capacitance between the signal vias and ground-plane layers. The anti-pad diameters should be up to 20 mils larger than the via pad diameters. See Figure 13-22. Clearance between the pad and the surrounding metal should be >= 10 mils.

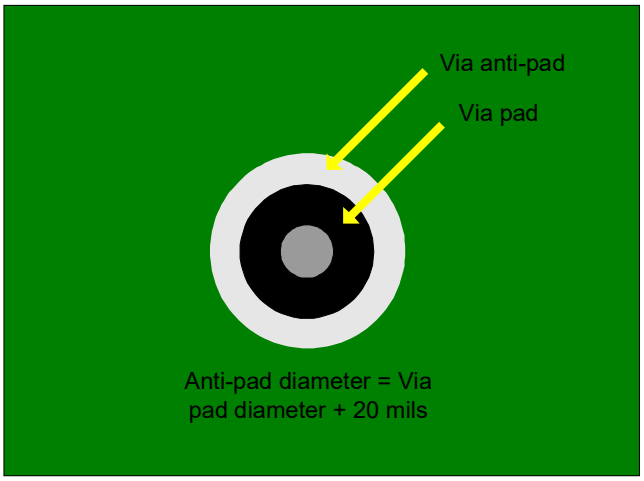


Figure 13-22. Anti-Pad Geometry

Each time differential signal vias pass through a plane layer, within each differential pair, the anti-pads should overlap. See Figure 13-23, Figure 13-24 and Figure 13-25.

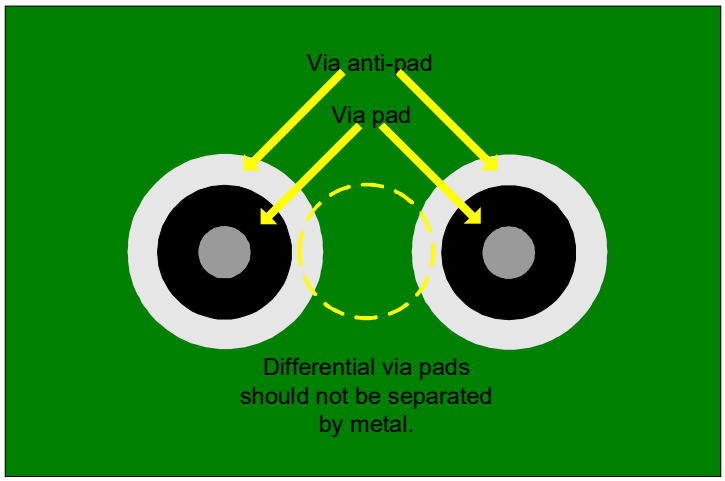


Figure 13-23. Differential Signal Vias: Improper Anti-pad Geometry

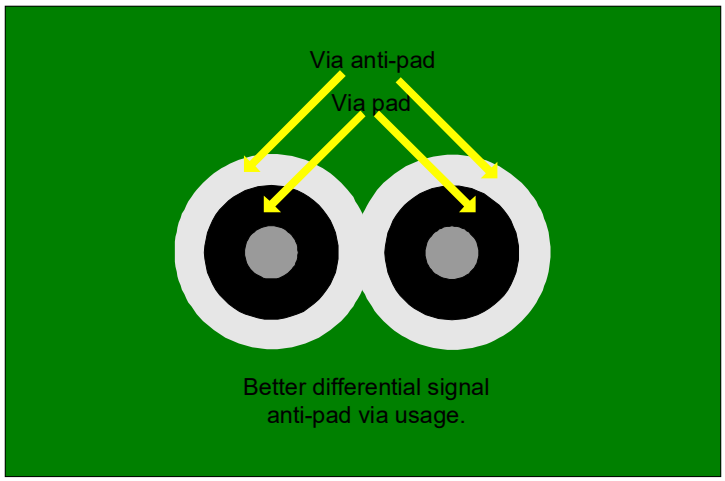


Figure 13-24. Differential Signal Vias: Acceptable Example

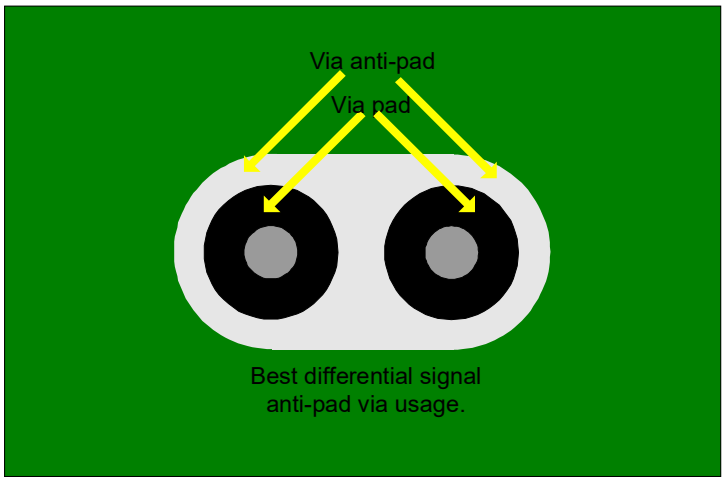


Figure 13-25. Differential Signal Vias: Optimal Example

13.12.10 Recommended Simulations

10GBASE-T signaling frequencies have frequency content in the range of 800 MHz and below so relatively short stubs, small discontinuities, and fairly small in-pair trace length differences can cause an undesirable increase in bit errors. Before ordering PCBs, verify that:

- Planned 10GBASE-T signal trace routing on the PCB complies with the interconnect characteristics recommended in IEEE 802.3an.
- With sufficient advance notice, Intel engineers can provide assistance under these conditions:
  - Trace routing should be optimized prior to the next steps – request a layout review (must be willing to provide board stack-up information and the MDI traces CAD artwork).
  - After MDI traces have been optimized, if the IEEE recommended electrical characteristics are still not being met, end-to-end MDI board channels S-parameter models should be extracted (preferably in Touchstone* S4p format) for additional investigative simulations by Intel signal integrity engineers. Please request the required S-parameter frequency range, step size, etc., before extracting Touchstone S-parameter models.

13.13 Bill Of Material (BOM)

Table 13-5 lists the BOM materials for all X550 LAN on Motherboard (LOM) designs.

Table 13-5. X550 LOMs

Component	Manufacturer	Manufacturer Part Number	Quantity
Crystal	TXC MMD Rami/Raltron	7A50020001 – 50 MHz V16DB1-50.000 MHz H130A-50.000-16-F-1010-TR-NS1	1
Discrete Magnetic	Bel-Fuse Pulse	S558-10GB-10 H7137NL	2
Integrated Magnetic	Bel-Fuse Bel-Fuse Bel-Fuse Pulse Tyco (TE) Delta Foxconn	G13-152T-099 G23-21YC-083E G18-48NE-083E JT4-1158HL 1840497-1 RJTGE1G4172J JFM5801J-710G-4F	2
RJ45	Lotes Bel-Fuse	ABA-JKM-002-Y02/ABA-JKM-002-Y03 SS60300-010 (No LED)	2
3.3 V Flash 2 MB	AMIC Spansion ESMT Winbond Macronix EON Silicon	A25L016 S25FL216K/S25FL116K F25L16PA/F25L16QA W25Q16DV/W25Q16CL MX25L1606E/MX25L1633E EN25Q16A/EN25QH16	1
3.3 V Flash 4 MB	AMIC Spansion ESMT Fidelix Microchip Winbond Macronix Micron EON Silicon	A25L032 S25FL032B F25L32PA/F25L32QA FM25Q32A SST25VF032B W25Q32FV MX25L3206E/MX25L3235E M25P32/M25PX32/N25Q032A13ESC40/ N25Q032A13ESE40F/N25Q032A13ESCA0F EN25Q32B / EN25QH32	1
Power Supplies FB	TDK Corporation	MPZ2012S101A	3

Chapter 14 Thermal Design Recommendations

14.1 Introduction

This section can be used as an aid to designing a thermal solution for systems implementing the the X550.

Note: The information contained in this section is subject to change. Contact your Intel representative to ensure that you have the most current information.

Properly designed solutions provide adequate cooling to maintain the X550 case temperature (Tcase) at or below those listed in Table 14-2. The device 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. Heat sinks and higher airflow might be required if case temperatures exceed those listed in Table 14-2.

Information in this section includes:

- Section 14.6, "Packaging Terminology" — Provides definitions for terminology used in this section.
- Section 14.7, "Thermal Specifications" — Provides the X550 case temperature specifications and where to find power requirements. This sub-section also discusses thermal packaging techniques.
- Section 14.8, "Thermal Attributes" — Provides the X550 thermal characteristic data, package mechanical attributes, and package thermal characteristic data. Use this sub-section to determine your thermal solution requirements.
- Section 14.9, "Thermal Enhancements" — Discusses the use of heat sinks, heat sink attach methods, interfacing, and reliability.
- Section 14.10, "Measurements for Thermal Specifications" — Provides instructions for measuring the X550 case temperature with and without a heat sink.
- Section 14.11, "Conclusion" — Provides a summary.
- Section 14.12, "Heat Sink and Attach Suppliers" — Provides manufacturer contact information.
- Section 14.13, "PCB Guidelines" — Includes general PCB recommendations.

14.2 Intended Audience

The intended audience for this section is system design engineers using the X550. System designers are required to address component and system-level thermal challenges as the market continues to adopt products with higher-speeds and port densities. New designs might be required to provide better cooling solutions for silicon devices depending on the type of system and target operating environment.

14.3 Measuring Thermal Conditions

This section provides a method for determining the operating temperature of the X550 in a specific system based on case temperature. Case temperature is a function of the local ambient and internal temperatures of the component. This section specifies a maximum allowable Tcase for the X550.

14.4 Thermal Considerations

In a system environment, the temperature of a component is a function of both the system and component thermal characteristics. System-level thermal constraints consist of the local ambient temperature at the component, the airflow over the component and surrounding board, and the physical constraints at, above, and surrounding the component that might limit the size of a thermal enhancement (heat sink).

The component's case/die temperature depends on:

- Component power dissipation
- Size
- Packaging materials (effective thermal conductivity)
- Type of interconnection to the substrate and motherboard
- Presence of a thermal cooling solution
- Power density of the substrate, nearby components, and motherboard

All 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.

14.5 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 can 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 (see Section 12.2.2).

14.6 Packaging Terminology

The following is a list of packaging terminology used in this section:

- **FCBGA Flip Chip Ball Grid Array** — 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. An Integrated Heat Spreader (IHS) might be present for larger FCBGA packages for enhanced thermal performance (but IHS is not present for the X550).
- **Junction** — Refers to a P-N junction on the silicon. In this section, it is used as a temperature reference point (for example, θJA 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 1inch upstream from the component edge.
- **Lands** — The pads on the PCB to which BGA balls are soldered.
- **Printed Circuit Board** — PCB.
- **Printed Circuit Assembly (PCA)** — An assembled PCB.
- **Thermal Design Power (TDP)** — The estimated maximum possible/expected power generated in a component by a realistic application. Use the maximum power requirements listed in Table 14-2.
- **Linear Feet Per Minute** — LFM (airflow).
- **θ_{JA} (Theta JA)** — Thermal resistance junction-to-ambient, °C/W.
- **Ψ_{JT} (Psi JT)** — Junction-to-top (of package) thermal characterization parameter, °C/W. 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. Ψ_{JT} is easy to characterize in simulations or measurements, and is equal to Tj minus Tcase divided by the total TDP. This parameter can vary by environment conditions like heat sink and airflow.

14.7 Thermal Specifications

Important: Information contained in this section is preliminary and subject to change without notice.

To ensure proper operation of the X550, the thermal solution must maintain a case temperature at or below the values listed in Table 14-2. System-level or component-level thermal enhancements are required to dissipate the generated heat to ensure the case temperature never exceeds the maximum temperatures listed in Table 14-2. Table 14-1 lists the thermal performance parameters per JEDEC JESD51-2 standard. In Table 14-1 the θ_{JA} values should be used as reference only and can vary by system environment. Ψ_{JT} values also can vary by system environment, and are listed in Table 14-1 as the maximum value for the X550 simulations.

Analysis indicates that real applications are unlikely to cause the X550 to be at Tcase-max for sustained periods of time, given that Tcase should reasonably be expected to be a distribution of temperatures. Sustained operation at Tcase-max might affect long-term reliability of the X550 and the system. Also, sustained operation at Tcase-max should be evaluated during the thermal design process and steps taken to further reduce the Tcase temperature.

Good system airflow is critical to dissipate the highest possible thermal power. The size and number of fans, vents, and/or ducts as well as their placement in relation to components and airflow channels within the system determine airflow. Acoustic noise constraints might limit the size and types of fans, vents and ducts that can be used in a particular design.

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.

Table 14-1. Package Thermal Characteristics in Standard JEDEC Environment for Reference

Package	θ_{JA} (°C/W)	Ψ_{JT} (°C/W)
25 mm FCBGA without IHS[1]	16.5[2]	N/A
25 mm FCBGA without IHS –HS[3]	7.51[4]	0.17[5]
17 mm FCBGA without IHS[1]	TBD	N/A
17 mm FCBGA without IHS -HS[3]	7.51	0.17

1. Integrated Heat Spreader. the X550 is bare die.
2. Integrated Circuit Thermal Measurement Method-Electrical Test Method EIA/JESD51-1, Integrated Circuits Thermal Test Method Environmental Conditions - Natural Convection (Still Air), No Heat sink attached EIAJESD51-2.
3. Heat sink dimensions are specified in Section 14.9.3.
4. Natural Convection (Still Air), Heat sink attached.
5. Psi_JT is given as maximum value for a worst-case X550 scenario and might vary to a lesser value in some scenarios.

Table 14-2. The X550 Absolute Thermal Maximum Rating (°C)

Application	Estimated TDP (W)[1]	Tcase Max[2] (°C)[3]
X550	11.5 W @ 105 Tj max	103

1. Power value listed here is estimated maximum power, also known as TDP. TDP is a system design target associated with the maximum component operating temperature specifications. Maximum power values are determined based on typical DC electrical specification and maximum ambient temperature for a worst-case realistic application running at maximum utilization.
2. Tcase Max-hs is defined as the maximum case temperature with the default enhanced thermal solution attached.
3. This is a not to exceed maximum allowable case temperature.

14.7.1 Case Temperature

The X550 is designed to operate properly as long as the Tcase rating is not exceeded. Section 14.10.1 discusses proper guidelines for measuring the case temperature.

14.8 Thermal Attributes

Important: Information contained in this section is preliminary and subject to change without notice.

14.8.1 Designing for Thermal Performance

Section 14.12 and Section 14.13 describe the PCB and system design recommendations required to achieve the proper X550 thermal performance.

14.8.2 Typical System Definition

A system with the following attributes was used to generate thermal characteristics data:

- A heat sink case with the default enhanced thermal solution (see Section 14.9).
- Six-layer, 4.5 x 4 inch PCB.

Note: Keep the following in mind when reviewing the data that is included in this sub-section:

- All data is preliminary and is not validated against physical samples.
- Your system design might be significantly different.
- A larger board with more than six copper layers might improve the X550 thermal performance.

14.8.3 Package Mechanical Attributes

The X550 is packaged in a 25 mm or 17 mm FCBGA as shown in Section 12.7.4.

14.8.3.1 Package Thermal Characteristics

Refer to Table 14-3 for an aid in determining the optimum airflow and heat sink combination for the X550. See Section 12.2.2) for more details.

Table 14-3 lists Tcase as a function of airflow and ambient temperature at the TDP for a typical X550 system. Again, your system design might vary considerably from the typical system board environment used to generate the values listed in Table 14-1 and Table 14-2.

Note: Thermal models are available upon request (Flotherm*). Contact your local Intel sales representative for the X550 thermal models.

Table 14-3. X550 Expected Tcase (°C) for High Heat Sink in Figure 14-3 at 11.5 W (JEDEC Card)

Ambient/ Airflow (LFM)	0	50	100	150	200	250	300	350	400
45	121.7	113.6	98.67	90.27	85.04	81.37	78.61	76.42	74.64
50	125.6	117.3	103	94.82	89.7	86.09	83.38	81.23	79.47
55	129.8	120.9	107.4	99.36	94.34	90.81	88.14	86.02	84.29
60	133	124.2	111.6	103.9	98.97	95.51	92.9	90.81	89.1
65	133.5	127.4	115.8	108.3	103.6	100.2	97.63	95.59	93.91
70	135.9	131	120	112.8	108.2	104.9	102.4	100.4	98.7
75	139.1	134.7	124.1	117.3	112.8	109.5	107.1	105.1	103.5
80	142.2	138.6	128.2	121.7	117.2	114.2	111.8	109.9	108.3
85	145.6	142.5	132.3	126.1	121.9	118.8	116.5	114.6	113

Note: The red blocked value(s) indicate airflow/ambient combinations that exceed the allowable junction temperature for the X550 at 11.5 W.

14.9 Thermal Enhancements

Important: Information contained in this section is preliminary and subject to change without notice.

One method frequently used to improve thermal performance is to increase the device surface area by attaching a metallic heat sink to the component top. Increasing the surface area of the heat sink reduces the thermal resistance from the heat sink to the air, which increases heat transfer.

14.9.1 Clearances

To be effective, a heat sink should have a pocket of air around it that is free of obstructions. Though each design can have unique mechanical restrictions, the recommended clearances for a heat sink used with the X550 are shown in Figure 14-1 assuming one of the 40 x 40 mm reference heat sinks is selected. Retention clip selection is open, and example keep-outs and board through holes are shown in Figure 14-1 and Figure 14-2 for a torsion retention clip.

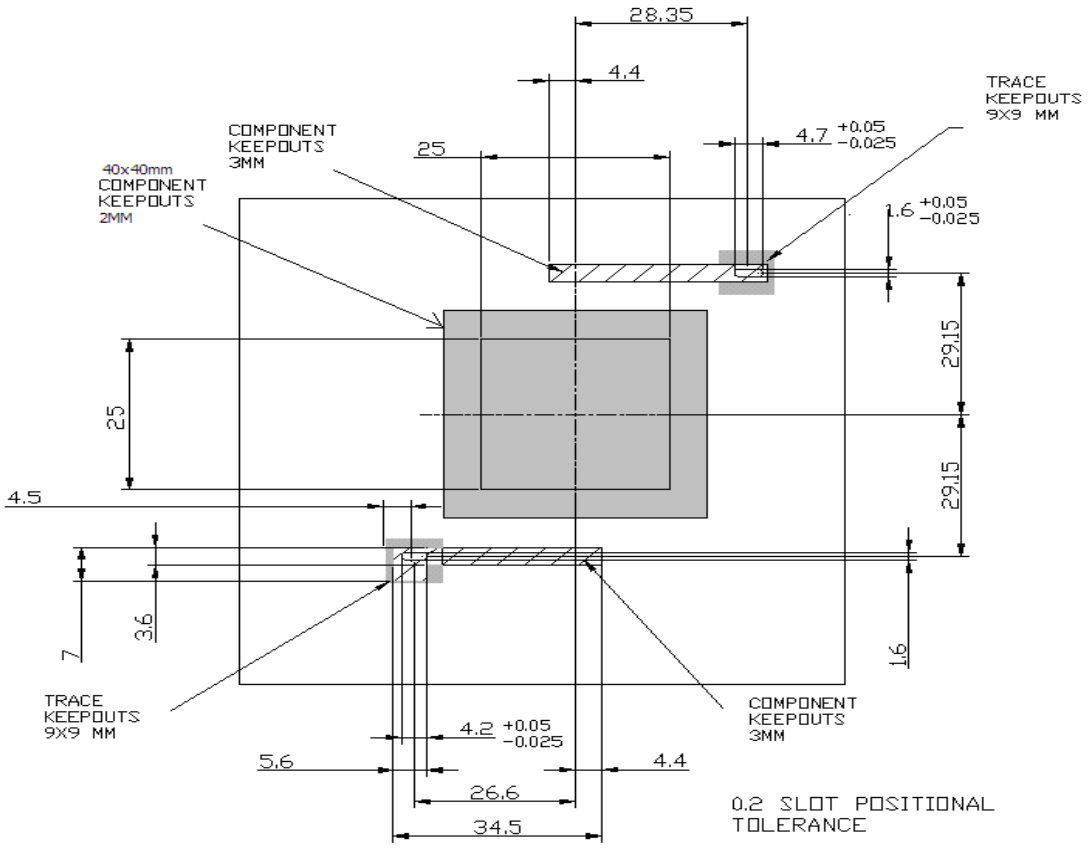


Figure 14-1. X550 Heat Sink Keep Out Restrictions (Top Side Keep Out)

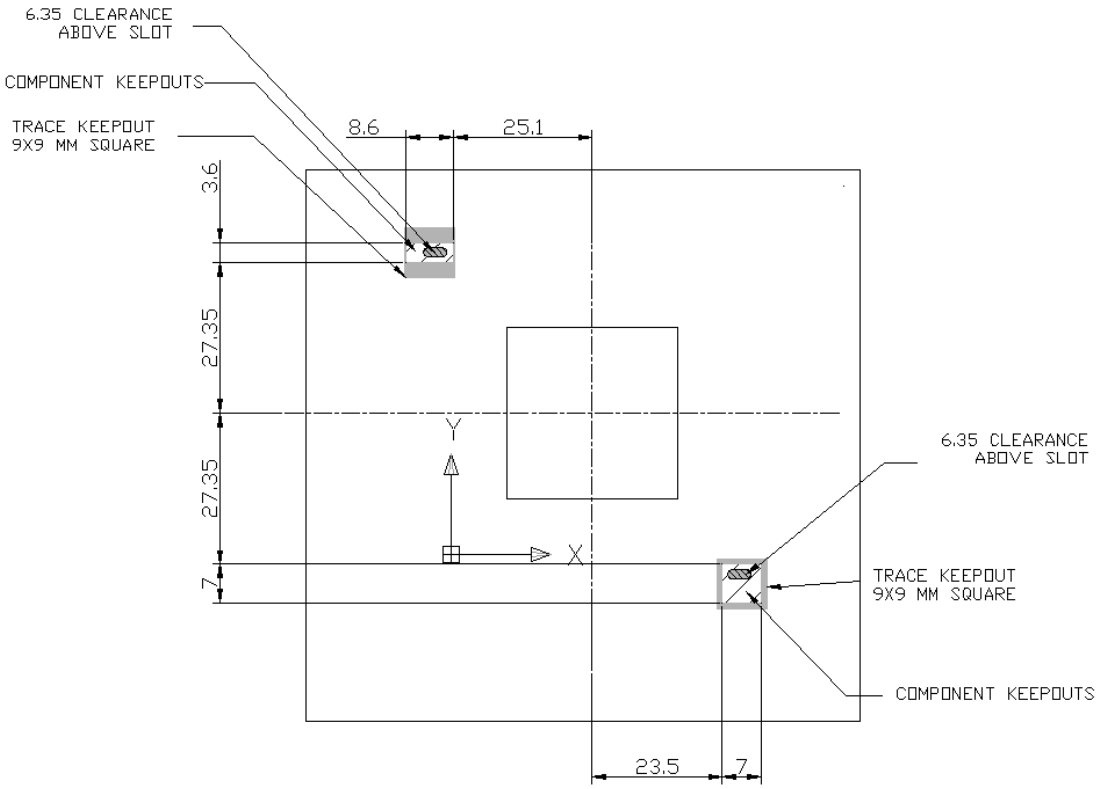


Figure 14-2. X550 Heat Sink Keep Out Restrictions (Bottom Side Keep Out)

14.9.2 Default Enhanced Thermal Solution

If you have no control over the end-user's thermal environment, or if you wish to bypass the thermal modeling and evaluation process, use the default enhanced thermal solutions (see Figure 14-2 and Figure 14-3). These solutions replicate the performance listed in Figure 14-3 at the TDP. If, after implementing the recommended enhanced thermal solution, the case temperature continues to exceed allowable values, additional cooling is needed. This additional cooling can be achieved by improving airflow to the component and/or adding additional thermal enhancements.

14.9.3 Extruded Heat Sinks

If required, the following extruded heat sinks are the suggested X550 thermal solutions (Figure 14-3). Also, see Figure 14-1and Figure 14-2 for heat sink information.

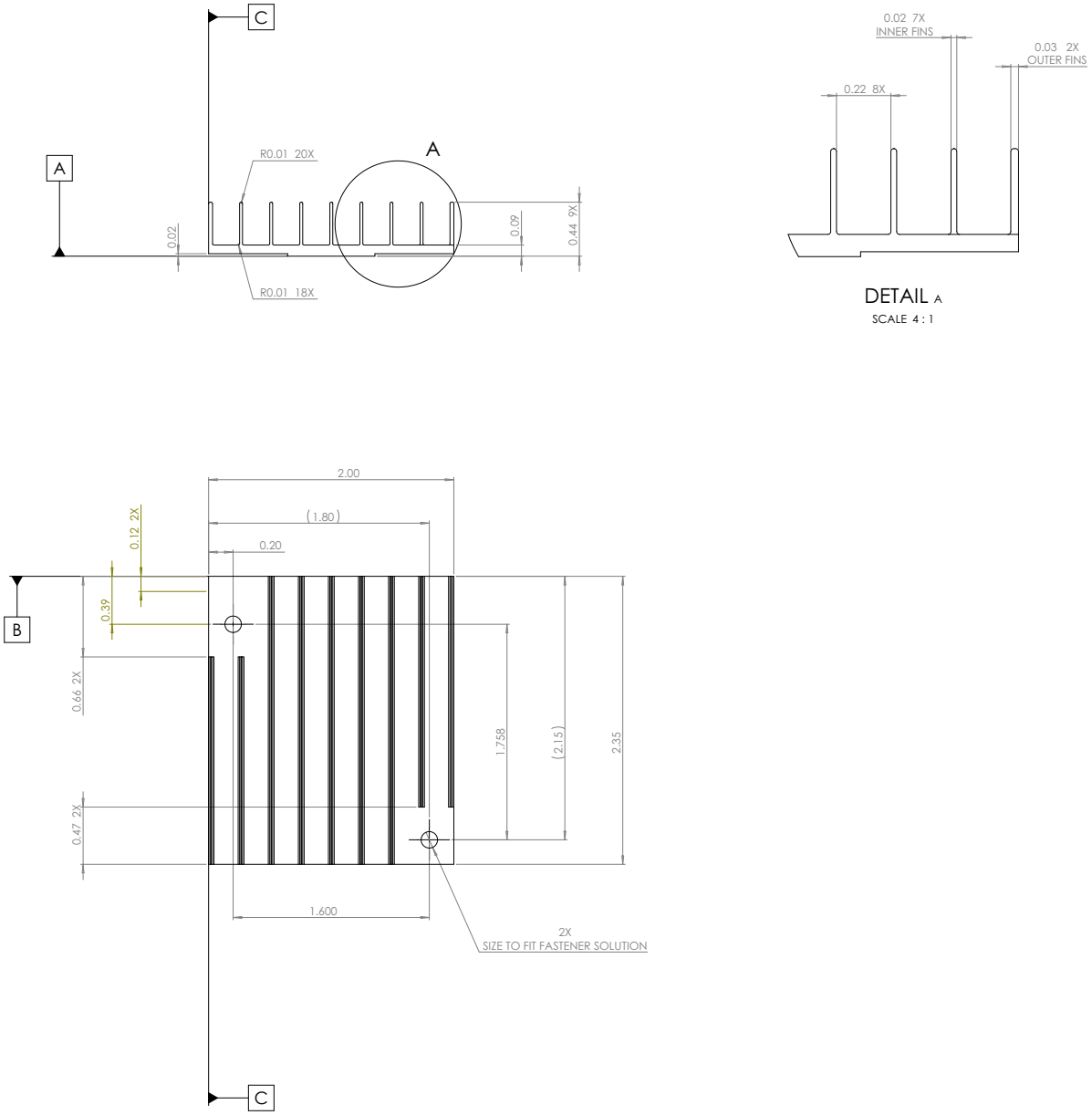


Figure 14-3. X550 Extruded Heat Sink

14.9.4 Attaching the Extruded Heat Sink

The extruded heat sink can be attached using clips or pins with a phase change thermal interface material.

14.9.4.1 Clips

A well-designed clip, in conjunction with a thermal interface material (tape, grease, etc.) often offers the best combination of mechanical stability and rework ability. Use of a clip requires significant advance planning as mounting holes are required in the PCB.

14.9.4.2 Thermal Interface (PCM45 Series)

The recommended thermal interface is PCM45 Series from Honeywell*. The PCM45 Series thermal interface pads are phase change materials formulated for use in high performance devices requiring minimum thermal resistance for maximum heat sink performance and component reliability. These pads consist of an electrically non-conductive, dry film that softens at device operating temperatures resulting in greasy-like performance. However, Intel has not fully validated the PCM45 Series TIM.

Each PCA, system and heat sink combination varies in attach strength. Carefully evaluate the reliability of double sided thermal interface tape attachments prior to high-volume use (see Section 14.9.5).

14.9.4.3 Avoid Damaging Die-side Capacitors with Heat Sink Attach

Capacitors on the die side are not protected and can be damaged during heat sink attach. If the heat sink is tilted from the die, it is possible that the heat sink will make contact with the capacitors prior to making contact with the package substrate. Figure 14-4 shows how the capacitors can be exposed to heat sink contact by drawing a plane from the die edge to the substrate edge. Figure 14-5 shows an example of the damage caused by heat sink contact. It is recommended that heat sinks be attached vertically, with the heat sink bottom surface parallel to the die surface to avoid contact with the capacitors.

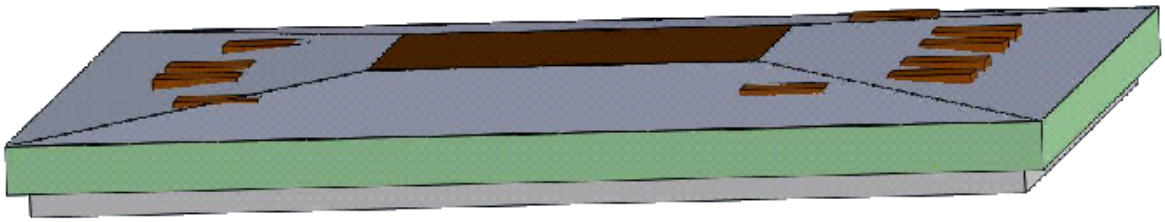


Figure 14-4. Die-Side Capacitors Exposed to Heat Sink Contact

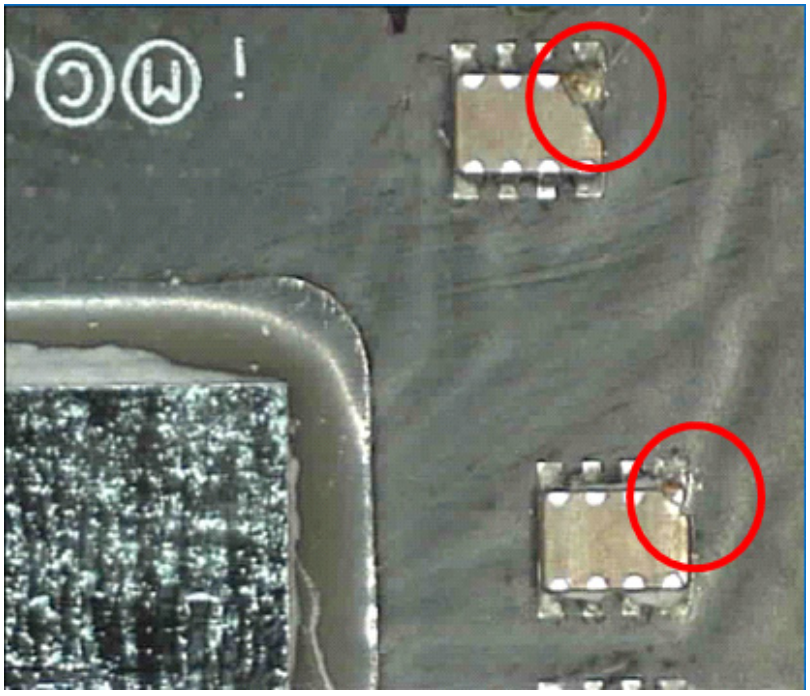


Figure 14-5. Damage Caused by Heat Sink Contact

14.9.4.4 Maximum Static Normal Load

The X550 has a bare die that is capable of sustaining a maximum static normal load of 15 lbf (67N). This load is a uniform compressive load in a direction perpendicular to the die top surface. This mechanical load limit must not be exceeded during heat sink installation, mechanical stress testing, standard shipping conditions, and/or any other use condition. Note that the heat sink attach solution must not include continuous stress to the package, with the exception of a uniform load to maintain the heat-sink-to-package thermal interface. This load specification is based on limited testing for design characterization, and is for the package only.

14.9.5 Reliability

Each PCA, system and heat sink combination varies in attach strength and long-term adhesive performance. Carefully evaluate the reliability of the completed assembly prior to high-volume use. Some reliability recommendations are listed in Table 14-4.

Table 14-4. Reliability Validation

Test[1]	Requirement	Pass/Fail Criteria[2]
Mechanical Shock	50 G trapezoidal, board level 11 ms, 3 shocks/axis	Visual and Mechanical Check.
Random Vibration	7.3 G, board level 45 minutes/axis, 50 to 2000 Hz	Visual and Mechanical Check.
High-Temperature Life	85 °C 2000 hours total Checkpoints occur at 168, 500, 1000, and 2000 hours	Visual and Mechanical Check.
Thermal Cycling	Per-target environment (for example: -40 °C to +85 °C) 500 cycles	Visual and Mechanical Check.
Humidity	85% relative humidity 85 °C, 1000 hours	Visual and Mechanical Check.

1. These tests were performed on a sample size of at least 12 assemblies from 3 lots of material (total = 36 assemblies).
2. Additional pass/fail criteria can be added at your discretion.

14.9.6 Thermal Interface Management for Heat Sink Solutions

To optimize the X550 heat sink design, it is important to understand the interface between the silicon die and the heat sink base. Thermal conductivity effectiveness depends on the following:

- Bond line thickness.
- Interface material area.
- Interface material thermal conductivity.

14.9.6.1 Bond Line Management

The gap between the silicon die and the heat sink base impacts heat sink solution performance. The larger the gap between the two surfaces, the greater the thermal resistance. The thickness of the gap is determined by the flatness of both the heat sink base and the silicon die, plus the thickness of the thermal interface material (for example, PSA, thermal grease, epoxy) used to join the two surfaces.

14.9.6.2 Interface Material Performance

The following factors impact the performance of the interface material between the silicon die and the heat sink base:

- Thermal resistance of the material.
- Wetting/filling characteristics of the material.

14.9.6.2.1 Thermal Resistance of the Material

Thermal resistance describes the ability of the thermal interface material to transfer heat from one surface to another. The higher the thermal resistance, the less efficient the heat transfer. The thermal resistance of the interface material has a significant impact on the thermal performance of the overall thermal solution. The higher the thermal resistance, the larger the temperature drop required across the interface.

14.9.6.2.2 Wetting/Filling Characteristics of the Material

The wetting/filling characteristic of the thermal interface material is its ability to fill the gap between the silicon die top surface and the heat sink. Since air is an extremely poor thermal conductor, the more completely the interface material fills the gaps, the lower the temperature-drop across the interface, increasing the efficiency of the thermal solution.

14.10 Measurements for Thermal Specifications

Determining the thermal properties of the system requires careful case temperature measurements. Guidelines for measuring the X550 case temperature are provided in Section 14.10.1.

14.10.1 Case Temperature Measurements

Maintain the X550 Tcase at or below the maximum case temperatures listed in Table 14-2 to ensure functionality and reliability. Special care is required when measuring the Tcase temperature to ensure an accurate temperature measurement. Use the following guidelines when making Tcase measurements:

- Measure the surface temperature of the case in the geometric center of the case top.
- Calibrate the thermocouples used to measure Tcase before making temperature measurements.
- Use 36-gauge (maximum) K-type thermocouples.

Care must be taken to avoid introducing errors into the measurements when measuring a surface temperature that is a different temperature from the surrounding local ambient air. Measurement errors can be due to a poor thermal contact between the thermocouple junction and the surface of the package, heat loss by radiation, convection, conduction through thermocouple leads, and/or contact between the thermocouple cement and the heat sink base (if used).

14.10.1.1 Attaching the Thermocouple (No Heat Sink)

The following approach is recommended to minimize measurement errors for attaching the thermocouple with no heat sink:

- Use 36-gauge or smaller-diameter K-type thermocouples.
- Ensure that the thermocouple has been properly calibrated.
- Attach the thermocouple bead or junction to the top surface of the package (case) in the center of the heat spreader using high thermal conductivity cements.

Note: It is critical that the entire thermocouple lead be butted tightly to the heat spreader.

Attach the thermocouple at a 0° angle if there is no interference with the thermocouple attach location or leads (see Figure 14-6). This is the preferred method and is recommended for use with packages without a heat sink.

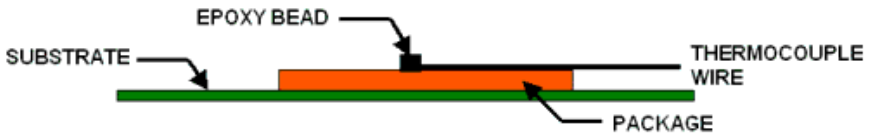


Figure 14-6. Technique for Measuring Tcase with 0° Angle Attachment and No Heat Sink

14.10.1.2 Attaching the Thermocouple (Heat Sink)

The following approach is recommended to minimize measurement errors for attaching the thermocouple with heat sink:

- Use 36-gauge or smaller diameter K-type thermocouples.
- Ensure that the thermocouple is properly calibrated.
- Attach the thermocouple bead or junction to the case's top surface in the geometric center using high thermal conductivity cement.

Note: It is critical that the entire thermocouple lead be butted tightly against the case.

- Attach the thermocouple at a 90° angle if there is no interference with the thermocouple attach location or leads (see Figure 14-7). This is the preferred method and is recommended for use with packages with heat sinks.
- For testing purposes, a hole (no larger than 0.150 inches in diameter) must be drilled vertically through the center of the heat sink to route the thermocouple wires out.
- Ensure there is no contact between the thermocouple cement and heat sink base. Any contact affects the thermocouple reading.

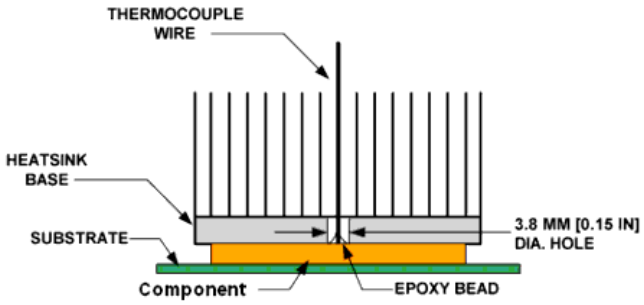


Figure 14-7. Technique for Measuring Tcase with 90° Angle Attachment

14.11 Conclusion

Increasingly complex systems require better power dissipation. Heat can be dissipated using improved system cooling, selective use of ducts, passive or active heat sinks, or any combination.

The simplest and most cost-effective method is to improve the inherent system cooling characteristics through careful design and placement of fans, vents, and ducts. When additional cooling is required, thermal enhancements can be implemented in conjunction with enhanced system cooling. The size of the fan or heat sink can be varied to balance size and space constraints with acoustic noise.

This described the conditions and requirements to properly design a cooling solution for systems implementing the X550. Properly designed solutions provide adequate cooling to maintain the X550 case temperature at or below those listed in Table 14-2. Ideally, this is accomplished by providing a low local ambient temperature and creating a minimal thermal resistance to that local ambient temperature. Alternatively, heat sinks might be required if case temperatures exceed those listed in Table 14-2.

By maintaining the X550 case temperature at or below those recommended in this section, the X550 will function properly and reliably.

Use this section to understand the X550 thermal characteristics and compare them to your system environment. Measure the X550 case temperatures to determine the best thermal solution for your design.

14.12 Heat Sink and Attach Suppliers

Part	Part Number	Supplier
Extruded Al Heat sink + Clip + PCM45 (TIM) Assembly	Generated specific to customer numbering scheme	Cooler Master
PCM45 Series	PCM45F	Honeywell

14.13 PCB Guidelines

The following general PCB design guidelines are recommended to maximize thermal performance of FCBGA packages:

- When connecting ground (thermal) vias to the ground planes, do not use thermal-relief patterns.
- Thermal-relief patterns are designed to limit heat transfer between vias and the copper planes, thus constricting the heat flow path from the component to the ground planes in the PCB.
- As board temperature also has an effect on the thermal performance of the package, avoid placing the X550 adjacent to high-power dissipation devices.
- If airflow exists, locate the components in the mainstream of the airflow path for maximum thermal performance. Avoid placing the components downstream, behind larger devices or devices with heat sinks that obstruct the air flow or supply excessively heated air.

Note: The previous information is provided as a general guideline to help maximize the thermal performance of the components.

Chapter 15 Diagnostics

15.1 JTAG Test Mode Description

The X550 includes a JTAG (TAP) port compliant with the *IEEE Standard Test Access Port and Boundary Scan Architecture 1149.1 Specification*. The TAP controller is accessed serially through five dedicated pins: TCK, TMS, TDI, TDO, and TRST_N. TMS, TDI, and TDO operate synchronously with TCK. TCK is independent of all other device clocks.

This interface can be used for test and debug purposes. System board interconnects can be DC tested using the boundary scan logic in pads. Table 15-1 shows TAP controller related pin descriptions. Table 15-2 describes the TAP instructions supported.

Table 15-1. TAP Controller Pins

Signal	I/O	Description
TCK	In	Test clock input for the test logic defined by IEEE1149.1. If utilizing JTAG, connect to this signal ground through a 1 KΩ pull-down resistor.
TDI	In	Test Data Input. Serial test instructions and data are received by the test logic at this pin. If utilizing JTAG, connect this signal to VCC33 through a 1 KΩ pull-up resistor.
TDO	O/D	Test Data Output. The serial output for the test instructions and data from the test logic defined in IEEE1149.1. If utilizing JTAG, connect this signal to VCC33 through a 1 KΩ pull-up resistor.
TMS	In	Test Mode Select input. The signal received at JTMS is decoded by the TAP controller to control test operations.
TRST_N	In	JTAG Reset Input. Active low reset for the JTAG port.

Table 15-2. Main TAP Instructions Supported

Instruction	Description	Comment
BYPASS	The BYPASS command selects the Bypass Register, a single bit register connected between TDI and TDO pins. This allows more rapid movement of test data to and from other components in the system.	IEEE 1149.1 Std. Instruction
EXTEST	The EXTEST Instruction allows circuitry or wiring external to the devices to be tested. Boundary-scan Register Cells at outputs are used to apply stimulus while Boundary-scan cells at input pins are used to capture data.	IEEE 1149.1 Std. Instruction
SAMPLE / PRELOAD	The SAMPLE/PRELOAD instruction is used to allow scanning of the boundary scan register without causing interference to the normal operation of the device. Two functions can be performed by use of the Sample/Preload instruction. • SAMPLE – allows a snapshot of the data flowing into and out of a device to be taken without affecting the normal operation of the device. • PRELOAD – allows an initial pattern to be placed into the boundary scan register cells. This allows initial known data to be present prior to the selection of another boundary-scan test operation.	IEEE 1149.1 Std. Instruction

Table 15-2. Main TAP Instructions Supported [continued]

Instruction	Description	Comment
IDCODE	The IDCODE instruction is forced into the parallel output latches of the instruction register during the Test-Logic-Reset TAP state. This allows the device identification register to be selected by manipulation of the broadcast TMS and TCK signals for testing purposes, as well as by a conventional instruction register scan operation. The ID code value for the X550 A0 is 0x11562013 (Intel's Vendor ID = 0x09, Device ID = 0x1562, Rev ID = 0x0) The ID code value for the X550 B0 is 0x21562013 (Intel's Vendor ID = 0x09, Device ID = 0x1562, Rev ID = 0x1)	IEEE 1149.1 Std. Instruction
HIGHZ	The HIGHZ instruction is used to force all outputs of the device (except TDO) into a high impedance state. This instruction selects the Bypass Register to be connected between TDI and TDO in the Shift-DR controller state.	IEEE 1149.1 Std. Instruction

15.2 MAC Loopback Operations

Loopback operations are supported by the X550 to assist with system and the X550 debug. Loopback operation can be used to test transmit and receive aspects of software drivers, as well as to verify electrical integrity of the connections between the X550 and the system (such as PCIe bus connections, etc.).

15.2.1 Tx->Rx MAC Loopback

This loopback is closed on the internal XGMII interface of the MAC core.

To configure the X550 for Tx->Rx loopback operation:

- Disable auto-negotiation in PHY register bit 7.0.C.
- Operate only at 10 GbE speed while no link partner is present. For other speeds, establish the link with a link partner at the desired speed while performing Tx -> Rx MAC loopback.
- In the MACC register, set the *FLU* bit to 1b to force link up.
- In the HLREG0 register, set the *LPBK* bit to 1b.

15.2.2 Rx->Tx MAC Loopback

This loopback is closed in the internal XGMII interface.

To configure the X550 for Rx->Tx loopback operation, the *MAC_RX2TX_LPBK_EN* bit in the MACC register should be set to 1b.

For loopback to be functional a functional link (with the partner) should be achieved (sync and alignment).

Link configuration should be done as in regular functional mode (see Section 4.6.3). All link modes can be configured.

Loopback limitation notes:

- Short preamble with minimal IPG is not supported with loopback operation.
- Transmitted data might violate the minimum IPG specification requirements.

15.3 NVM Recovery Mode

NVM recovery mode is intended to recover from a misconfiguration, which can be caused by an interrupted firmware update, power failure, host software access, or interrupted BMC configuration access. This misconfiguration prevents firmware from executing successful initialization flows. These flows enable the software device driver/BMC to fix the misconfiguration and cause firmware to initialize the correct data path.

For more information on Recovery Mode, see the *Recovery Mode in Intel® Ethernet Devices/Adapters Application Note* (Doc ID: 338102).

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

Chapter 16 Glossary and Acronyms

Term	Definition
1’s complement	A system known as ones' complement can be used to represent negative numbers in a binary system. The ones' complement form of a negative binary number is the bitwise NOT applied to it.
1000BASE-BX	1000BASE-BX is the PICMG 3.1 electrical specification for transmission of 1 Gb/s Ethernet or 1 Gb/s fibre channel encoded data over the backplane.
1000BASE-CX	1000BASE-X over specialty shielded 150 W balanced copper jumper cable assemblies as specified in IEEE 802.3 Clause 39.
1000BASE-T	1000BASE-T is the specification for 1 Gb/s Ethernet over category 5e twisted pair cables as defined in IEEE 802.3 clause 40.
2’s complement	A system of two's-complement arithmetic represents negative integers by counting backwards and wrapping around. Any number whose left-most bit is 1 is considered negative.
AAD	Additional Authentication Data input, which is authenticated data that must be left un-encrypted.
ACK	Acknowledgment
ACPI	Advanced Configuration and Power Interface — ACPI reset is also known as D3hot-D0 transition.
AEN	Address Enable
AER	Advanced Error Reporting
AFE	Analog Front End
AH	IP Authentication Header — An IPsec header providing authentication capabilities defined in RFC 2402 For an example of an AH packet diagram see below: • Next Header: Identifies the protocol of the transferred data. • Payload Length: Size of AH packet. • RESERVED: Reserved for future use (all zero until then). • Security Parameters Index (SPI): Identifies the security parameters, which, in combination with the IP Address, then identify the Security Association implemented with this packet. • Sequence Number: Monotonically increasing number, used to prevent replay attacks. Authentication Data: Contains the integrity check value (ICV) necessary to authenticate the packet; it may contain padding.
AMT	Active Management Technology (Intel® AMT)
AN	Auto-negotiation
AN	Association Number
APIC	Advanced Programming Interrupt Controller
APM	Advanced Power Management
APT	Advanced Pass-Through mode
ARI	Alternative Routing ID capability structure — This is a new capability that allows an interpretation of the Device and Function fields as a single identification of a function within the bus.
ARI	Alternate Requester ID Interpretation
ARP	Address Resolution Protocol
Backbone	A bus shared by many clients for example a management backbone or a host backbone
BAR	Base Address Register
BDF	Bus/Device/Function
BER	Bit Error Rate
BIOS	Basic Input/Output System
BIST	Built-In Self Test

Term	Definition
BKM	Best Known Method
BMC	Baseboard Management Controller
BME	Bus Master Enable
BT	Byte Time
BW (0f b/w)	Bandwidth
BWG	Bandwidth Group
Byte alignment	Implies that the physical addresses can be odd or even. Examples: 0FECBD9A1h, 02345ADC6h.
CAM	Content Addressable Memory
CCS	Current Cipher Suite
Ciphertext	Encrypted data, whose length is exactly that of the plaintext.
CNM	Congestion Notification Message
Concurrency	The concurrent (simultaneous) execution of multiple interacting computational tasks. These tasks may be implemented as separate programs, or as a set of processes or threads created by a single program.
Core	Network Interface Registers
Corner case	Is a problem or situation that occurs only outside of normal operating parameters — specifically one that manifests itself when multiple environmental variables or conditions are simultaneously at extreme levels. For example, a computer server may be unreliable, but only with the maximum complement of 64 processors, 512 GB of memory, and over 10,000 signed-on users. From Wiki.
CPID	Congestion Point Identifier – which should include the congestion point Ethernet MAC Address, as well as a local identifier for the local congestion entity, usually a queue in the switch.
CRC	Cyclic Redundancy Check A cyclic redundancy check (CRC) is a type of function that takes as input a data stream of unlimited length and produces as output a value of a certain fixed size. The term CRC is often used to denote either the function or the function's output. A CRC can be used in the same way as a checksum to detect accidental alteration of data during transmission or storage. CRCs are popular because they are simple to implement in binary hardware, are easy to analyze mathematically, and are particularly good at detecting common errors caused by noise in transmission channels. From Wiki
CRS	Carrier Sense Indication
CSMA/CD	802.3 Carrier Sense Multiple Access / Collision Domain Ethernet LCI-2 Interface to an external LAN Connected Device to provide wired LAN connectivity.
CSR	Control/Status Register
D0a D0 Active	Active fully operational state. Once memory space is enabled all internal clocks are activated and the LAN Controller enters an active state.
D0u D0 Uninitialized	The D0u state is a low-power state used after PCI Reset (SPXB Reset) is de-asserted following power-up (cold or warm), or on D3 exit.
D3Cold	Power Off. If Vcc is removed from the device, all of its PCI functions transition immediately to D3 cold. When Power is restored a PCI Reset must be asserted.
D3Hot	In D3 the LAN Controller only responds to PCI configuration accesses and does not generate master cycles.
DA	Destination Address
DAC	Digital to Analog Converter
DAC	Dual Address Cycle messages
Data Frame	FC Frames that carry read or write data.
DBU	Data Buffer Unit
DCA	Direct Cache Access

Term	Definition
DCB	Data Center Ethernet
DCX	DCB Configuration Exchange protocol
DDP	Direct Data placement
DEI	Drop Eligible Indicator (802.1Q)
DFT	Testability
DFX	Design for *
DHCP	Dynamic Host Configuration Protocol (protocol for automating the configuration of computers that use TCP/IP).
DLLP	Data Link Layer Packet /PCIe
DMA	Direct Memory Access
DMTF NC-SI	Distributed Management Task Force BMC-NIC interconnect for management.
DQ	Descriptor Queue
Dr	Internal Power management state when minimal function is provided (WoL, Manageability).
DSP	Digital Signal Processor
DUT	Device Under Test
DWORD (Double-Word) alignment	Implies that the physical addresses may only be aligned on 4-byte boundaries, In other words, the last nibble of the address may only end in 0, 4, 8, or Ch. For example, 0FECBD9A8h.
E-SOF	FCoE Start of Frame
EAPOL	Extensible Authentication Protocol over LAN
EAS	External Architecture Specification.
ECC	Error Correction Coding
ECRC	End to End CRC
EDB	End Data Bit
EEPROM	Electrically Erasable Programmable Memory. A non-volatile memory located on the LAN controller that is directly accessible from the host.
EHS	External Heat Sink
EOP	End-Of-Packet; when set indicates the last descriptor making up the packet.
EP	End Point
ESN	Extended Sequence Number
ESP	IP Encapsulating Security Payload — An IPsec header providing encryption and authentication capabilities defined in RFC 4303. The Encapsulating Security Payload (ESP) extension header provides origin authenticity, integrity, and confidentiality protection of a packet. ESP also supports encryption-only and authentication-only configurations, but using encryption without authentication is strongly discouraged. Unlike the AH header, the IP packet header is not accounted for. ESP operates directly on top of IP, using IP protocol number 50. ESP fields: • Security Parameters Index (SPI): See AH • Sequence Number: See AH • Payload Data: See AH • Padding: Used with some block ciphers to pad the data to the full length of a block. • Pad Length: Size of padding in bytes. • Next Header: Identifies the protocol of the transferred data. • Authentication Data: Contains the data used to authenticate the packet.
EUI	IEEE defined 64-bit Extended Unique Identifier
Extension Header	IPv6 protocol.

Term	Definition
Fail-over	Fail-over is the ability to detect that the LAN connection on one port is lost, and enable the other port for traffic.
FC	Fiber Channel
FC	Flow Control
FC Exchange	Complete Fiber Channel Read or Fiber Channel Write flow. It starts with the read or write requests by the initiator (the host system) till the completion indication from the target (the remote disk).
FC Frame	Fiber Channel Frames are the smallest units sent between the initiator and the target. The FC-FS-2 specification defined the maximum frame size as 2112 bytes. Each Fiber Channel frame includes an FC header and optional FC payload. It can also may include Extended headers and FC optional headers. Extended headers are not expected in FCoE network and FC optional headers may not be used as well.
FC Sequence	A Fiber Channel Exchange is composed of multiple Fiber Channel sequences. Fiber Channel Sequence can be a single or multiple frames that are sent by the initiator or the target. Each FC Sequence has a unique “Sequence ID”.
FCoE	Fiber Channel over Ethernet
FCP_RSP Frame	Fiber Channel control Frames that are sent from the target to the initiator which defines the completion of an FC read or write exchange.
FCS	Frame Check Sequence of Ethernet frames (also known as CRC)
FEC	Forward Error Correction
FEXT	Far End Crosstalk
Firmware (FW)	Embedded code on the LAN controller that is responsible for the implementation of the NC-SI protocol and pass-through functionality.
FLR	Function level reset An OS in a VM must have complete control over a device, including its initialization, without interfering with the rest of the functions.
FML	Fast Management Link
Fragment Header	An IPv6 extension Header
Frame	A unit composed of headers, data and footers that are sent or received by a device. Same as a Packet
FSM	Finite State Machine
FTS	Fast Training Sequence
GbE	Gigabit Ethernet (IEEE 802.3z-1998)
GMRP	GARP Multicast Registration Protocol (Cisco)
GPIO	General Purpose I/O
GSP	Group Strict Priority
HBA	Host Bus Adapters
Host Interface	RAM on the LAN controller that is shared between the firmware and the host. RAM is used to pass commands from the host to firmware and responses from the firmware to the host.
HPC	High-Performance Computing
HT core option	Hyper Thread Intel's trademark for implementation of the simultaneous multi-threading technology on the Pentium 4 micro architecture. It is a more advanced form of Super-threading that debuted on the Intel Xeon processors and was later added to Pentium 4 processors. The technology improves processor performance under certain workloads by providing useful work for execution units that would otherwise be idle, for example during a cache miss. A Pentium 4 with Hyper-Threading enabled is treated by the operating system as two processors instead of one. From Wiki
IANA	Internet Assigned Number Authority
IDS	Intrusion Detection Systems
IFCS	Insert Frame Check Sequence of Ethernet frames
IFS	Inter Frame Spacing

Term	Definition
IKE	Internet Key Exchange
IOAT	I/O Acceleration Technology
IOH	I/O Hub
IOV	Input Output Virtualization
IOV mode	Operating through an IOVM or IOVI
IOVI	I/O Virtual Intermediary: A special virtual machine that owns the physical device and is responsible for the configuration of the physical device (also known as IOVM).
IOVM	I/O Virtual Machine: A special virtual machine that owns the physical device and is responsible for the configuration of the physical device (also known as IOVI).
IP — CPMP	Carrier Performance Measurement Plan
IP Tunneling	IP tunneling is the process of embedding one IP packet inside of another, for the purpose of simulating a physical connection between two remote networks across an intermediate network. IP tunnels are often used in conjunction with IPsec protocol to create a VPN between two or more remote networks across a “hostile” network such as the Internet.
IPC	Inter Processor Communication
IPG	Inter Packet Gap
IPsec	IP Security is a suite of protocols for securing Internet Protocol (IP) communications by authenticating and/or encrypting each IP packet in a data stream. IPsec also includes protocols for cryptographic key establishment. IPsec is implemented by a set of cryptographic protocols for (1) securing packet flows and (2) internet key exchange. There are two families of key exchange protocols. The IP security architecture uses the concept of a security association as the basis for building security functions into IP. A security association is simply the bundle of algorithms and parameters (such as keys) that is being used to encrypt a particular flow. The actual choice of algorithm is left up to the users. A security parameter index (SPI) is provided along with the destination address to allow the security association for a packet to be looked up. For multicast, therefore, a security association is provided for the group, and is duplicated across all authorized receivers of the group. There may be more than one security association for a group, using different SPIs, thereby allowing multiple levels and sets of security within a group. Indeed, each sender can have multiple security associations, allowing authentication, since a receiver can only know that someone knowing the keys sent the data. Note that the standard does not describe how the association is chosen and duplicated across the group; it is assumed that a responsible party will make the choice. From Wiki
iSCSI	Internet SCSI (iSCSI) is a network protocol standard, officially ratified on 2003-02-11 by the Internet Engineering Task Force, that allows the use of the SCSI protocol over TCP/IP networks. iSCSI is a transport layer protocol in the SCSI-3 specifications framework. Other protocols in the transport layer include SCSI Parallel Interface (SPI), Serial Attached SCSI (SAS) and Fibre Channel. From Wiki.
ISR	Interrupt Service Routine
ITR	Interrupt Throttling
IV	Integrity Value
IV	Initialization Vector
IV	Initial Value
KaY	Key agreement entity (KaY – in 802.1AE specification terminology) i.e. control and access the off loading engine (SecY in 802.1AE specification terminology)
KVM	Keyboard/Video/Mouse
LACP	Link Aggregation Control Protocol
LAN auxiliary Power-Up	The event of connecting the LAN controller to a power source (occurs even before system power-up).
landing Zone requirements	General targets for the product.
LDPC	Low-Density Parity-check. Coding used in 802.3an (10GBASET) to protect transmitted data.

Term	Definition
LF	Local Fault
LLC header	802.2 defines a special header that includes a SNAP (subnetwork access protocol) header + EtherType. Some protocols, particularly those designed for the OSI networking stack, operate directly on top of 802.2 LLC, which provides both datagram and connection-oriented network services. This 802.2 header is currently embedded in modern 802.3 frames (Ethernet II frames, aka. DIX frames). The LLC header includes two additional eight-bit address fields, called service access points or SAPs in OSI terminology; when both source and destination SAP are set to the value 0xAA, the SNAP service is requested. The SNAP header + EtherType allows EtherType values to be used with all IEEE 802 protocols, as well as supporting private protocol ID spaces. In IEEE 802.3x-1997, the IEEE Ethernet standard was changed to explicitly allow the use of the 16-bit field after the Ethernet MAC Addresses to be used as a length field or a type field. This definition is from Wiki
LLDP	Link Layer Discovery Protocol
LLINT	Low Latency Interrupt
Local Traffic	In a virtual environment traffic between virtual machines.
LOM	LAN on Motherboard
LP	Link Partner
LS	Least significant / Lowest order (for example: LS bit = Least significant bit)
LSC	Link Status Change
LSO	Large Send Offload (also known as TSO)
LSP	Link Strict Priority
LTSSM	Link Training and Status State Machine Defined in the PCIe specs.
MAC	Media Access Control
MAUI	Multi Speed Attachment Unit Interface
MCH	Memory Controller Hub
MDC	Management Data Clock over MDC/MDIO lines.
MDI	Media Dependent Interface
MDIO	Management Data Input/Output Interface over MDC/MDIO lines.
MFVC	Multi-Function Virtual Channel Capability structure
MIB	Management Interface Bus
MIFS/MIPG	Minimum Inter Frame Spacing/Minimum Inter Packet Gap.
MMW	Maximum Memory Window
Mod / Modulo	In computing, the modulo operation finds the remainder of division of one number by another.
MPA	Marker PDU Aligned Framing for TCP.
MRQC	Multiple Receive Queues Command register
MS	Most significant / Highest order (for example: MS byte = Most significant byte)
MSFT	Microsoft®
MSI	Message Signaled Interrupt
MSS	Maximum Segment Size
MTA	Multicast Table Array
MTU	Maximum Transmission Unit
NACK	Negative Acknowledgment
native mode	Used for GPIO pin that is set to be controlled by the internal logic rather than by software.
NC-SI	Network Controller - Sideband Interface

Term	Definition
NEXT	Near End Crosstalk
NFS	Network File Server
NFTS	Number of Fast Training Signals
NIC	Network Interface Controller
Nonce	96-bits initialization vector used by the AES-128 engine, which is distinct for each invocation of the encryption operation for a fixed key. It is formed by the AES-128 SALT field stored for that IPsec flow in the Tx SA Table, appended with the Initialization Vector (IV) field included in the IPsec packet:
NOS	Network Operating System
NPRD	Non-Posted Request Data
NRZ	Non-return-to-Zero signaling
NSE	No Snoop Enable
NTL	No Touch Leakage
NTP	Network Time Protocol
NVM	Non Volatile Memory
OEM	Original Equipment manufacturer
Packet	A unit composed of headers, data and footers that are sent or received by a device. Also known as a frame.
Pass Filters	Needs Definition Packets that match this type of filter continue on to their destination?
PB	Packet Buffer
PBA	Pending Bit Array (in MSI-X context)
PBA	Printed Board Assembly (in NVM or board context). The nine-digit number used for Intel-manufactured adapter cards.
PCS	Physical Coding Sub-layer
PDU	Protocol Data Units
PF	Physical Function (in a virtualization context).
PFC	Priority Flow Control
PHY	Physical Layer Device
Plaintext	Data to be both authenticated and encrypted.
PMA	Physical Medium Attachment
PMC	Power Management Capabilities
PMD	Physical Medium Dependent
PME	Power Management Event
PN	Packet Number
Pool	Virtual ports
Power State D0a	Active fully operational state. Once memory space is enabled all internal clocks are activated and the LAN Controller enters an active state.
Power State D0u	The D0u state is a low-power state used after SPXB Reset is de-asserted following power-up (cold or warm), or on D3 exit.
Power State D3Cold	A Power down state with the PCI also in a power down state.
Power State D3Hot	A Power down state with the PCI continuing to receive a proper power supply.
Power State Dr	Device state when PCIe reset is asserted.

Term	Definition
Power State Sx	LAN Connected Device: SMBus Active and PCI Powered down.
PPM	Part Per Million
PRBS	Pseudo-Random Binary Sequence
PT	Pass-Through
PTP	Precision Time Protocol
QoS	Quality of Service
QWORD (Quad-Word) alignment	Implies that the physical addresses may only be aligned on 8byte boundaries; i.e., the last nibble of the address may only end in 0, or 8. For example, 0FECBD9A8h.
RDMA	Remote Direct Memory Access
RDMAP	Remote Direct Memory Access Protocol
Receive latency	Measured from packet reception from the wire and until the descriptor is updated on PCIe.
Relax ordering	When the strict order of packets is not required, the device can send packets in an order that allows for less power consumption and greater CPU efficiency.
RID	Requester ID
RLT	Rate-Limited flag bit
RMCP	Remote Management and Control Protocol (Distributed Management Task Force)
RMII	Reduced Media Independent Interface (Reduced MII)
RMON statistics	Remote Network Monitoring or Remote Monitoring
RPC header	Remote Procedure Call
RS	Rate Scheduler
RSC	Receive Side Coalescing coalesces incoming TCP/IP (and potentially UDP/IP) packets into larger receive segments.
RSS	Receive-Side Scaling is a mechanism to distribute received packets into several descriptor queues. Software then assigns each queue to a different processor, therefore sharing the load of packet processing among several processors.
RSTD	Reset Sequence Done
RSTI	Reset Sequence in Process
RT	DCB (formerly ReedTown)
Rx, RX	Receive
SA	Security Association (in IPsec context)
SA	Source Address (in Ethernet frame context)
SAC	Single Address Cycle (SAC) messages
SAK	Security Associations Key
salt	In cryptography, a salt consists of random bits used as one of the inputs to a key derivation function. Sometimes the initialization vector, a previously generated (preferably random) value, is used as a salt. The other input is usually a password or passphrase. The output of the key derivation function is often stored as the encrypted version of the password. A salt value can also be used as a key for use in a cipher or other cryptographic algorithm. A salt value is typically used in a hash function. from Wiki
SAN	Storage Area Networks
SAP	Service Access Point — An identifying label for network endpoints used in OSI networking.
SC	Secure Channel — Authentication and key exchange.
SC	Secure Channel (SC) — A security relationship used to provide security guarantees for frames transmitted from one member of a CA to the others. An SC is supported by a sequence of SAs thus allowing the periodic use of fresh keys without terminating the relationship.

Term	Definition
SCI	Secure Channel Identifier — A globally unique identifier for a secure channel, comprising a globally unique Ethernet MAC Address and a Port Identifier, unique within the system allocated that address.
SCL signal	SM Bus Clock
SCSI	Small Computer System Interface is a set of standards for physically connecting and transferring data between computers and peripheral devices. The SCSI standards define commands, protocols, and electrical and optical interfaces. SCSI is most commonly used for hard disks and tape drives, but it can connect a wide range of other devices, including scanners, and optical drives (CD, DVD, etc. From Wiki.
SCTP	Stream Control Transmission Protocol
SDA signal	SM Bus Data
SDP	Software-Definable Pins
SecY	802.1AE specification terminology Security entity
Segment	Subsections of a packet.
SerDes	Serializer and De-Serializer Circuit
SFD	Start Frame Delimiter
SFI	Serial Flash Interface
SGMII	Serialized Gigabit Media Independent Interface
SKU	Subsets of features of a chip that can be disabled for marketing purposes
SMB	Semaphore Bit
SMBus	System Management Bus — A bus that carries various manageability components, including the LAN controller, BIOS, sensors and remote-control devices.
SN	Sequence Number — Contains a counter value that increases by one for each Ethernet frame sent.
SNAP	Subnetwork Access Protocol
SNMP	Standard Network Management Protocol
SoL	Serial Over LAN — A mechanism that enables the input and output of the serial port of a managed system to be redirected via an IPMI (Internet Protocol Multicast Initiative) session over IP.
SPD	Smart Power Down
SPI	The Security Parameter Index is an identification tag added to the header while using IPsec for tunneling the IP traffic. This tag helps the kernel discern between two traffic streams where different encryption rules and algorithms may be in use. The SPI (as per RFC 2401) is an essential part of an IPsec SA (Security Association) because it enables the receiving system to select the SA under which a received packet is processed. An SPI has only local significance, since is defined by the creator of the SA; an SPI is generally viewed as an opaque bit string. However, the creator of an SA may interpret the bits in an SPI to facilitate local processing. from Wikipedia
Spoofing	In computer networking, the term IP Address spoofing is the creation of IP packets with a forged (spoofed) source IP Address with the purpose to conceal the identity of the sender or impersonating another computing system. IP stands for Internet Protocol. from Wiki
SPXB interface	PCI Express Backbone
SR-IOV	PCI-SIG single-root I/O Virtualization initiative
SW Switch acceleration mode	Central management of the networking resources by an IOVM or by the VMM. Also known as VMDq2 mode.
SWIZZLE	To convert external names, array indices, or references within a data structure into address pointers when the data structure is brought into main memory from external storage (also called pointer swizzling);
Sx	LAN Connected Device: SMBus Active and PCI Powered down.
SYN Attack	A SYN attack is a form of denial-of-service attack in which an attacker sends a succession of SYN (synchronize) requests to a target's system.

Term	Definition
TC	Traffic Class
TCI	For 802.1q, Tag Header field Tag Control Information (TCI); 2 octets.
TCO	Total Cost of Ownership (Management)
TCP/IP	Transmission Control Protocol/Internet Protocol
TDESC	Transmit Descriptor
TDP	Total Device Power
TDR	Time Domain Reflectometry
TFCS	Transmit Flow Control Status
TLP	Transaction layer Packets
ToS	Type of Service
TPID	For 802.1q, Tag Header field Tag Protocol Identifier; 2 octets.
TPPAC	Transmit Packet Plane Arbitration Control
Transmit latency	Measured from Tail update until the packet is transmitted on the wire. It is assumed that a single packet is submitted for this traffic class and its latency is then measured in presence of traffic belonging to other traffic classes.
TS	Time Stamp
TSO	TCP or Transmit Segmentation offload — A mode in which a large TCP/UDP I/O is handled to the device and the device segments it to L2 packets according to the requested MSS.
TSS	Transmit Side Scaling
Tx, TX	Transmit
UBWG	User Bandwidth Group
ULP	Upper Layer Protocol
UP	User Priority
UR	Error Reporting Unsupported Request Error
VF	Virtual Function – A part of a PF assigned to a VI
VI	Virtual Image – A virtual machine to which a part of the I/O resources is assigned. Also known as a VM.
VM	Virtual Machine
VMDq2	SW switch acceleration mode — Central management of the networking resources by an IOVM or by the VMM. Virtual Machine Devices queue (VMDq) is a mechanism to share I/O resources among several consumers. For example, in a virtual system, multiple OSs 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 OS may 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. VMDs (Virtual Machine Devices) are designed to reduce the burden of VMM by making certain functions of an I/O device shared and thus can be accessed directly from each guest OS or Virtual Machine (VM). From Nahum
VMM	Virtual Machine Monitor
VPD	Vital Product Data (PCI protocol).
VT	Virtualization
WB	Write Back
WC	Worst Case
WoL	Wake-on-LAN Now called APM Wake-up or Advanced power management Wake-up.

Term	Definition
WORD alignment	Implies that physical addresses must be aligned on even boundaries; i.e., the last nibble of the address may only end in 0, 2, 4, 6, 8, Ah, Ch, or Eh. For example, 0FECBD9A2h.
WRR	Weighted Round-Robin
WSP	Weighted Strict Priority
XAUI	10 Gigabit Attachment Unit Interface
XFI	Serial Interface that can connect to other devices supporting XFI
XFP	10 Gigabit Small Form Factor Pluggable modules
XGMII	10 Gigabit Media Independent Interface
XGXS	XGMII Extender Sub layer
XMT	Transmit

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

Appendix A Packet Formats

A.1 Legacy Packet Formats

A.1.1 ARP Packet Formats

A.1.1.1 ARP Request Packet

Offset	# of Bytes	Field	Value	Action
0	6	Destination Address		Compare
6	6	Source Address		Stored
12	E=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore
12 + E	S=(0/4)	Possible VLAN Tag		Stored
12 + E + S	2	Type	0x0806	Compare
14 + E + S	2	Hardware Type	0001	Compare
16 + E + S	2	Protocol Type	0x0800	Compare
18 + E + S	1	Hardware Size	06	Compare
19 + E + S	1	Protocol Address Length	04	Compare
20 + E + S	2	Operation	0001	Compare
22 + E + S	6	Sender Hardware Address	-	Stored
28 + E + S	4	Sender IP Address	-	Stored
32 + E + S	6	Target Hardware Address	-	Ignore
38 + E + S	4	Target IP Address	ARP IP Address	Compare

A.1.1.2 ARP Response Packet

Offset	# of Bytes	Field	Value
0	6	Destination Address	ARP Request Source Address
6	6	Source Address	Programmed from NVM or BMC
12	E=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	
12 + E	S=(0/4)	Possible VLAN Tag	From ARP Request
12 + E + S	2	Type	0x0806
14 + E + S	2	Hardware Type	0x0001
16 + E + S	2	Protocol Type	0x0800
18 + E + S	1	Hardware Size	0x06
19 + E + S	1	Protocol Address Length	0x04

Offset	# of Bytes	Field	Value
20 + E + S	2	Operation	0x0002
22 + E + S	6	Sender Hardware Address	Programmed from NVM or BMC
28 + E + S	4	Sender IP Address	Programmed from NVM or BMC
32 + E + S	6	Target Hardware Address	ARP Request Sender Hardware Address
38 + E + S	4	Target IP Address	ARP Request Sender IP Address

A.1.1.3 Gratuitous ARP Packet

Offset	# of Bytes	Field	Value
0	6	Destination Address	Broadcast address.
6	6	Source Address	
12	E=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	
12 + E	S=(0/4)	Possible VLAN Tag	
12 + E + S	D=(0/8)	Possible Length + LLC/SNAP Header	
12 + E + S + D	2	Type	0x0806
14 + E + S + D	2	Hardware Type	0x0001
16 + E + S + D	2	Protocol Type	0x0800
18 + E + S + D	1	Hardware Size	0x06
19 + E + S + D	1	Protocol Address Length	0x04
20 + E + S + D	2	Operation	0x0001
22 + E + S + D	6	Sender Hardware Address	
28 + E + S + D	4	Sender IP Address	
32 + E + S + D	6	Target Hardware Address	
38 + E + S + D	4	Target IP Address	

A.1.2 IP and TCP/UDP Headers for TSO

This section outlines the format and content for the IP, TCP and UDP headers. The X550 requires baseline information from the device driver to construct the appropriate header information during the segmentation process.

Header fields that are modified by the X550 are highlighted in the figures below.

Note: The IP header is first shown in the traditional (i.e. RFC 791) representation, and because byte and bit ordering is confusing in that representation, the IP header is also shown in Little Endian format. The actual data is fetched from memory in Little Endian format.

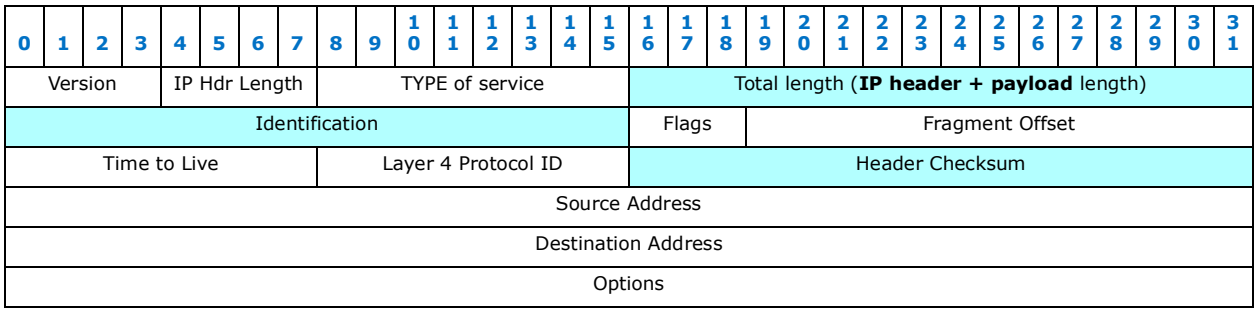


Figure A-1. IPv4 Header (Traditional Representation - Most Left Byte First On the Wire)

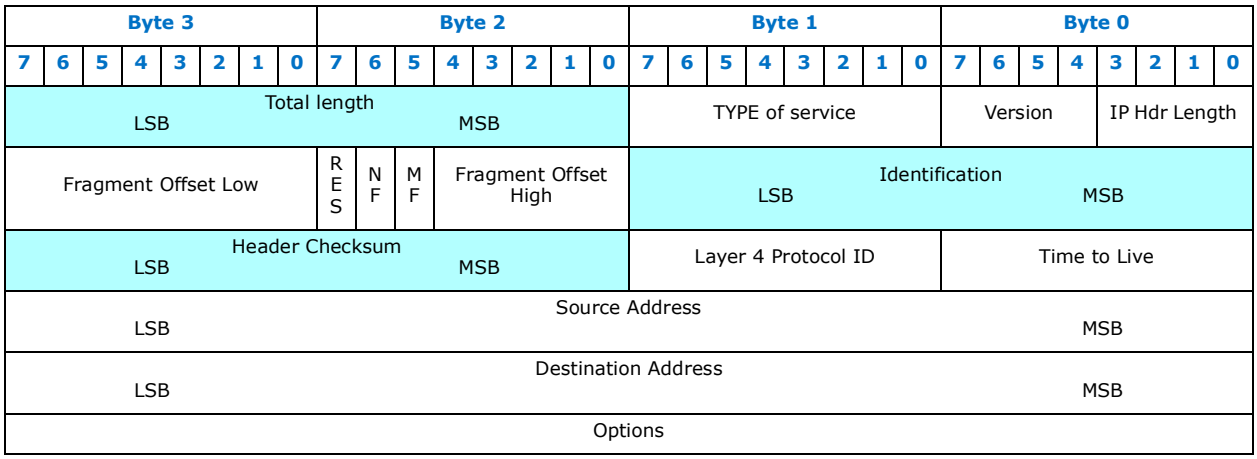


Figure A-2. IPv4 Header (Little Endian Order - Byte 0 First On the Wire)

Identification is increased on each packet.

Flags Field Definitions:

The Flags field is defined below. Note that hardware does not evaluate or change these bits.

- MF - More Fragments
- NF - No Fragments
- Reserved

The X550 does TCP segmentation, not IP Fragmentation. IP Fragmentation may occur in transit through a network's infrastructure.

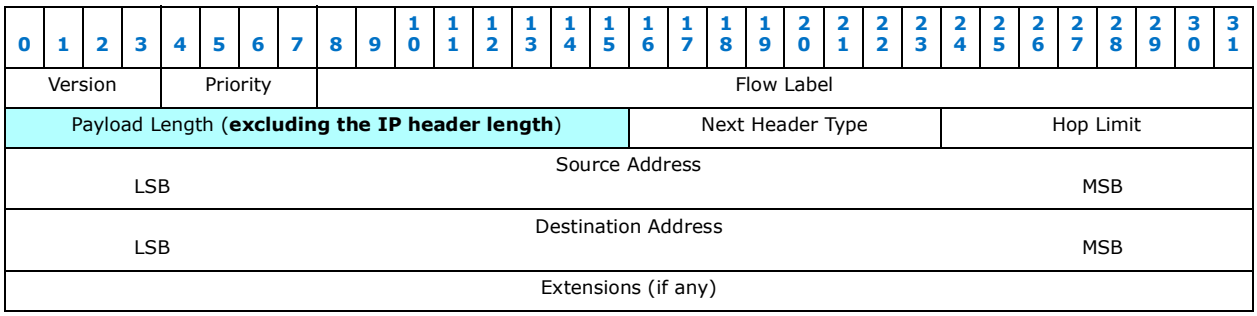


Figure A-3. IPv6 Header (Traditional Representation - Most Left Byte First On the Wire)

Byte 3								Byte 2								Byte 1								Byte 0							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Source Address LSB MSB																								Version				Priority			
Hop Limit								Next Header Type								Payload Length (**excluding the IP header length**) LSB MSB															
Source Address LSB MSB																															
Destination Address LSB MSB																															
Extensions																															

Figure A-4. IPv6 Header (Little Endian Order - Byte 0 First On the Wire)

A TCP or UDP frame uses a 16-bit wide one's complement checksum. The checksum word is computed on the outgoing TCP or UDP header and payload, and on the Pseudo Header. Details on checksum computations are provided in Section 7.2.4.6.

Note: TCP and UDP over IPv6 requires the use of checksum, where it is optional for UDP over IPv4.

Note: The TCP header is first shown in the traditional (i.e. RFC 793) representation, and because byte and bit ordering is confusing in that representation, the TCP header is also shown in Little Endian format. The actual data is fetched from memory in Little Endian format.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Source Port																Destination Port															
Sequence Number																															
Acknowledgment Number																															
TCP Header Length				Reserved						URG	ACK	PSH	RST	SYN	FIN	Window															
Checksum																Urgent Pointer															
Options																															

Figure A-5. TCP Header (Traditional Representation)

Byte 3								Byte 2								Byte 1								Byte 0							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Destination Port																Source Port															
Sequence Number LSB MSB																															
Acknowledgment Number																															
Window																RES	URG	ACK	PSH	RST	SYN	FIN	TCP Header Length					Reserved			
Urgent Pointer																Checksum															
Options																															

Figure A-6. TCP Header (Little Endian)

The TCP header is always a multiple of 32-bit words. TCP options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. All options are included in the checksum.

The checksum also covers a 96-bit pseudo header prefixed to the TCP Header (see Figure A-7 below). For IPv4 packets, this pseudo header contains the IP Source Address, the IP Destination Address, the IP Protocol field, and TCP Length. Software pre-calculates the partial pseudo header sum, that includes IPv4 SA, DA and protocol types, but NOT the TCP length, and stores this value into the TCP *Checksum* field of the packet. For both IPv4 and IPv6, hardware needs to factor in the TCP length to the software supplied pseudo header partial checksum.

Note: When calculating the TCP pseudo header, the byte ordering can be tricky. One common question is whether the *Protocol ID* field is added to the "lower" or "upper" byte of the 16-bit sum. The *Protocol ID* field should be added to the least significant byte (LSB) of the 16-bit pseudo header sum, where the most significant byte (MSB) of the 16-bit sum is the byte that corresponds to the first checksum byte out on the wire.

The *TCP Header Length* field is the TCP header length including option fields plus the data length in bytes, which is calculated by hardware on a frame-by-frame basis. The TCP Length does not count the 12 bytes of the pseudo header. The TCP length of the packet is determined by hardware as:

TCP Length = min(MSS,PAYLOADLEN) + L5_LEN

The two flags that may be modified are defined as:

- **PSH** — Receiver should pass this data to the app without delay.
- **FIN** — Sender is finished sending data.

The handling of these flags is described in Section 7.2.4.7.

"Payload" is normally MSS except for the last packet where it represents the remainder of the payload.

IPv4 Source Address		
IPv4 Destination Address		
Zero	Layer 4 Protocol ID	TCP/UDP Length

Figure A-7. TCP/UDP Pseudo Header Content for IPv4 (Traditional Representation)

IPv6 Source Address	
IPv6 Final Destination Address	
TCP/UDP Packet Length	
Zero	Next Header

Figure A-8. TCP/UDP Pseudo Header Content for IPv6 (Traditional Representation)

Notes: From RFC2460:

- If the IPv6 packet contains a Routing header, the Destination Address used in the pseudo-header is that of the final destination. At the originating node, that address is in the last element of the Routing header; at the recipient(s), that address is in the *Destination Address* field of the IPv6 header.
- The Next Header value in the pseudo-header identifies the upper-layer protocol (e.g., 6 for TCP, or 17 for UDP). It differs from the Next Header value in the IPv6 header if there are extension headers between the IPv6 header and the upper-layer header.

- The Upper-Layer Packet Length in the pseudo-header is the length of the upper-layer header and data (e.g., TCP header plus TCP data). Some upper-layer protocols carry their own length information (e.g., the *Length* field in the UDP header); for such protocols, that is the length used in the pseudo- header. Other protocols (such as TCP) do not carry their own length information, in which case the length used in the pseudo-header is the Payload Length from the IPv6 header, minus the length of any extension headers present between the IPv6 header and the upper-layer header.
  - Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error.

A type 0 Routing header has the following format:

Next Header	Hdr Ext Len	Routing Type "0"	Segments Left "n"
Reserved			
Address[1]			
Address[2]			
...			
Final Destination Address [n]			

Figure A-9. IPv6 Routing Header (Traditional Representation)

- **Next Header** — 8-bit selector. Identifies the type of header immediately following the Routing header. Uses the same values as the IPv4 protocol field [RFC-1700 et seq.].
- **Hdr Ext Len** — 8-bit unsigned integer. Length of the Routing header in 8-octet units, not including the first 8 octets. For the Type 0 Routing header, Hdr Ext Len is equal to two times the number of addresses in the header.
- **Routing Type** — 0.
- **Segments Left** — 8-bit unsigned integer. Number of route segments remaining (i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination). Equal to "n" at the source node.
- **Reserved** — 32-bit reserved field. Initialized to zero for transmission; ignored on reception.
- **Address[1...n]** — Vector of 128-bit addresses, numbered 1 to n.

The UDP header is always 8 bytes in size with no options.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Source Port																Destination Port															
Length																Checksum															

Figure A-10. UDP Header (Traditional Representation)

Byte 3								Byte 2								Byte 1								Byte 0							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Destination Port																Source Port															
Checksum																Length															

Figure A-11. UDP Header (Little Endian Order)

UDP pseudo header has the same format as the TCP pseudo header. The pseudo header prefixed to the UDP header contains the IPv4 source address, the IPv4 destination address, the IPv4 protocol field, and the UDP length (same as the TCP Length discussed above). This checksum procedure is the same as is used in TCP.

Unlike the TCP checksum, the UDP checksum is optional (for IPv4). Software must set the *TXSM* bit in the TCP/IP Context Transmit Descriptor to indicate that a UDP checksum should be inserted. Hardware does not overwrite the UDP checksum unless the *TXSM* bit is set.

A.1.3 Magic Packet

The Magic Packet is a broadcast frame, but could also be a Multicast or Unicast Ethernet MAC Addresses. The X550 accepts this packet if it matches any of its pre-programmed Ethernet MAC Addresses. Magic packet can be sent over a variety of connectionless protocols (usually UDP or IPX). The Magic packet pattern is composed of the following sequences:

- Synchronization stream composed of 6 bytes equal to 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
- Unique pattern composed of 16 times the end node Ethernet MAC Address. The X550 expect for the Ethernet MAC Address stored in the RAL[0] and RAH[0] registers.

The X550 looks for the synchronization pattern and the sequence of 16 Ethernet MAC Addresses. It does not check the packet content and the length of the header that precedes the magic pattern nor any data that follows it.

A.2 Packet Types for Packet Split Filtering

The following packet types are supported by the 'Packet Split' feature in the X550. This section describes the packets from the 'split-header' point of view. This means that when describing the different fields that are checked and compared, it emphasizes only the fields that are needed to calculate the header length. This document describes the checks that are done after the decision to pass the packet to the host memory was done.

Terminology:

- **Compare** — The field values are compared and must be exactly equal to the value specified in this document.
- **Checked** — The field values are checked for calculation (header length …).
- **Ignore** — The field values are ignored but the field is counted as part of the header.

A.2.1 Type 1.1: Ethernet (VLAN/SNAP) IP packets

A.2.1.1 Type 1.1: Ethernet, IP, Data

This type contains only Ethernet header and IPv4 header while the payload header of the IP is not IPv6/ TCP/UDP.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-Tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100 ****	Compare	
12+D+S	D=(0/8)	Possible Length + LLC/ SNAP Header		Compare	
12+D+S	2	Type	0800h	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	>0 or MF bit is set	Check	Check that the packet is fragmented.
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol		Ignore	Has no meaning if the packet is fragmented.
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	

A.2.1.2 Type 1.2: Ethernet (SNAP/VLAN), IPv4, UDP

This type contains only Ethernet header, IPv4 header, and UDP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100 ****	Check	
12+D+S	2	Type	0x0800	Compare	IP
IP Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	(xx00) 0x000	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x11	Compare	UDP header.
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
UDP Header					
34+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet.
36+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet.
38+D+S+N	2	Length	-	Ignore	
40+D+S+N	2	Checksum	-	Ignore	

In this case, the packet is split after (42+D+S+N) bytes.

A.2.1.3 Type 1.3: Ethernet (VLAN/SNAP) IPv4 TCP

This type contains only Ethernet header, IPv4 header, and TCP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100 ****	Check	
12+D+S	2	Type	0x0800	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x06	Compare	TCP header.
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
TCP Header					
34+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet.
36+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet.
38+D+S+N	4	Sequence number	-	Ignore	
42+D+S+N	4	Acknowledge number	-	Ignore	
46+D+S+N	1/2	Header Length		Check	
46.5+D+S+N	1.5	Different bits	-	Ignore	
48+D+S+N	2	Window size	-	Ignore	
50+D+S+N	2	TCP checksum	-	Ignore	
52+D+S+N	2	Urgent pointer	-	Ignore	
54+D+S+N	F	TCP options	-	Ignore	

In this case, the packet is split after (54+D+S+N+F) bytes.

- N = (IP HDR length -5) * 4.
- F = (TCP header length - 5) * 4.

A.2.1.4 Type 1.4: Ethernet IPv4 IPv6

A.2.1.4.1 IPv6 Header Options Processing

This type of processing looks at the next-header field and header length to determine the identity of the next-header processes, the IPv6 options, and it's length.

If the next header in the IPv6 header is equal to 0x00/0x2B/0x2C/0x3B/0x3c it means that the next header is an IPv6 option header and this is its structure:

Next Header (8 bit)	Header Len (8 bit)	
Option Header Parameters		

Header Len determines the length of the header while the next header field determines the identity of the next header (should be any IPv6 extension header for another IPv6 header option).

A.2.1.4.2 IPv6 Next Header Values

When parsing an IPv6 header, the X550 does not parse all possible extension headers. If there is an extension header that is not supported by the X550, the packet is treated as an unknown payload after the IPv6 header.

Value	Header Type
0x00	Hop by Hop
0x2B	Routing
0x2C	Fragment
0x3B	No next header (EOL)
0x3C	Destination option header

The next header in a fragment header is ignored and this extension header is expected to be the last header.

A.2.1.4.3 Type 1.4.1: Ethernet IPv4 IPv6 Data

This type contains only Ethernet header, IPv4 header, and IPv6 header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x29	Compare	IPv6
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
IPv6 Header					
34+D+S+N	1	Version/ Traffic Class	0x6X	Compare	Check IPv6.
35+D+S+N	3	Traffic Class/Flow Label	-	Ignore	
38+D+S+N	2	Payload Length	-	Ignore	
40+D+S+N	1	Next Header	IPv6 extension headers	Check	
41+D+S+N	1	Hop Limit	-	Ignore	
42+D+S+N	16	Source Address	-	Ignore	
48+D+S+N	16	Destination Address		Ignore	
74+D+S+N	B	Possible IPv6 Next Headers	-	Ignore	

In this case the packet is split after (74+D+S+N+B) bytes.

- N = (IP HDR length - 5) * 4.
- One of the extension headers of the IPv6 packets must be a “fragment header” for the packet to be parsed.

A.2.1.4.4 Type 1.4.2: Ethernet (VLAN/SNAP) IPv4 IPv6 UDP

This type contains only Ethernet header, IPv4 header, IPv6 header and UDP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x29	Compare	IPv6
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
IPv6 Header					
34+D+S+N	1	Version/ Traffic Class	0x6X	Compare	Check IPv6.
35+D+S+N	3	Traffic Class/Flow Label	-	Ignore	
38+D+S+N	2	Payload Length	-	Ignore	
40+D+S+N	1	Next Header	IPv6 extension header or 0x11	Check	IPv6 extension headers.
41+D+S+N	1	Hop Limit	-	Ignore	
42+D+S+N	16	Source Address	-	Ignore	
58+D+S+N	16	Destination Address		Ignore	
74+D+S+N	B	Possible IPv6 Next Headers	-	Ignore	
UDP Header					
74+D+S+N+B	2	Source Port	Not (0x801)	Check	Not NFS packet.
76+D+S+N+B	2	Destination Port	Not (0x801)	Check	Not NFS packet.
78+D+S+N+B	2	Length	-	Ignore	
80+D+S+N+B	2	Checksum	-	Ignore	

In this case the packet is split after (82+D+S+N+B) bytes.

N = (IP HDR length - 5) * 4.

A.2.1.4.5 Type 1.4.3: Ethernet (VLAN/SNAP) IPv4 IPv6 TCP

This type contain only Ethernet header, IPv4 header, IPv6 header and TCP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x2A	Compare	IPv6
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
IPv6 Header					
34+D+S+N	1	Version/ Traffic Class	0x6X	Compare	Check IPv6
35+D+S+N	3	Traffic Class/Flow Label	-	Ignore	
38+D+S+N	2	Payload Length	-	Ignore	
40+D+S+N	1	Next Header	IPv6 extension header Or 0x06	Check	IPv6 extension headers
41+D+S+N	1	Hop Limit	-	Ignore	
42+D+S+N	16	Source Address	-	Ignore	
58+D+S+N	16	Destination Address		Ignore	
74+D+S+N	B	Possible IPv6 Next Headers	-	Ignore	
TCP Header					
74+T	2	Source Port	Not (0x801)	Check	Not NFS packet
76+T	2	Destination Port	Not (0x801)	Check	Not NFS packet
78+T	4	Sequence number	-	Ignore	
82+T	4	Acknowledge number	-	Ignore	
86+T	1/2	Header Length		Check	
86.5+T	1.5	Different bits	-	Ignore	

Offset	# of Bytes	Field	Value	Action	Comment
88+T	2	Window size	-	Ignore	
90+T	2	TCP checksum	-	Ignore	
92+T	2	Urgent pointer	-	Ignore	
94+T	F	TCP options	-	Ignore	

In this case the packet is split after (94+D+S+N+B+F) bytes.

- T = D+S+N+B
- N = (IP HDR length - 5) * 4.
- F = (TCP HDR length - 5)*4

A.2.2 Type 2: Ethernet, IPv6

A.2.2.1 Type 2.1: Ethernet, IPv6 Data

This type contains only an Ethernet header and an IPv6 header while the packet should be a fragmented packet. If the packet is not fragmented and the Next header is not supported, the header is not split. The supported packet types for header split are programmed in PSRTYPE register (per VF).

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
IPv6 Header					
12+D+S	2	Type	0x86DD	Compare	IP
14+D+S	1	Version/ Traffic Class	0x6X	Compare	Check IPv6
15+D+S	3	Traffic Class/Flow Label	-	Ignore	
18+D+S	2	Payload Length	-	Ignore	
20+D+S	1	Next Header	IPv6 next header types	Check	The last header must be fragmented header for the header to be split.
21+D+S	1	Hop Limit	-	Ignore	
22+D+S	16	Source Address	-	Ignore	
38+D+S	16	Destination Address		Ignore	
54+D+S	N	Possible IPv6 Next Headers	-	Ignore	

In this case the packet is split after (54+D+S+N) bytes.

- The last next header field of the IP section field should not be 0x11/0x06 (TCP/UDP).

A.2.2.1.1 Type 2.2: Ethernet (VLAN/SNAP) IPv6 UDP

This type contains only Ethernet header, IPv6 header, and UDP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag		Check	
IPv6 Header					
12+D+S	2	Type	0x86DD	Compare	IP
14+D+S	1	Version/ Traffic Class	0x6X	Compare	Check IPv6.
15+D+S	3	Traffic Class/Flow Label	-	Ignore	
18+D+S	2	Payload Length	-	Ignore	
20+D+S	1	Next Header	IPv6 next header types Or 0x11	Check	
21+D+S	1	Hop Limit	-	Ignore	
22+D+S	16	Source Address	-	Ignore	
38+D+S	16	Destination Address		Ignore	
54+D+S	N	Possible IPv6 Next Headers	-	Ignore	
UDP Header					
54+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet.
56+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet.
58+D+S+N	2	Length	-	Ignore	
60+D+S+N	2	Checksum	-	Ignore	

In this case the packet is split after (62+D+S+N) bytes.

- The last 'next-header' field of the last header of the IP section must be 0x06.

A.2.2.2 Type 2.3: Ethernet (VLAN/SNAP) IPv6 TCP

This type contains only Ethernet header, IPv6 header, and UDP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag		Check	
IPv6 Header					
12+D+S	2	Type	0x86DD	Compare	IP
14+D+S	1	Version/ Traffic Class	0x6X	Compare	Check IPv6
15+D+S	3	Traffic Class/Flow Label	-	Ignore	
18+D+S	2	Payload Length	-	Ignore	
20+D+S	1	Next Header	IPv6 next header types Or TCP	Check	
21+D+S	1	Hop Limit	-	Ignore	
22+D+S	16	Source Address	-	Ignore	
38+D+S	16	Destination Address		Ignore	
54+D+S	N	Possible IPv6 Next Headers	-	Ignore	
TCP Header					
54+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet
56+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet
58+D+S+N	4	Sequence number	-	Ignore	
62+D+S+N	4	Acknowledge number	-	Ignore	
66+D+S+N	1/2	Header Length		Check	
66.5+D+S+N	1.5	Different bits	-	Ignore	
68+D+S+N	2	Window size	-	Ignore	
70+D+S+N	2	TCP checksum	-	Ignore	
72+D+S+N	2	Urgent pointer	-	Ignore	
74+D+S+N	F	TCP options	-	Ignore	

In this case the packet is split after (54+D+S+N+F) bytes.

- F = (TCP header length - 5) * 4.
- The last 'next-header' field of the last header of the IP section must be 0x11.

A.2.3 Type 3: Reserved

Type 3 used to be iSCSI packets (Header split is not supported for iSCSI packets in the X550).

A.2.4 Type 4: Reserved

A.2.5 Type 5: Cloud Packets

A.2.5.1 Type 5.1: Ethernet, IPv4, NVGRE, IPv4/6, TCP/UDP

This type contains an Ethernet header an IPv4 Header, a GRE header an IPv4/6 header and a TCP or UDP header. The supported packet types for header split are programmed in PSRTYPE register (per VF).

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP Split on outer L2 is done after this field.
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x2F	Compare	GRE
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
GRE Header					
34 + D +S +N	2	Flags/Version	0x2000	Compare	
36 + D +S +N	2	Protocol Type	0x6558	Compare	
38+ D +S +N	3	TNI		Store	Used for Flow director.
41 + D +S + N	1	Reserved	0x0	Ignore	Split on Cloud header splits after this field.
Inner MAC Header					
42 + D +S + N	6	Inner Destination Address		Ignore	Used for Flow director.
48 + D +S + N	6	Inner Source Address		Ignore	

Offset	# of Bytes	Field	Value	Action	Comment
54 + D + S + N	I=(0/4)	Possible Inner VLAN Tag	0x8100	Store	Used for Flow director.
54 + D + S + N + I	2	Type	0x0800/0x86DD	Compare	Split on L2 is done after this field.
IPv4/6 header - as described above					Split on L3 is done after these fields.
UDP/TCP - as described above					Split on L4 is done after these fields.

A.2.5.2 Type 5.2: Ethernet, IPv4, VXLAN, MAC, IPv4/6, TCP/ UDP

This type contains an Ethernet header an IPv4 Header, a UDP header with a specific UDP destination port a VXLAN header, an inner MAC header, an IPv4/6 header and a TCP or UDP header. The supported packet types for header split are programmed in PSRTYPE register (per VF).

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP Split on outer L2 is done after this field
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x11	Compare	UDP
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
UDP Header					
34+D+S+N	2	Source Port		Ignore	
36+D+S+N	2	Destination Port		Compare	Compare to VXLANCN-TRL.UDPPORT
38+D+S+N	2	Length	-	Ignore	

Offset	# of Bytes	Field	Value	Action	Comment
40+D+S+N	2	Checksum	-	Ignore	
VXLAN Header					
42 + D +S + N	1	Flags	0x08	Compare	VNI exists
43 + D + S + N	3	Reserved	0x000000	Compare	
46 + D + S + N	3	VNI		Store	Used for flow director
49 + D + S + N	1	Reserved	0x00	Compare	Split on Cloud header splits after this field.
Inner MAC Header					
50 + D +S + N	6	Inner Destination Address		Ignore	Used for Flow director
56 + D +S + N	6	Inner Source Address		Ignore	
62 + D + S + N	I=(0/4)	Possible Inner VLAN Tag	0x8100	Store	Used for Flow director
62 + D + S + N + I	2	Type	0x0800/0x86DD	Compare	Split on L2 is done after this field
IPv4/6 header - as described above					Split on L3 is done after these fields
UDP/TCP - as described above					Split on L4 is done after these fields

A.3 IPsec Formats Run Over the Wire

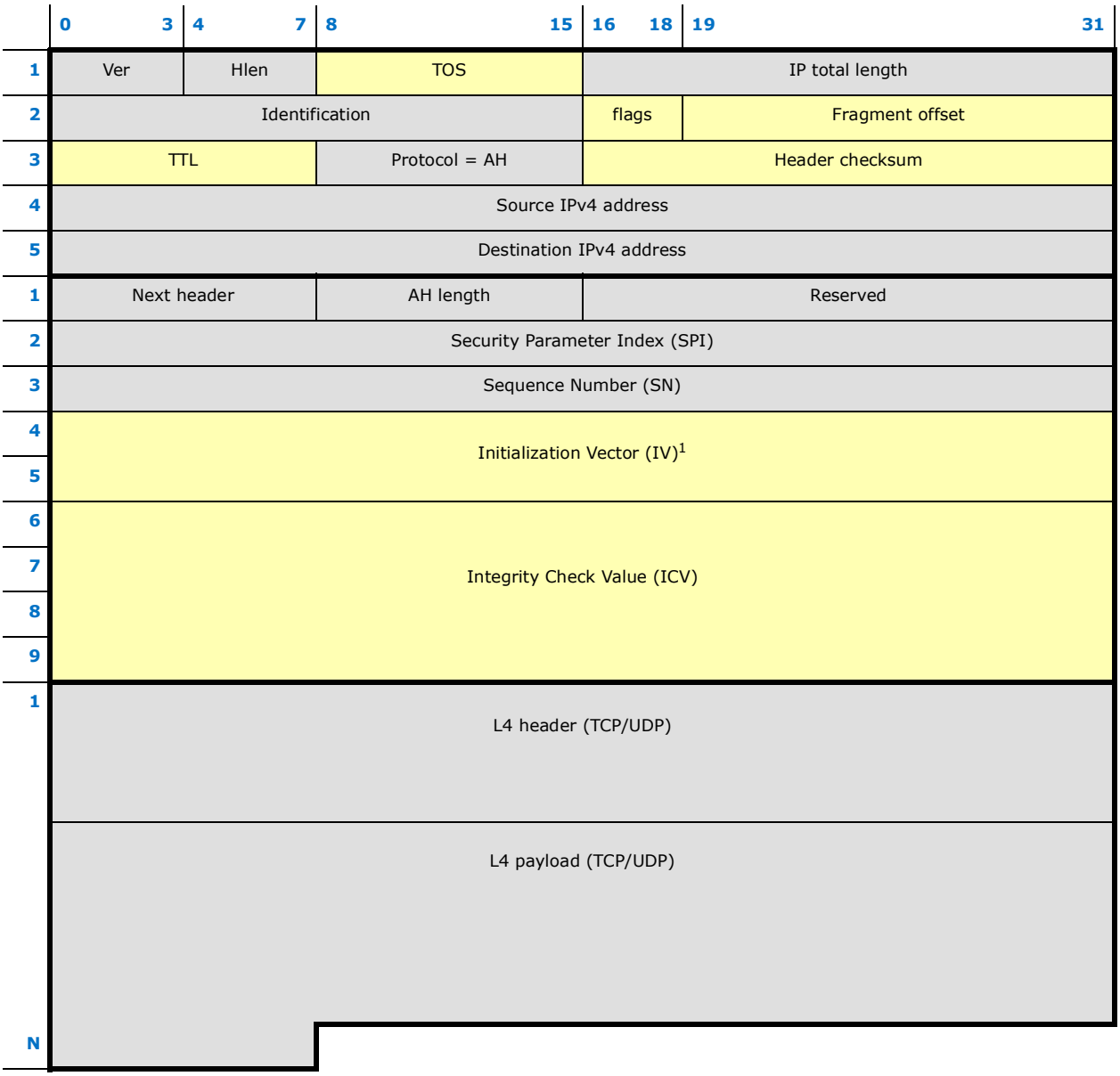
This section describes the IPsec packet encapsulation formats run over the wire by IPsec packets concerned with the off load in either Tx or Rx direction.

The following legend is valid for Figure A-12 through Figure A-17 of this appendix:

Shaded fields correspond to the portion of the data that is protected by the integrity check.
Yellow colored fields are mutable fields that might be changed when traveling between the source and the destination, and are zeroed when computing ICV or when encrypting/decrypting.
Cyan colored fields correspond to the portion of data that is protected for both integrity and confidentiality.
Non-colored fields are not protected either for integrity or for confidentiality.

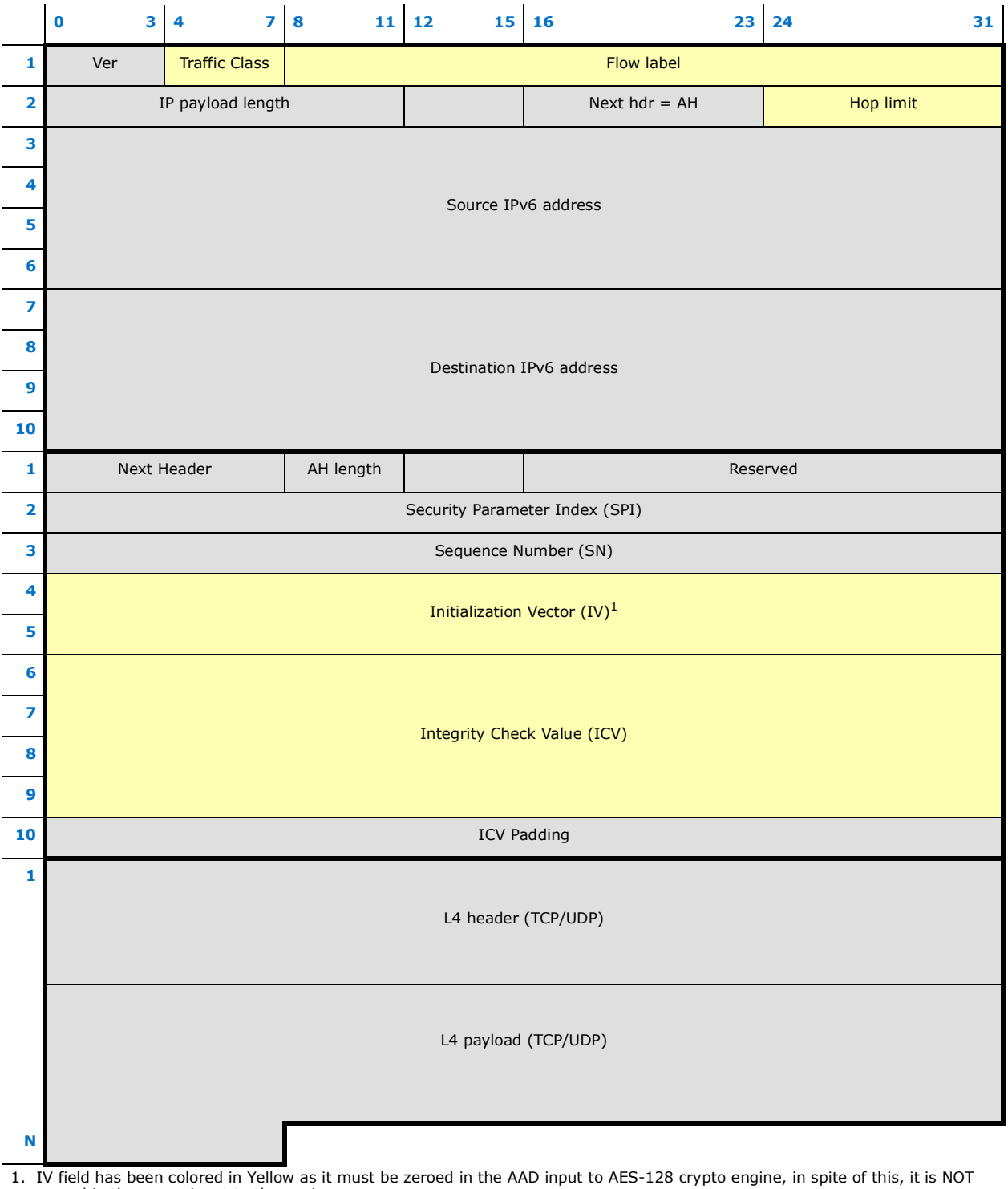
A.3.1 AH Formats

- IPv4 header:
  - **IP total length** (2 bytes) — Total IP packet length in bytes, including IP header, AH header, TCP/UDP header, and TCP/UDP payload.
  - **Protocol** (1 byte) — AH protocol number, i.e. value 51.
- IPv6 header:
  - **IP payload length** (2 bytes) — IP payload length in bytes, including AH header, TCP/UDP header, and TCP/UDP payload.
  - **Next header** (1 byte) — AH protocol number, i.e. value 51.
- AH header:
  - **Next header** (1 byte) — Layer4 protocol number, 6 for TCP, 17 for UDP, etc.
  - **AH length** (1 byte) — Authentication Header length in 32-bit DWords units, minus “2”, i.e. for AES-128 its value is 7 for IPv4 and 8 for IPv6.
  - **Reserved** (2 bytes) — Must be set to zero.
  - **SPI** (4 bytes) — Arbitrary 32-bit Security Parameters Index allocated by the receiver to identify the SA to which the incoming packet is bound. It is required that the local OS allocates SPIs in a unique manner per local IP Address.
  - **SN** (4 bytes) — Unsigned 32-bit Sequence Number that contains a counter value that increases by one for each Ethernet frame sent. It is initialized to 0 by the sender (and the receiver) when the SA is established, i.e. the first packet sent using a given SA has a sequence number of 1.
  - **IV** (8 bytes) — Initialization Vector to be used ‘as is’ in the nonce input to AES-128 crypto engine, but it must be zeroed prior to using it in the AAD input to the engine.
  - **ICV** (16 bytes) — Integrity Check Value for this packet, authentication tag output of the AES-128 crypto engine. As being part of the AH header, this field is also included in the AAD input to the crypto engine, and it should be zeroed prior to the computation.
  - **ICV Padding** (4 bytes) — *Explicit* padding bytes appended to the ICV field in IPv6, as it is required to maintain the (Authentication) extension header length as a multiple of 64 bits. By *explicit* we mean that these bytes are sent over the wire. It is formed by 4 arbitrary bytes that need not be random to achieve security. For TSO, it is replicated from the header provided by the driver in every frame.
- L4 header (for example - TCP/UDP) — Length (in bytes) depend on the protocol.
- L4 payload (for example - TCP/UDP) — Can be any length in bytes.



1. IV field has been colored in Yellow as it must be zeroed in the AAD input to AES-128 crypto engine, in spite of this it is NOT zeroed in the nonce input to the engine.

Figure A-12. AH Packet Over IPv4



1. IV field has been colored in Yellow as it must be zeroed in the AAD input to AES-128 crypto engine, in spite of this, it is NOT zeroed in the nonce input to the engine.

Figure A-13. AH Packet Over IPv6

A.3.2 ESP Formats

- IPv4 header:
  - **IP total length** (2 bytes) — Total IP packet length in bytes, including IP header, ESP header, TCP/UDP header, TCP/UDP payload, ESP trailer, and ESP ICV if present.
  - **Protocol** (1 byte) — ESP protocol number, i.e. value 50.
- IPv6 header:
  - **IP payload length** (2 bytes) — IP payload length in bytes, including ESP header, TCP/UDP header, TCP/UDP payload, ESP trailer, and ESP ICV if present.
  - **Next header** (1 byte) — ESP protocol number, i.e. value 50.
- ESP header:
  - **SPI** (4 bytes) — arbitrary 32-bit Security Parameters Index allocated by the receiver to identify the SA to which the incoming packet is bound. It is required that the local OS allocated SPIs in a unique manner per local IP Address.
  - **SN** (4 bytes) — unsigned 32-bit Sequence Number that contains a counter value that increases by one for each Ethernet frame sent. It is initialized to 0 by the sender (and the receiver) when the SA is established, i.e. the first packet sent using a given SA has a sequence number of 1.
  - **IV** (8 bytes) — Initialization Vector to be used in the nonce input field of the AES-128 crypto engine, and for authenticated-only ESP packets it is used also in the AAD input.
- L4 header (for example - TCP/UDP):
  - Length (in bytes) depend on the protocol.
  - TCP/UDP checksum computed from the TCP/UDP header up to the end of TCP payload, excluding ESP trailer.
  - TCP/UDP header is encrypted if ESP encryption is required.
- L4 payload (for example - TCP/UDP)— Can be any length in bytes. It is encrypted if ESP encryption is required.
- ESP trailer:
  - **Padding** (0-255 bytes) — unsigned 1-byte integer values, with its content started by 1 and making up a monotonically increasing sequence: 1, 2, 3,... Though in Tx it is only 0-15 bytes long, in Rx it might be longer, if the sender's policy is to hide the packet length.
  - **Padding length** (1 byte) — Number of *explicit* padding bytes (Padding length and Next header bytes excluded) required to get 4-bytes alignment of the ESP header, TCP/UDP header, TCP/UDP payload, and ESP trailer. By *explicit* we mean that these bytes are sent over the wire. A remote IPsec implementation may also add more padding bytes (up to 255-bytes) than the minimum required for getting the 4-bytes alignment with the aim of hiding the packet length.
  - **Next header** (1 byte) — Layer4 protocol number, 6 for TCP, 17 for UDP, etc.
  - **ESP** trailer is encrypted if ESP encryption is required.
- **ESP ICV** (16 bytes) — Integrity Check Value for this packet, authentication tag output of the AES-128 crypto engine.

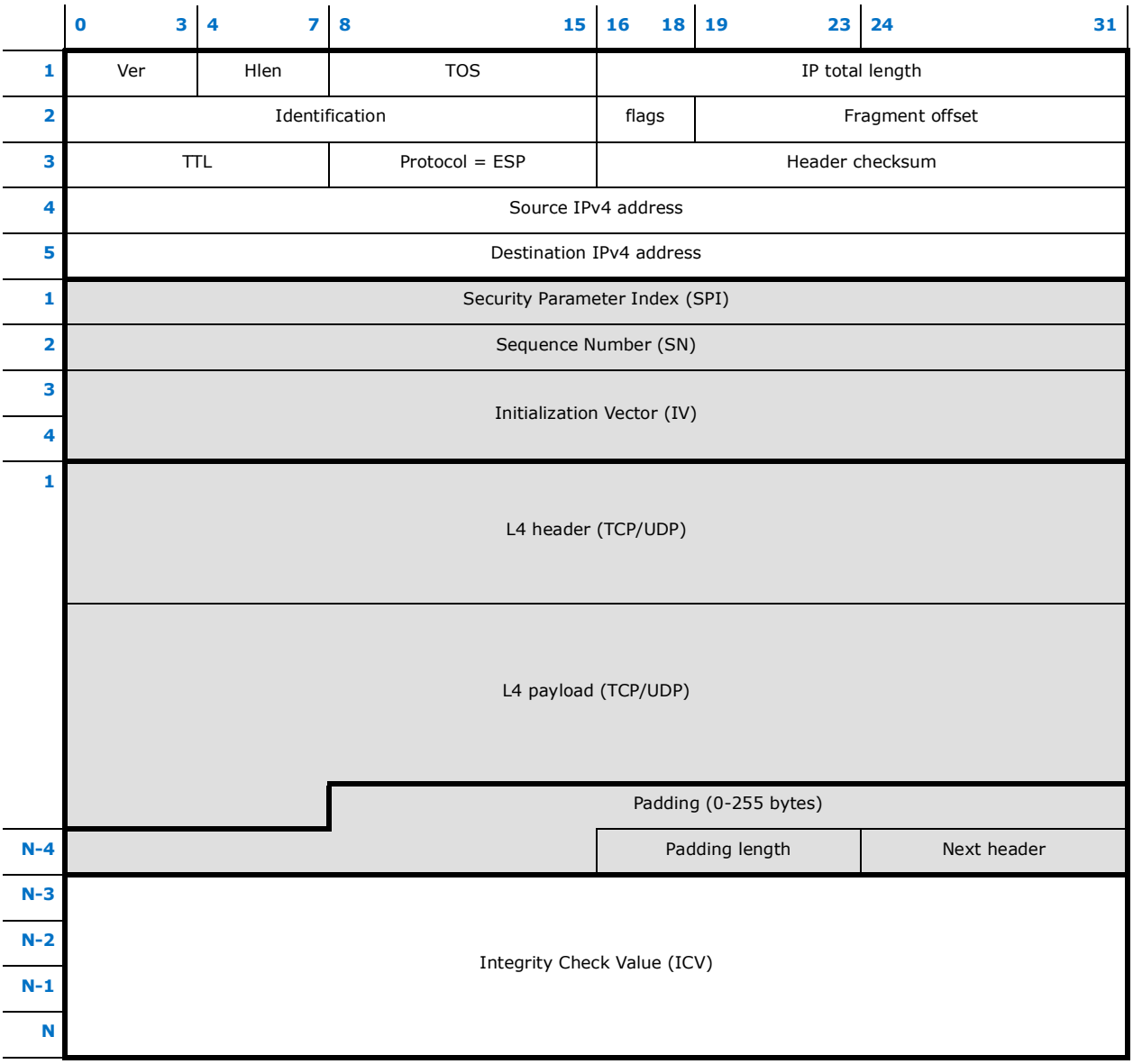


Figure A-14. Authenticated Only ESP Packet Over IPv4

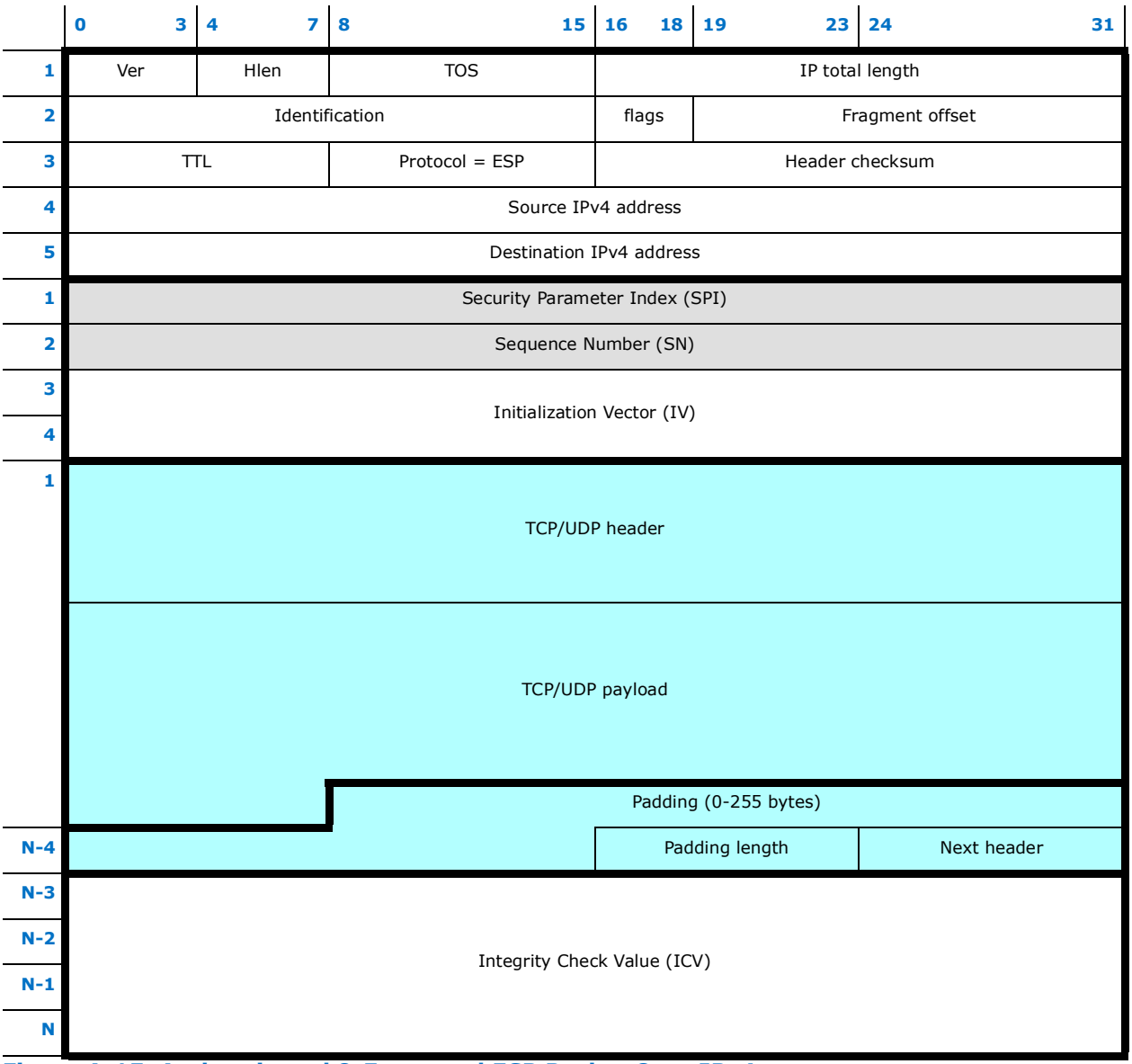


Figure A-15. Authenticated & Encrypted ESP Packet Over IPv4

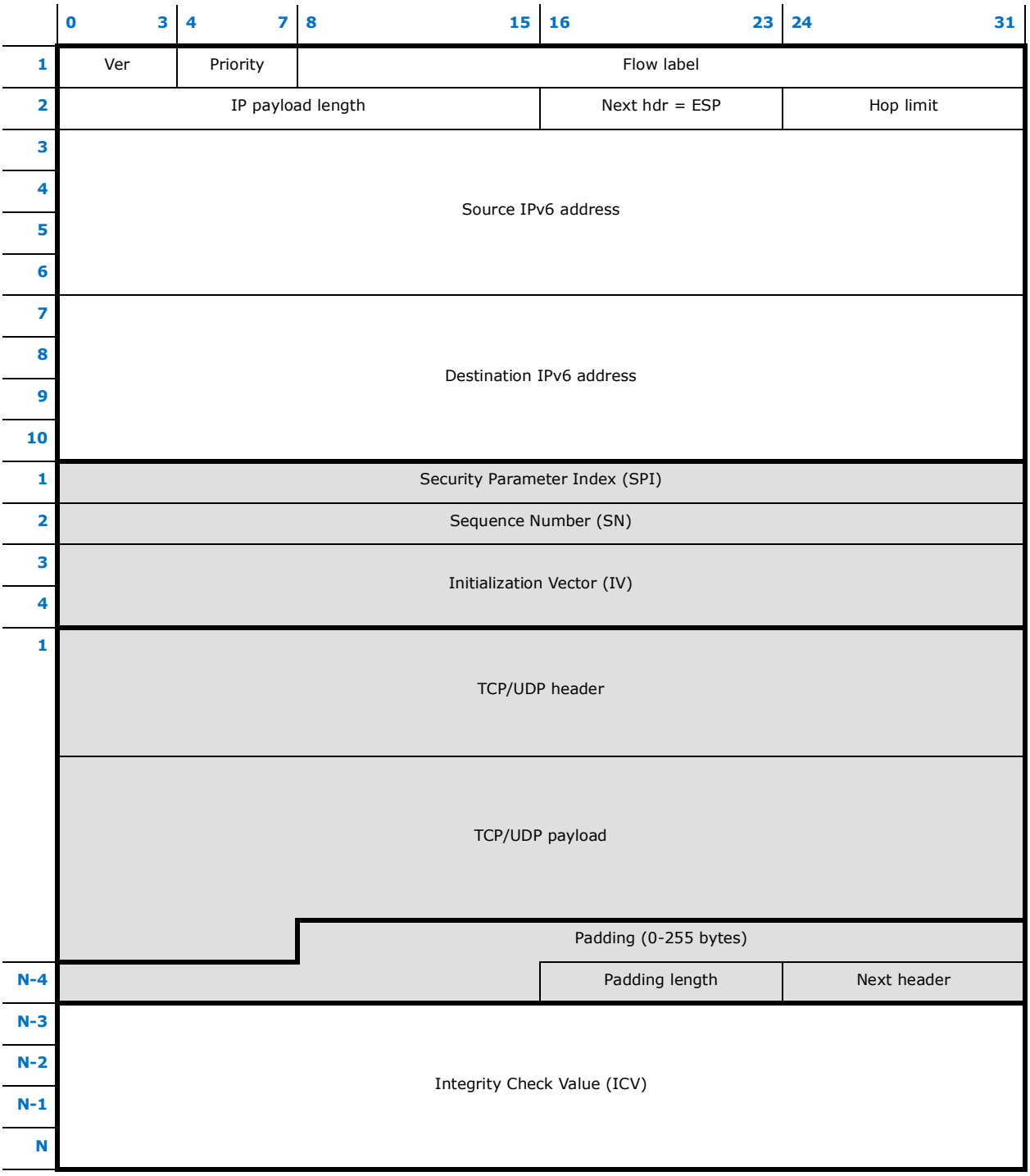


Figure A-16. Authenticated Only ESP Packet Over IPv6

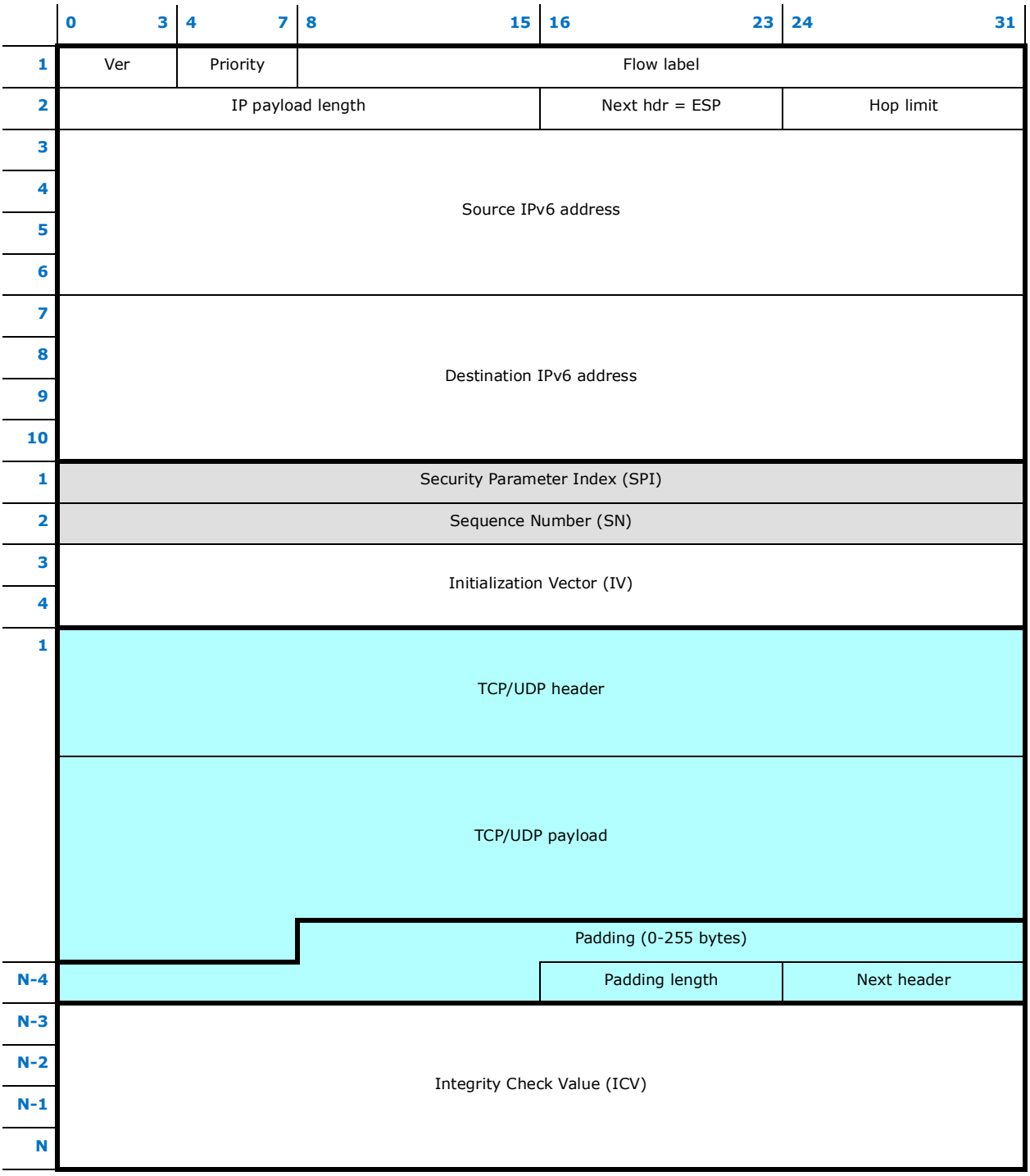


Figure A-17. Authenticated and Encrypted ESP Packet Over IPv6

For authenticated and encrypted ESP packets, though it is used in the Nonce input to the AES-128 crypto engine, the IV field was left non-colored because it is not a part of the ADD or the Plaintext input fields.

A.4 FCoE Framing

Related standards:

- **FC-FS-2** — FRAMING AND SIGNALING-2 (FC-FS-2) Rev 1.00.
- **FCoE** — Fibre Channel Backbone - 5 (FC-BB-5), v2.0, 6/4/2009, INCITS 462-2010 Document: available from http://www.incits.org/ (09-056v5.pdf).

A.4.1 FCoE Frame Format

FC over Ethernet packets encapsulate FC frames as shown in Figure A-18 and Figure A-19. Maximum expected FCoE frame size is 2164 bytes. This size does not include FC extension headers (not expected in FCoE), without optional encapsulation (expected to be off-loaded by the hardware) and without CN tag (not expected in receive).

All fields in the FCoE frame are treated as any other field in the network. The MS Byte is first on the wire and the LS bit on each byte is first on the wire. This rule applies for all fields including those ones that span on non complete byte boundaries such as the FCoE VER field.

Note: LSB goes first on the wire.

<table>
<tr><td rowspan="2">Ethernet MAC Addresses</td><td rowspan="2">VLAN Tag</td><td rowspan="2">FCoE Header</td><td colspan="3">FC Frame</td><td rowspan="2">EOF</td><td rowspan="2">Ethernet CRC</td></tr>
<tr><td>FC Header</td><td>Optional FC Data</td><td>FC CRC</td></tr>
</table>

Figure A-18. Ethernet Encapsulation to FC frames

	msb 31 lsb 24 First byte on the wire	msb 23 lsb 16	msb 15 lsb 12	msb 11 lsb 8	msb 7 lsb 0 Last byte on the wire
0	Destination Ethernet MAC Address				
4					
8	Source Ethernet MAC Address				
12	802.1Q Tag (VLAN + Priority)				
16	FCoE Ethernet Type = 0x8906		Ver	Reserved	
20	Reserved				
24	Reserved				
28	Reserved				SOF
32	Routing Control (R_CTL)	Destination Identification (D_ID)			
36	Class Specific Control (CS_CTL)	Source Identification (S_ID)			
40	TYPE	Frame Control (F_CTL)			
44	Sequence ID (SEQ_ID)	Data Field Control (DF_CTL)	Sequence Count (SEQ_CNT)		
48	Originator Exchange ID (OX_ID)		Responder Exchange ID (RX_ID)		
52	Parameter (PARAM)				
0...N	Optional FC Data... always 4 byte align (including optional FC padding)				
56 +N	Fibre Channel CRC (FC_CRC)				
N+8	EOF	Reserved			
M	Ethernet CRC				

Figure A-19. FCoE Packet Structure

A.4.1.1 Ethernet MAC Addresses

L2 destination and Source Ethernet MAC Addresses (each of them is 6 bytes long). The Ethernet MAC Address of the target is assumed to be assigned by the network. It could be done by the FCoE repeater or LAN administrator. The mechanism that is used for Ethernet MAC Address assignment and Ethernet MAC Address detection is outside of the scope of this document.

A.4.1.2 802.1Q Tag

802.1Q tagging is mandatory for FCoE usage. FCoE assumes DCB functionality which provides Priority-based FC and QCN. These functions require 802.1Q tag presence to define packet priority.

A.4.1.3 FCoE Header

The FCoE header is composed of a new FCoE Ethernet type, FCoE version tag and the Start Of Frame tag.

- **FCoE Ethernet Type** — Equals 0x8906
- **Version (Ver)** — A 4-bit field that indicates the FCoE protocol version number. The X550 supports FCoE version as defined by FCRXCTRL.FCOEVER.
- **Start of Frame (SOF)** — The FCoE Start of frame is a subset of the FC-FS-2 SOF codes as defined in the Table A-1.

Table A-1. E_SOF Mapping

FC SOF	FCoE SOF Code	FC Traffic Class	Comment
SOFf	0x28	F	Fabric start of frame. Not expected in an FCoE.
SOFi2	0x2D	2	Used in the first frames in a sequence.
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.

A.4.1.4 FCoE Packet Encapsulation Trailer

The FCoE trailer is composed of an End of frame, optional padding and Ethernet CRC.

- **End of Frame (EOF)** — The FCoE End of frame maps the FC-FS-2 EOF codes as defined in the Table A-2.
- **Ethernet Padding** — The minimal FC frame length could be as small as 28 bytes long (FC header with no data). In such a case, the encapsulated FC frame must include padding so that the total Ethernet packet size is not smaller than 64 bytes.
- **Ethernet CRC** — The IEEE 802.3 CRC as defined by the following polynomial:

$$X_{32}+X_{26}+X_{23}+X_{22}+X_{16}+X_{12}+X_{11}+X_{10}+X_8+X_7+X_5+X_4+X_2+X+1$$

Table A-2. 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.

A.4.2 FC Frame Format

Note: This section is provided as a background on FC and is not required for the hardware implementation. For a complete description of the FC fields, refer to FC-FS-2 specification.

The FC frame as defined in FC-FS-2 specification is shown in the Figure A-20 below while relevant fields are detailed in this section.

Figure A-20. FC Frame Format

A.4.2.1 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 A-1 and Table A-2.

A.4.2.2 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 hardware adds this field if the *FCoE* bit is set in the transmit context descriptor. The FC CRC off load is described in more detail in Section 7.10.3.2.

A.4.2.3 FC Header

The FC header fields are provided in the header buffer by the FCoE driver. The FC header includes fields that are modified by the hardware as part of Large Send off load. These fields are indicated in this section as "**Dynamic**" while fields that are not modified by the hardware are indicated as "**Static**".

Routing Control (R_CTL)

The R_CTL is a one-byte **Static** field that contains routing and information bits to categorize the frame function.

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.

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 is further described in Section A.4.2.3.1. Section 7.10.2.7 describes how the F_CTL field is modified during large send.

Data Field Control (DF_CTL)

The DF_CTL is a 1 byte **Dynamic** field that specifies the presence of optional headers at the beginning of the Data_Field. The Optional headers supported by large send are present only on the first frame in the FC sequence. Section 7.10.2.7.1 describes how the DF_CTL field is modified during large send.

Sequence ID (SEQ_ID)

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.

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 increased 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 increased 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 must 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 X550 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 must provide a unique, locally meaningful exchange identifier at the Responder. The Responder of the Exchange must 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.

A.4.2.3.1 Frame Control (F_CTL)

The Frame Control (F_CTL) is a three-byte field that contains control information relating to the frame content. If an error in bit usage is detected, the software initiates a reject frame (P_RJT) in response with an appropriate reason code (FCoE software driver responsibility). The F_CTL format is shown in Table A-3 below.

When a bit(s) is designated as "**Static**" it is provided by the FCoE driver as part of the large send header. The hardware keeps this bit(s) as is in all preceding frames of the large send.

When a bit(s) is designated as "**Dynamic**" it is provided by the FCoE driver as part of the large send header. The hardware may change it in some of the packets in the large send as described in Section 7.10.2.7. Exact setting of these bit(s) is described in the Table A-3 below.

When a bit(s) is designated as meaningful under a set of conditions, that bit must be ignored if those conditions are not present.

Table A-3. F_CTL Format

Control Field	Bit(s)	Type	Description
Exchange Context	23	Static	0b = Originator of exchange. 1b = Responder of exchange.
Sequence Context	22	Static	0b = Sequence initiator. 1b = Sequence recipient.
First Sequence	21	Static	0b = Sequence other than first of exchange. 1b = first Sequence of exchange.
Last Sequence	20	Static	0b = Sequence other than last of exchange. 1b = 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.
End Sequence	19	Dynamic	0b = Data frame other than last of sequence. 1b = last Data frame of sequence.
End Connection	18	Static	Relevant for Class 1 or 6.
CS CTL / Priority Enable	17	Static	Defines the meaning of the CS_CTL byte in the FC header to be either CS_CTL or Priority indication. 0b = Word 1, Bits 31-24 = CS_CTL 1b = Word 1, Bits 31-24 = Priority
Sequence Initiative	16	Dynamic	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). 0b = Hold sequence initiative. 1b = Transfer sequence initiative.
X_ID reassigned	15	Static	Obsolete.
Invalidate X_ID	14	Static	Obsolete.
ACK Form	13:12	Static	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. 00b = No assistance provided10b = reserved. 01b = Ack_1 Required11b = Ack_0 Required.
Data Compression	11	Static	Obsolete.
Data Encryption	10	Static	Obsolete.
Retransmitted Sequence	9	Static	Meaningful in Class 1 and 6 and only. 0b = Original sequence transmission. 1b = sequence retransmission.
Unidirectional Transmit	8	Static	Relevant for Class 1
Continue Sequence Condition	7:6	Dynamic	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'.

Table A-3. F_CTL Format [continued]

Control Field	Bit(s)	Type	Description
Abort Sequence Condition	5:4	Static	**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 must 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 sequences with immediate re-XMT.
Relative offset present	3	Static	0b = Parameter field defined for some frames 1b = Parameter Field = relative offset
Exchange reassembly	2	Static	Reserved for exchange reassembly.
Fill Bytes	1:0	Dynamic	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).

A.4.2.4 FC Extended Headers

Note: The following section is provided as a background on FC and is not required for the hardware implementation or the software implementation since in FCoE frames, extended headers are not expected. The following quote is from the FCoE specification Rev 0.7 "No Extended Headers are relevant for the operations of a Reedtown NIC".

The diagrams in Figure A-21, Figure A-22 and Table A-4 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. The LAN controller does not support these Extended headers since they are not expected in FCoE usage.

FC Frame

Extended Headers [Optional]	FC Header	Optional FC Header	[optional] FC Data (including optional padding)	FC CRC

Figure A-21. FC Frame format with Extended Headers

	msb 31 lsb 24 Last byte on the wire	msb 23 lsb 16	msb 15 lsb 8	msb 7 lsb 0 First byte on the wire
0	Extended Header Specific Fields			R_CTL
4...				

Figure A-22. Extended Header Format

Table A-4. FC Extended Headers

R_CTL	Extended Header	Length
0x50	VFT_Header (Virtual Fabric Tagging Header)	8 Bytes
0x51	IFR_Header (Inter-Fabric Routing Header)	8 Bytes
0x52	Enc_Header (Encapsulation Header)	24 Bytes
0x53…0x5F	Reserved	-

A.4.2.5 FC Optional Headers

Note: Most of the following section is provided as a background on FC and is not required for the hardware implementation. The reader may skip the detailed explanation of the Optional headers provided below and concentrate in the tables and figures that follow the text.

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. In Large send functionality, the FC Optional headers may show only on the first frame as shown in Figure 7-46 and Table A-5 (in Section 7.10.2.7.1).

The following diagrams Table A-5, Figure A-23 and Figure A-24 show the Fibre Channel frame structure with Optional headers and lists the Optional headers. The Optional headers (that are present) are always ordered as shown in Figure A-23 and Figure A-24. Their presence is indicated in the Data Field Control (DF_CTL) field in the FC Header as indicated in Table A-5

Maximum FC frame size: The sum of the length in bytes of the FC Payload, the number of fill bytes, and the lengths in bytes of all optional headers must not exceed 2112.

Table A-5. FC Optional Headers

DF_CTL	Optional Header	Length
bit 6	ESP Header / ESP Trailer	Variable
bit 5	Network Header	16 Bytes
bit 4	Association Header	32 Byte
bits 1:0	Device Header	0, 16, 32, or 64 Bytes

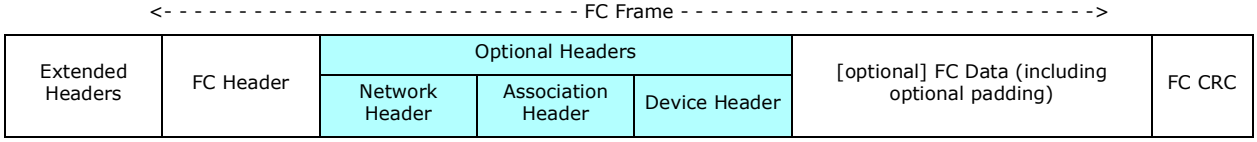


Figure A-23. FC Frame Format with Optional Headers (without ESP Header)

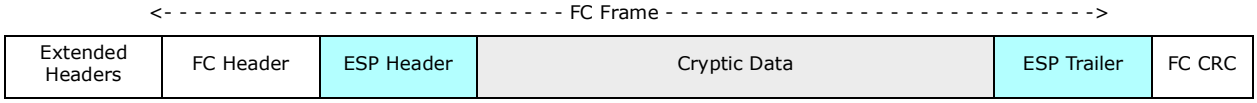


Figure A-24. FC Frame Format with Optional Headers (with ESP Header)

ESP Header

This is the first Optional header which covers the whole FC frame other than the FC header which is transmitted on the clear (as plain text). When an ESP header is present there is also the ESP trailer. If required, the software is responsible for the cryptic calculation and preparing the ESP header and trailer. Its presence is indicated by bit 6 in the DF_CTL field being set to one. The hardware does not support Large Send off load when ESP Optional header is used. When present, the ESP header and trailer are present in all frames of the exchange.

Network Header

The Network Header, if used, must be present only in the first Data frame of a Sequence. A bridge or a gateway node that interfaces to an external Network may use the Network Header. The Network Header, is an optional header 16 Bytes long within the FC Data Field content. Its presence is indicated by bit 5 in the DF_CTL field being set to one. The Network Header may be used for routing between Fibre Channel networks of different Fabric address spaces, or Fibre Channel and non-Fibre Channel networks. The Network Header contains Name Identifiers for Network Destination Address and Network Source Address.

Association Header

The Association Header, if used, must be present only in the first Data frame of a Sequence. The Association Header is an optional header 32 Bytes long within the Data Field content. Its presence is indicated by bit 4 in the DF_CTL field being set to one. The Association Header may be used to identify a specific Process or group of Processes within a node associated with an Exchange. When an Nx_Port has indicated during Login that an Initial Process Associator is required to communicate with it, the Association Header should be used by that Nx_Port to identify a specific Process or group of Processes within a node associated with an Exchange. The X550 does not use the Association for any filtering purposes but rather uses the OX_ID.

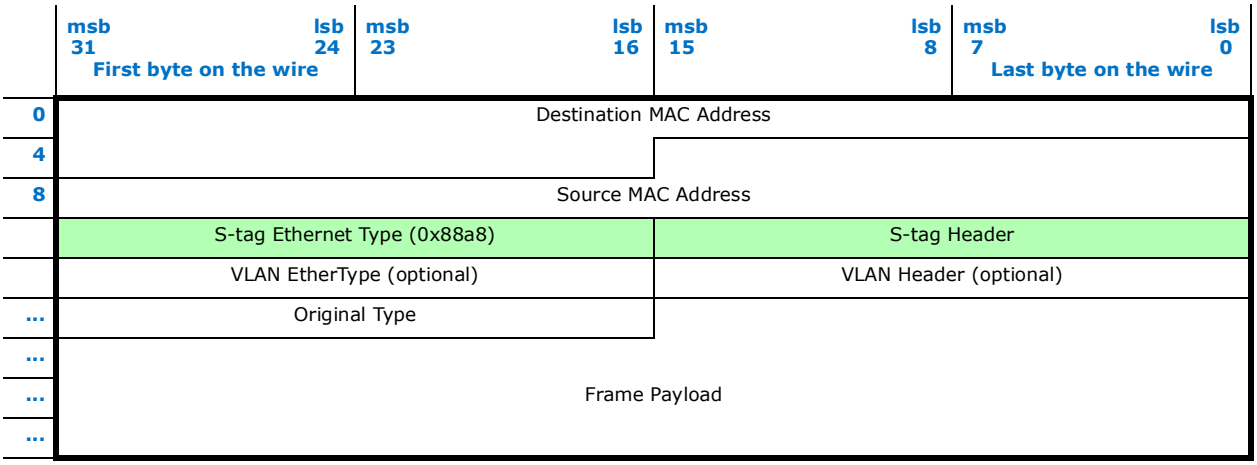
Device Header

The Device Header, if present, must be present either in the first Data frame or in all Data frames of a Sequence. If Large Send off load is used, the Device header, if present, is present only in the first frame of the same large send. The Device Header, if present, must be 16, 32, or 64 bytes in size as defined by bits 1:0 in the DF_CTL field. The contents of the Device Header are controlled at a level above FC-2. Upper layer protocol (ULP) may use a Device Header, requiring the Device Header to be supported. The Device Header may be ignored and skipped, if not needed. If a Device Header is present for a ULP that does not require it, the related FC-4 may reject the frame with the reason code of “TYPE not supported”.

A.5 E-tag and S-tag Formats

Note: S-tags are not supported in the X550.

Packets with S-tag has the following format:



The S-tag format is the format defined for S-tags in the IEEE 802.1ad specification as described below:

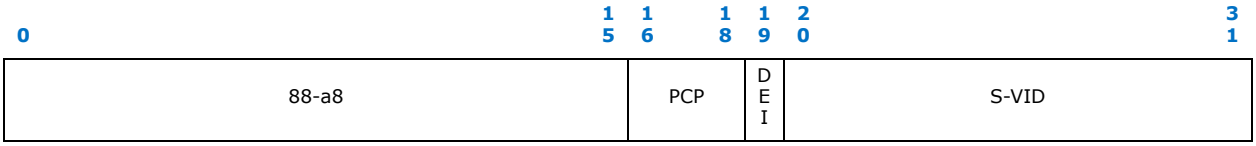
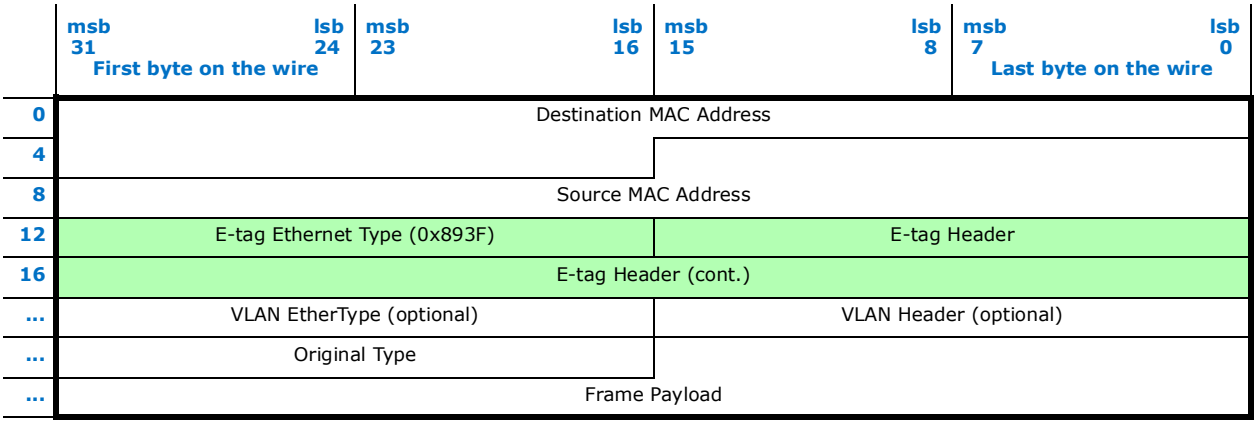


Figure A-25. S-tag Format

Packets with E-tag has the following format:



The E-tag format is the format defined in the IEEE 802.1BR specification as described below:

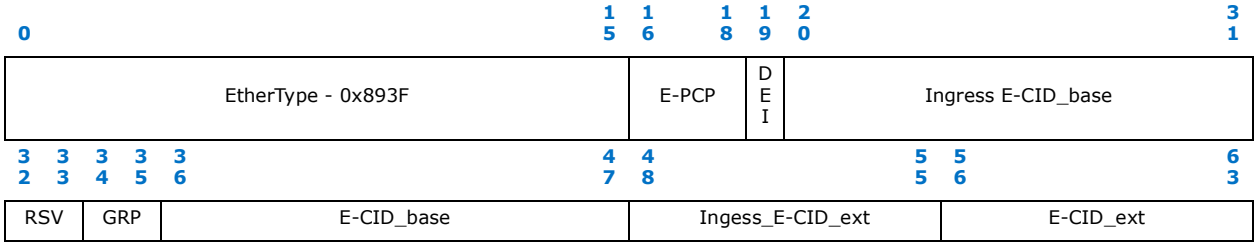


Figure A-26. E-tag Format

The Ingess_E-CID_ext and E-CID_ext are always zero for endpoints and are effectively reserved.