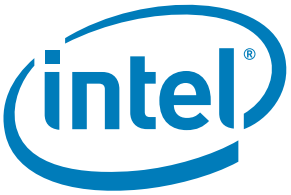


Intel® Ethernet Controller X540 Datasheet

Networking Division

PRODUCT FEATURES

General

- Serial Flash interface
- Configurable LED operation for software or customizing OEM LED displays
- Device disable capability
- Package size - 25 mm x 25 mm

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
- DCA support
- TCP timer interrupts
- No snoop
- 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 base specification 2.1 (2.5GT/s or 5GT/s)
- Bus width — x1, x2, 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

Order # 333168-001

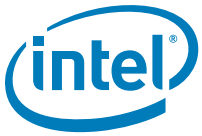
Revision Number: 2.9

September 2015

Revisions

Rev	Date	Notes
2.9	September 2015	• Updated section 4 (Reset Effects - step 2e). • Updated section 6.3.3 Boot Configuration Block — Word Address 0x17 (Added iSCSI Configuration Block description). • Updated section 8.2.4.2.3.8 (RX Read Buffer Pointers - RXRDPTR[n]). • Updated section 8.2.4.2.3.12 (TX Read Buffer Pointers - TXRDPTR[n]). • Updated section Glossary and Acronyms (LDPC description).
2.8	February 2015	• Updated section 2.1.10 (Miscellaneous). • Updated section 4.6.6 (Interrupt Initialization). • Updated section 4.6.9 (FCoE Initialization Flow). • Updated table 8-2 (MAC Registers). • Updated section 8.2.4.10.3 (DCB Transmit Packet plane Control and Status - RTTPCS (0x0000CD00); bits 31:22). • Updated section 8.2.4.18.3 (Error Byte Packet Count - ERRBC (0x00004008); EBC field). • Updated section 13.10 [(Power On Reset (POR)].
2.7	March 2014	• Added MCTP footnote to table 1-7. • Added a note to section 1.4.2 (MCTP Over SMBus). • Revised section 3.6.3.2 (Auto-Negotiation and Link Setup). • Revised section 4.6.3.2 (Global Reset and General Configuration) • Revised section 8.2.4.1.3 (Device Status Register; bit 7). • Added a note to section 11.6.1.1 (NC-SI over MCTP) • Revised section 11.6.3 (MCTP over SMBus; Message Type = 0x02 instead of 0x05). • Removed all Simplified MCTP Mode references (not supported). • Revised table 12-1 (Notes for Power On/Off Sequence Diagram; note 5). • Added section 16 (Packets Format).
2.6	December 2013	Revised sections/tables/figures: • Table 1-2 (Support of non Auto-Negotiation Partner). • 3.6.3.2 (Auto-Negotiation and Link Setup). • 5.2.3 (removed). • Notes below Figure 5.4 (first paragraph). • 5.3.3.1 (last paragraph). • Figure 5.3 and 5.4 (removed AN enabled block). • 6.4.4.13 (bits 10:5). • 7.2.3.2.2 (RS (bit 3 description). • Table 11-1 (Select Package/Deselect Package commands). • Tables 12-2 through 12-4 (Current Consumption). • 12.4.2 (new section - Peak Current Consumption). • 13.0 (added design considerations and guidelines for integrated magnetics).
2.5	September 2013	Revised sections/tables/figures: • 2.1.6 (changed NC-SI_CRS_DV pull up/pull down value). • 2.1.10 (changed note 6 to note 5; last row of table). • 4.4.1 (revised note concerning single-port NVM). • 4.6.11.3.1 (updated 8 TCs mode and 4 TCs mode description). • 4.6.11.4.3 (removed MTQC.DCB_Ena set to 1b sub-bullet). • 7.2.3.1 (added Rate-Scheduler to second paragraph). • 7.2.3.2.4 (removed "in IOV mode" under Check Context Bit description). • 7.6 (revised LINK/ACTIVITY description). • 8.1.1.2 (revised Memory-Mapped Accesses to Flash description). • Table 8-4 (changed SECRXSAECC and SECRXAESECC offset values). • 8.2.4.1.5 (changed NC-SI Configuration 1 word to NC-SI Configuration 2 word). • 8.2.4.1.11 (changed bits 21:20, 22, and 23 initial values). • 8.2.4.29.20 and 8.2.4.29.21 (removed). • 8.2.4.29.55 and 8.2.4.29.56 (changed offset values). • 12.1.4 (added note after tables 12-2, 12-3, and 12-4). • Table 12-4 (added X540-BT2 Dual-Port Current Consumption using Single-port NVM power values). • 12.3.9 (changed min/max values; threshold for 0.8 Vdc supply). • Table 13-5 (added Integrated Magnetics vendor information).

2.4	April 2013	Changed single port SKU to single port configuration. Revised sections: • 3.2.5.1 (changed flags off to flags on). • 3.4.8 (added 82575). • 3.5 (removed old table 3.15, added new SDP settings table, and added new signal names). • 4.6.9 (add new text; last bullet). • 4.6.11.4.3 (changed INT[13:0] to DEC[13:0]. • 5.3.5 (new section). • 6.1 (removed note). • 6.4.4.8 (revised SDP_FUNC_OFF_EN bit description). • 7.2.3.2.3 (added new EOF Codes in TSO table and new table references; also revised HEADLEN description). • 7.7.2.4.1 (User Priority (UP) description). • 7.9.3 (changed MAC reset to Master reset). • 7.9.4 (revised table note). • 7.13.1.1 (revised FC Frame description). • 7.13.3.3.6 (SEQ_ID (8 bit) and SEQ_CNT (16 bit) descriptions). • 8.2.4.1.5 (revised table note 2). • 8,2.4.5.1 (revised FLOW_DIRECTOR bit description. • 8.2.4.7.9 and 8.2.4.7.10 (revised notes at the end of register tables). • 8.2.4.9.10 (revised WTHRESH bit description). • 8.2.4.10.1 (changed bits 18:16 to reserved). • 8.2.4.21.1 through 8.2.4.21.4 (new sections). • 8.2.4.22.10 and 8.2.4.22.11 (revised bit descriptions) • 8.2.4.29.44 through 8.2.4.29.51;8.2.4.29.73 and 8.2.4.29.73 (removed). • 8.2.4.29.73 and 8.2.4.29.74 (removed register RXFECCSTATC and RXFECCSTATUC). • 10.5.8 (changed 100BASE-TX Test Mode [1:0] bit setting 11b to reserved). • 10.6.12 (revised F bit default setting). • 12.4.1 (added new current consumption tables). • 12.7.4 (added new mechanical package diagram). • 16.0 (revised MDC and MDI descriptions).
2.3	November 2012	• Added single-port SKU information.
2.2	July 2012	• Revised footnote to table 1.5 (LAN Performance Features).



2.1	July 2012	• Added footnote to table 1.5 (LAN Performance Features). • Revised section 4.6.7.2 (Replaced "ITR Interval bit" with "ITR_INTERVAL bit" and "RSC Delay field" with "RSC_DELAY field". • Revised sections 4.6.11.3.1, 4.6.11.3.3, and 4.6.11.3.4 (removed LLTC references). • Section 5.3.2 (removed the statement directly above section 5.3.3). • Revised table 5.4 (Start-up and Power-State Transition Timing Parameters; tppg, tfl, and tpgres values). • Revised section 7.1.2.4 (replaced text "The receive packet is parsed and the OX_ID or RX_ID . . ."). • Revised section 8.2.4.22.20 (Flow Director Filters VLAN and FLEX bytes - FDIRVLAN (0x0000EE24) DBU-RX; bits 15:0). • Revised section 8.2.4.9.10 (Transmit Descriptor Control - TXDCTL[n] (0x00006028 + 0x40*n, n=0...127) DMA-TX; revised note from bit 25 description). • Revised section 7.1.2.7.11 (Query Filter Flow table). • Revised section 7.1.2.3 (ETQF flow) • Revised section 7.3.2.1.1 (Replaced "RSC Delay field" with "RSC_DELAY field"). • Revised figures 7.6 and 7.7. • Revised section 10.6.14 (Global Reserved Provisioning 1: Address 1E.C470; updated bits E:D description). • Revised section 10.4.21 (Auto-Negotiation Reserved Vendor Provisioning 1: Address 7.C410; updated bits F:E, A:8, and bits 7:6). • Revised section 10.6.19 (Global Cable Diagnostic Status 2: Address 1E.C801; bits 7:0). • Revised section 10.6.33 (Global Reserved Status 1: Address 1E.C885; removed XENPAK references). • Revised section 10.6.38 (Global Interrupt Mask 1: Address 1E.D400; changed bit E default to 1b). • Added new section 10.2.35 (PMA Receive Reserved Vendor State 1: Address 1.E810). • Added new section 10.2.36 (PMA Receive Reserved Vendor State 2: Address 1.E811). • Revised section 12.3.9 (Power On Reset). • Revised section 13.5.3.3 (Special Delay Requirements). • Revised section 13.8.1 (LAN Disable).
2.0	March 2012	• Revised section 2.1.10 (Miscellaneous; GPIO_7 description). • Revised section 3.5 (2nd bullet after Table 3.15; lowest SDP pins (SDP0_0 or SDP1_0) description). • Revised section 6.1 (added note about reserved fields). • Revised section 6.3.7.1 (PXE Setup Options PCI Function 0 — Word Address 0x30; bits 12:10 description). • Revised section 6.5.5.7 (NC-SI Configuration 1 - Offset 0x06; bits 4:0 description). • Revised section 6.5.5.8 (NC-SI Configuration 2 - Offset 0x07; bit 15 description). • Revised section 8.2.4.28.4 (Software Status Register; bit 8 description). • Revised section 8.2.4.25.13 (Priority XON Transmitted Count; bits 15:0 description). • Revised section 8.2.4.25.14 (Priority XON Received Count; bits 15:0 description). • Revised section 8.2.4.25.15 (Priority XOFF Transmitted Count; bits 15:0 description). • Revised section 8.2.4.25.16 (Priority XOFF Received Count; bits 15:0 description). • Revised table note references in section 11.7.2.2.3 (Read Status Command). • Revised section 6.2.1 (NVM Organization). • Revised section 8.2.4.23.1 (Core Control 0 Register; bit 1 description). • Revised section 8.2.4.4.14 (PCIe Control Extended Register; bit 30 description). • Revised section 8.2.4.8.9 (PCIe Control Extended Register; bit 1 description). • Revised section 8.2.4.23.10 (MAC Control Register; bits 7:5). • Removed PSRTYPE from note 11 in section 4.2.3.
1.9	January 2012	Initial public release.

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

1.0 Introduction

1.1 Scope

This document describes the external architecture (including device operation, pin descriptions, register definitions, etc.) for the Intel® Ethernet Controller X540, a single or 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 X540.

1.2 Product Overview

The X540 is a derivative of the 82599, the Intel 10 GbE Network Interface Controller (NIC) targeted for blade servers. Many features of its predecessor remain intact; however, some have been removed or modified as well as new features introduced.

The X540 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.

The X540 also supports a single port configuration.

1.2.1 System Configurations

The X540 is targeted for system configurations such as rack mounted or pedestal servers, where it can be used as an add-on NIC or LAN on Motherboard (LOM). Another system configuration is for high-end workstations.

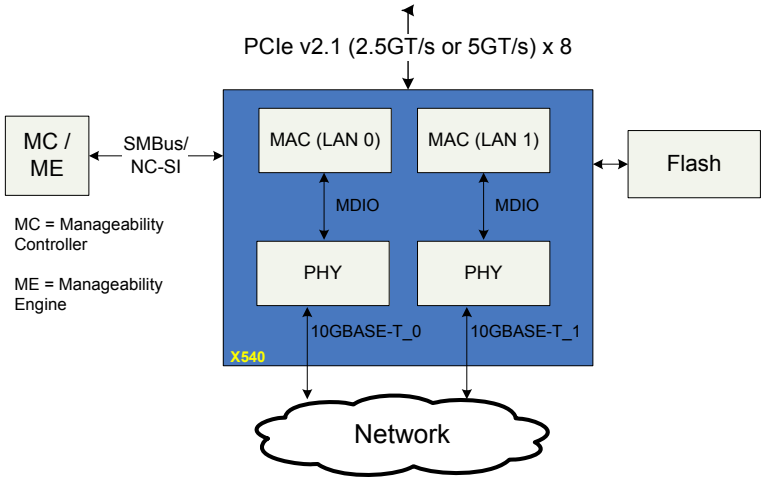


Figure 1-1 Typical Rack / Pedestal System Configuration

1.2.2 External Interfaces

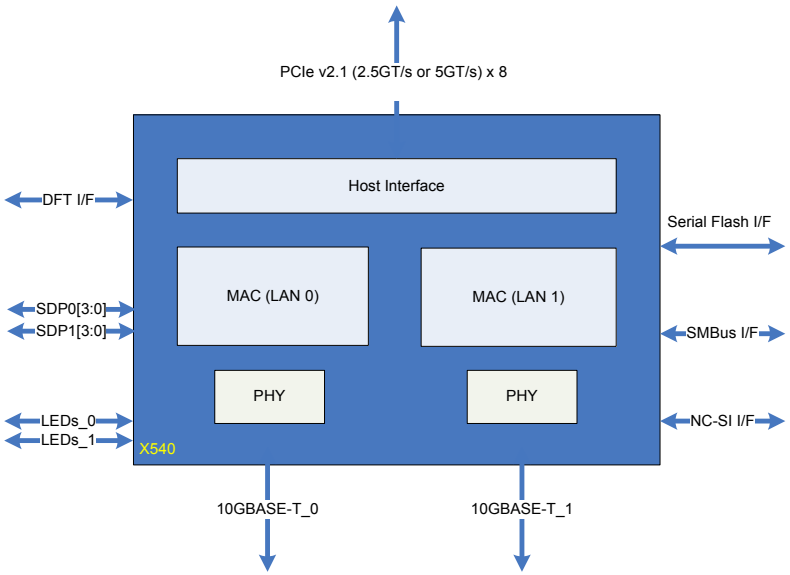


Figure 1-2 X540 External Interfaces Diagram (Dual Port)

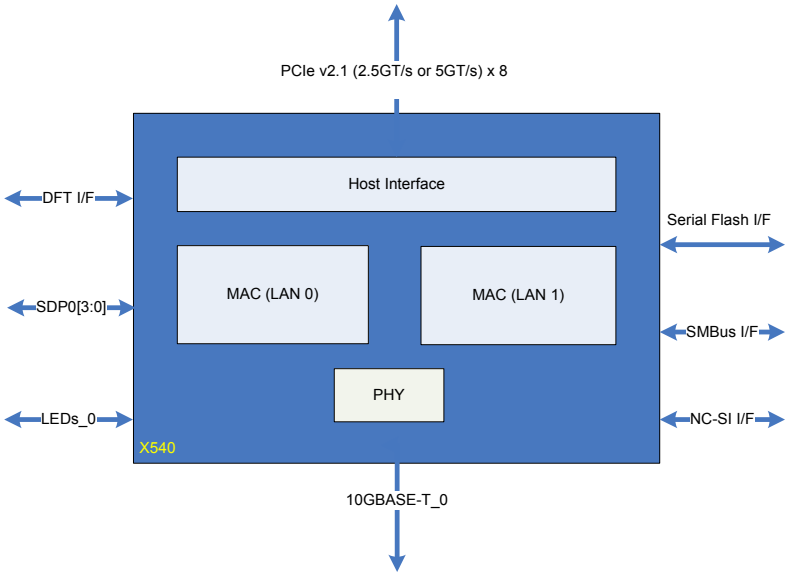


Figure 1-3 X540 External Interfaces Diagram (Single Port Configuration)

1.2.3 PCIe* Interface

The X540 supports PCIe v2.1 (2.5GT/s or 5GT/s). See Section 2.1.2 for full pin description and Section 12.4.7 for interface timing characteristics.

1.2.4 Network Interfaces

Two independent 10GBASE_T (10BASE-T_0 and 10GBASE-T_1) interfaces are used to connect the two the X540 ports to external devices. Each 10GBASE-T interface can operate at any of the following speeds:

- 10 Gb/s, 10GBASE-T mode
- 1 Gb/s, 1000BASE-T mode
- 100 Mb/s, 100BASE-TX mode

Refer to Section 2.1.3 for full-pin descriptions.For the timing characteristics of those interfaces, refer to the relevant external specifications listed in Section 12.4.8.

1.2.5 Serial Flash Interface

The X540 provides an external SPI serial interface to a Flash device, also referred to as Non-Volatile Memory (NVM). The X540 supports serial Flash devices with up to 16 Mb (2 MB) of memory.

1.2.6 SMBus Interface

SMBus is an optional interface for pass-through and/or configuration traffic between an external Manageability Controller (MC) and the X540.

The X540's SMBus interface supports a standard SMBus, up to a frequency of 400 KHz. Refer to Section 2.1.5 for full-pin descriptions and Section 12.4.6.3 for timing characteristics of this interface.

1.2.7 NC-SI Interface

NC-SI is an optional interface for pass-through traffic to and from an MC. The X540 meets the NC-SI version 1.0.0 specification.

Refer to Section 2.1.6 for the pin descriptions, and Section 11.7.1 for NC-SI programming.

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

The X540 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 X540 can support IEEE1588 auxiliary device connections, and other functionality. For more details on the SDPs see Section 3.5 and the ESDP register section.

1.2.9 LED Interface

The X540 implements four output drivers intended for driving external LED circuits per port. Each of the four LED outputs can be individually configured to select the particular event, state, or activity, which is indicated on that output. In addition, each LED can be individually configured for output polarity as well as for blinking versus non-blinking (steady-state) indications.

The configuration for LED outputs is specified via the LEDCTL register. In addition, the hardware-default configuration for all LED outputs can be specified via an NVM field (see Section 6.4.6.3), thereby supporting LED displays configured to a particular OEM preference.

1.3 Features Summary

Table 1-1 to Table 1-7 list the X540's features in comparison to previous dual-port 10 GbE Ethernet controllers.

Table 1-1 General Features

Feature	X540	82599	82598	Reserved
Serial Flash Interface	Y	Y	Y	
4-wire SPI EEPROM Interface	N	Y	Y	
Configurable LED Operation for Software or OEM Customization of LED Displays	Y	Y	Y	
Protected EEPROM/NVM[1] 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	31 x 31 mm	

Table 1-1 General Features

Feature	X540	82599	82598	Reserved
Embedded Thermal Diode	Y	N	Y	
Watchdog Timer	Y	Y	N	
Time Sync (IEEE 1588)	Y[2]	Y	N	

1. X540 Only.
2. Time sync not supported at 100 Mb/s link speed.

Table 1-2 Network Features

Feature	X540	82599	82598	Reserved
Compliant with the 10 GbE and 1 GbE Ethernet/ 802.3ap (KX/KX4) Specification	N	Y	Y	
Compliant with the 10 GbE 802.3ap (KR) specification	N	Y	N	
Support of 10GBASE-KR FEC	N	Y	N	
Compliant with the 10 GbE Ethernet/802.3ae (XAUI) Specification	N	Y	Y	
Compliant with XFI interface	N	Y	N	
Compliant with SFI interface	N	Y	N	
Support for EDC	N	N	N	
Compliant with the 1000BASE-BX Specification	N	Y	Y	
Auto Negotiation/Full-Duplex at 100 Mb/s Operation	Y (100 Mb/s FDX)	Y (100 Mb/s FDX)	NA	
10000/1000/100 Mb/s Copper PHYs Integrated On-Chip	Y	N	N	
Support Jumbo Frames of up to 15.5 KB	Y [1]	Y[1]	Y	
Auto-Negotiation Clause 73 for Supported Modes	N	Y	Y	
MDIO Interface Clause 45	Y (internally)	Y	Y	
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	N	Y	Y	

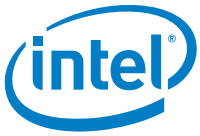


Table 1-2 Network Features

Feature	X540	82599	82598	Reserved
SGMII Interface	N	Y (100 Mb/s and 1 GbE only)	N	
Support of non Auto-Negotiation Partner	N	Y	Y	
Double VLAN	Y	Y	N	

1. The X540 and 82599 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 the X540 supports 9.5 KB jumbo packets. Packets to/from MC longer than 2KB are filtered out.

Table 1-3 Host Interface Features

Feature	X540	82599	82598	Reserved
PCIe* version (speed)	PCIe v2.1 (5GT/s)	PCIe v2.0 (2.5GTs & 5GT/s)	PCIe Gen 1 v2.0 (2.5GT/s)	
Number of Lanes	x1, x2, x4, x8	x1, x2, x4, x8	x1, x2, x4, x8	
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	4	
Credits for P-H/P-D/NP-H/NP-D (shared for the two ports)	16/16/4/4	16/16/4/4	8/16/4/4	
Max Payload Size Supported	512 Bytes	512 Bytes	256 Bytes	
Max Request Size Supported	2 KB	2 KB	256 Bytes	
Link Layer Retry Buffer Size (shared for the two ports)	3.4 KB	3.4 KB	2 KB	
Vital Product Data (VPD)	Y	Y	N	
End to End CRC (ECRC)	Y	Y	N	
TLP Processing Hints (TPH)	N	N	N	
Latency Tolerance Reporting (LTR)	N	N	N	
ID-Based Ordering (IDO)	N	N	N	
Access Control Services (ACS)	Y	N	N	
ASPM Optional Compliance Capability	Y	N	N	
PCIe Functions Off Via Pins, While LAN Ports Are On	Y	N	N	

Table 1-4 LAN Functions Features

Feature	X540	82599	82598	Reserved
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 & D3 States	Y	Y	Y	
Integrated MACsec, 801.2AE Security Engines: AES-GCM 128-bit; Encryption + Authentication; One SC x 2 SA Per Port. Replay Protection with Zero Window	Y	Y	N	
Integrated IPsec Security Engines: AES-GCM 128-bit; AH or ESP encapsulation; IPv4 and IPv6 (no option or extended headers)	1024 SA / port	1024 SA / port	N	
Software-Controlled Global Reset Bit (Resets Everything Except the Configuration Registers)	Y	Y	Y	
Software-Definable Pins (SDP) (per port)	4	8	8	
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 EEPROM/Flash[1]) Wake-up Flexible Filters	Y	Y	Y	
Default Configuration by EEPROM/Flash[1] 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	320 KB / port	
Programmable Memory Receive Buffers	384 KB / port	512 KB / port	512 KB / port	

1. X540 Only.

Table 1-5 LAN Performance Features[1]

Feature	X540	82599	82598	Reserved
TCP/UDP Segmentation Offload	256 KB in all modes	256 KB in all modes	256 KB in legacy mode, 32 KB in DCB	
TSO Interleaving for Reduced Latency	Y	Y	N	
TCP Receive Side Coalescing (RSC)	32 flows / port	32 flows / port	N	

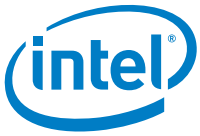


Table 1-5 LAN Performance Features[1]

Feature	X540	82599	82598	Reserved
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) N	
Rate Limit VM Tx Traffic per TC (per TxQ)	Y	Y	N	
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	N	
FCoE Transmit Segmentation	256 KB	256 KB	N	
FCoE Coalescing and Direct Data Placement	512 outstanding Read — Write requests / port	512 outstanding Read — Write requests / port	N	
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	
Multiple Rx Queues (RSS)	Y (multiple modes)	Y (multiple modes)	8x8 16x4	
Flow Director Filters: up to 32 KB -2 Flows by Hash Filters or up to 8 KB -2 Perfect Match Filters	Y	Y	N	
Number of Rx Queues (per port)	128	128	64	
Number of Tx Queues (per port)	128	128	32	
Low Latency Interrupts DCA Support TCP Timer Interrupts No Snoop Relax Ordering	Yes to all	Yes to all	Yes to all	
Rate Control of Low Latency Interrupts	Y	Y	N	

1. The X540 performance features are focused on 10 GbE performance improvement whereas 1 GbE was optimized for power saving.

Table 1-6 Virtualization Features

Feature	X540	82599	82598	Reserved
Support for Virtual Machine Device Queues (VMDq1 and Next Generation VMDq)	64	64	16	
L2 Ethernet MAC Address Filters (unicast and multicast)	128	128	16	
L2 VLAN filters	64	64	-	
PCI-SIG SR IOV	Y	Y	N	
Multicast and Broadcast Packet Replication	Y	Y	N	
Packet Mirroring	Y	Y	N	
Packet Loopback	Y	Y	N	
Traffic Shaping	Y	Y	N	

Table 1-7 Manageability Features

Feature	X540	82599	82598	Reserved
Advanced Pass Through-Compatible Management Packet Transmit/Receive Support	Y	Y	Y	
SMBus Interface to an External MC	Y	Y	Y	
NC-SI Interface to an External MC	Y	Y	Y	
New Management Protocol Standards Support (NC-SI)	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	4	
L3 Address Filters (IPv4)	4	4	4	
L3 Address Filters (IPv6)	4	4	4	
Host-Based Application-to-BMC Network Communication Patch (OS2BMC)	Y	N	N	
Flexible MAC Address	Y	N	N	
MC Inventory of LOM Device Information	Y	N	N	
iSCSI Boot Configuration Parameters via MC	Y	N	N	

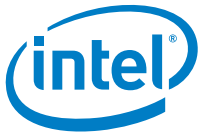


Table 1-7 Manageability Features

Feature	X540	82599	82598	Reserved
MC Monitoring	Y	N	N	
NC-SI to MC	Y	N	N	
NC-SI Arbitration	Y	N	N	
MCTP over SMBus[1]	Y	N	N	
NC-SI Package ID Via SDP Pins	Y	N	N	

1. The X540's MCTP protocol implementation is based on an early draft of the DSP0261 Standard and it includes a *Payload Type* field that was removed in the final release of the standard.

1.4 Overview of New Capabilities Beyond 82599

1.4.1 OS-to-BMC Management Traffic Communication (OS2BMC)

OS2BMC is a filtering method that enables server management software to communicate with a MC[1] via standard networking protocols such as TCP/IP instead of a chipset-specific interface. Functionality includes:

- A single PCI function (for multi-port devices, each LAN function enables communication to the MC)
- One or more IP address(es) for the host along with a single (and separate) IP address for the MC
- One or more host MAC address(es) along with a single (and separate) MAC address for the MC
- ARP/RARP/ICMP protocols supported in the MC

1.4.2 MCTP Over SMBus

Allow reporting and controlling of all the information exposed in a LOM device via NC-SI, in NIC devices via MCTP over SMBus.

MCTP is a transport protocol that does not provide a way to control a device. In order to allow a consistent interface for both LOM and NIC devices, it is planned to implement an NC-SI over MCTP protocol.

1. Also referred to as Baseboard Management Controller (BMC).

An Intel NIC can connect through MCTP to a MC. The MCTP interface will be used by the MC to control the NIC and not for pass-through traffic.

Note: The X540's MCTP protocol implementation is based on an early draft of the DSP0261 Standard and it includes a *Payload Type* field that was removed in the final release of the standard.

1.4.3 PCIe v2.1 Features

1.4.3.1 Access Control Services (ACS)

the X540 supports ACS Extended Capability structures on all functions. the X540 reports no support for the various ACS capabilities in the ACS Extended Capability structure. Further information can be found in Section 9.4.5.

1.4.3.2 ASPM Optionality Compliance Capability

A new capability bit, ASPM (Active State Power Management) Optionality Compliance bit has been added to the X540. Software is permitted to use the bit to help determine whether to enable ASPM or whether to run ASPM compliance tests. New bit indicates that the X540 can optionally support entry to L0s. Further information can be found in Section 9.3.11.7.

1.5 Conventions

1.5.1 Terminology and Acronyms

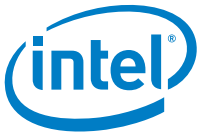
See Section 17.0.

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

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

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

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



1.6 References

The X540 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 802.3-2005 (Ethernet). Incorporates various IEEE Standards previously published separately. Institute of Electrical and Electronic Engineers (IEEE).
- 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.

PCI-SIG Specifications

- PCI Express® Base Specification Revision 2.1, March 4, 2009
- PCI Express 2.1 Card Electromechanical Specification
- PCI Express 2.0 Base specification, 12/20/2006.
- PCI Express™ 2.0 Card Electromechanical Specification, Revision 0.9, January 19, 2007.
- 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 Specification 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)
- RFC4106 — The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP).
- RFC4302 — IP Authentication Header (AH)
- RFC4303 — IP Encapsulating Security Payload (ESP)

- RFC4543 — The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH.
- IETF Internet Draft, Marker PDU Aligned Framing for TCP Specification.
- IETF Internet Draft, Direct Data Placement over Reliable Transports.
- IETF Internet Draft, RDMA Protocol Specification.

Other

- Advanced Configuration and Power Interface Specification, Rev 2.0b, October 2002
- RDMA Consortium, RDMA Protocol Verbs Specification
- Network Controller Sideband Interface (NC-SI) Specification, Version cPubs-0.1, 2/18/2007.
- System Management Bus (SMBus) Specification, SBS Implementers Forum, Ver. 2.0, August 2000.
- EUI-64 specification, http://standards.ieee.org/regauth/oui/tutorials/EUI64.html.
- Backward Congestion Notification Functional Specification, 11/28/2006.
- 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. http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf, January 2004.
- 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

In addition, the following document provides application information:

- 82563EB/82564EB Gigabit Ethernet Physical Layer Device Design Guide, Intel Corporation.

1.7 Architecture and Basic Operation

1.7.1 Transmit (Tx) Data Flow

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

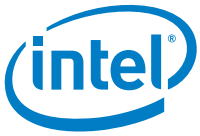


Table 1-8 Tx Data Flow

Step	Description
1	The host creates a descriptor ring and configures one of the X540's transmit queues with the address location, length, head, and tail pointers of the ring (one of 128 available Tx queues).
2	The host is requested by the TCP/IP stack to transmit a packet, it gets the packet data within one or more data buffers.
3	The host 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 at the appropriate Tx ring.
4	The host updates the appropriate Queue Tail Pointer (TDT).
5	The X540'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 being 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 MAC.
11	The MAC appends the L2 CRC to the packet and delivers the packet to the integrated PHY.
12	The PHY performs the PCS encoding, scrambling, Loopback Dropped Packet Count (LDPC) encoding, and the other manipulations required to deliver the packet over the copper wires at the selected speed.
13	When all the PCIe completions for a given packet are complete, the DMA updates the appropriate descriptor(s).
14	The descriptors are written back to host memory using PCIe posted writes. The head pointer is updated in host memory as well.
15	An interrupt is generated to notify the host driver that the specific packet has been read to the X540 and the driver can then release the buffer(s).

1.7.2 Receive (Rx) Data Flow

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

Table 1-9 Rx Data Flow

Step	Description
1	The host creates a descriptor ring and configures one of the X540's receive queues with the address location, length, head, and tail pointers of the ring (one of 128 available Rx queues).
2	The host initializes descriptor(s) that point to empty data buffer(s). The host places these descriptor(s) in the correct location at the appropriate Rx ring.
3	The host 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, descrambling, PCS decoding, etc.
6	The PHY delivers the packet to the Rx MAC.
7	The MAC forwards the packet to the Rx filter.
8	If the packet matches the pre-programmed criteria of the Rx filtering, it is forwarded to an Rx FIFO.
9	The receive DMA fetches the next descriptor from the appropriate host memory ring to be used for the next received packet.
10	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.
11	When the packet is placed into host memory, the receive DMA updates all the descriptor(s) that were used by the packet data.
12	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.
13	The X540 initiates an interrupt to the host to indicate that a new received packet is ready in host memory.
14	The host reads the packet data and sends it to the TCP/IP stack for further processing. The host releases the associated buffer(s) and descriptor(s) once they are no longer in use.

2.0 Pin Interface

2.1 Pin Assignments

2.1.1 Signal Type Definition

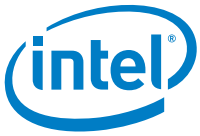
Signal	Definition	DC Specification
In	Standard 2.5V I/O buffer, functions as input-only signal. 3.3V tolerance.	
Out (O)	Standard 2.5V I/O buffer, functions as output-only signal. 3.3V tolerance.	
T/s	Tri-state is a 2.5V bi-directional, tri-state input/output pin. 3.3V tolerance.	
O/d	Open drain enables multiple devices to share as a wire-OR.	Section 12.4.3
A-in	Analog input signals.	Section 12.4.6 and Section 12.4.7
A-out	Analog output signals.	Section 12.4.6 and Section 12.4.7
A-Inout	Bi-directional analog signals.	
B	Input BIAS.	
NCSI-in	NC-SI 3.3V input signal.	Section 12.4.4
NCSI-out	NC-SI 3.3V output signal.	Section 12.4.4
In-1p2	1.2V input-only signal. 3.3V tolerance.	
In-Only	Standard 2.5V buffer input-only signal. 3.3V tolerance.	
Out-Only	Standard 2.5V buffer output-only signal.	

Signal	Definition	DC Specification
LVDS-O	Low voltage differential signal - output.	
Pup	Pull up.	
Pdn	Pull down.	

2.1.2 PCIe

See AC/DC specifications in Section 12.4.6.

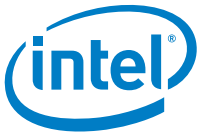
Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	PET_0_p PET_0_n	AC3 AD3	A-Out			PCIe Serial Data Output. A serial differential output pair running at 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end.
	PET_1_p PET_1_n	AC4 AD4	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.
	PET_2_p PET_2_n	AC9 AD9	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.
	PET_3_p PET_3_n	AC10 AD10	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.
	PET_4_p PET_4_n	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.



Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	PET_5_p PET_5_n	AC16 AD16	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.
	PET_6_p PET_6_n	AC21 AD21	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.
	PET_7_p PET_7_n	AC22 AD22	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.
	PER_0_p PER_0_n	AB2 AB1	A-In			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.
	PER_1_p PER_1_n	AD6 AC6	A-In			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.
	PER_2_p PER_2_n	AD7 AC7	A-In			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.
	PER_3_p PER_3_n	AD12 AC12	A-In			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.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	PER_4_p PER_4_n	AD13 AC13	A-In			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.
	PER_5_p PER _5_n	AD18 AC18	A-In			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.
	PER _6_p PER _6_n	AD19 AC19	A-In			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.
	PER _7_p PER _7_n	AB23 AB24	A-In			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.
	PE_CLK_p PE_CLK_n	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.
	PE_RBIAS0	V1	A-Inout			Connection point for the band-gap reference resistor. This should be a precision 1% 3.01 KΩ resistor tied to ground.
	PE_RBIAS1	V2	A-Inout			Connection point for the band-gap reference resistor. This should be a precision 1% 3.01 KΩ resistor tied to ground.
	PE_WAKE_N	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	W2	In			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.



2.1.3 MDI

See AC/DC specifications in Section 12.4.7.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	MDI0_p_0	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_n_0	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_p_1	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_n_1	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_p_2	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_n_2	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_p_3	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_n_3	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_p_4	A11	A-Inout			Port 0 Analog Test+. Connects to the pair E+ input of the transformer.
	MDI0_n_4	B11	A-Inout			Port 0 Analog Test-. Connects to the pair E- input of the transformer.
	MDI1_p_0[1]	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.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	MDI1_n_0[1]	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_p_1[1]	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_n_1[1]	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_p_2[1]	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_n_2[1]	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_p_3[1]	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_n_3[1]	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_p_4[1]	A14	A-Inout			Port 1 Analog Test+. Connects to the pair E+ input of the transformer.
	MDI1_n_4[1]	B14	A-Inout			Port 1 Analog Test-. Connects to the pair E- input of the transformer.
	BG_REXT	D12	A-Inout			Connection point for the band-gap reference resistor. Should be a precision 1% 2 K Ω resistor tied to ground.
	TM_REXT	C12	A-Inout			Connection point for the band-gap reference resistor. Should be a precision 1% 140 Ω resistor tied to 2.5V.
	XTAL_I	D23	A-In			Positive 50.0 MHz crystal oscillator input.
	XTAL_O	D24	A-Out			Positive 50.0 MHz crystal oscillator output.

1. These pins are a No Connect for the the X540 single port configuration.

2.1.4 Serial Flash

See AC/DC specifications in Section 12.4.5.4.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	FLSH_SI	K2	Out			Serial data output to the Flash.
	FLSH_SO	K1	In	Pup		Serial data input from the Flash.
	FLSH_SCK	J1	Out			Flash serial clock. Operates at the maximum frequency of 25 MHz.
	FLSH_CE_N	J2	Out		Pup[1]	Flash chip select output.

1. Pup value should be considered as 3.3 KΩ.

2.1.5 SMBus

See the AC/DC specifications in Section 12.4.5.3.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	SMBCLK	L2	O/d		Pup[1]	SMBus Clock. One clock pulse is generated for each data bit transferred.
	SMBD	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	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.

2.1.6 NC-SI

See AC specifications in Section 12.4.5.5.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	NCSI_CLK_ IN	G2	NCSI-In		Pdn[1]	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	G4	NCSI-In		Pdn[1]	MC Transmit Enable. Indicates that received data from MC is valid.
	NCSI_TXD0 NCSI_TXD1	H2 G3	NCSI-In		Pup[2]	MC Transmit Data. Data signals from the MC to the X540.
	NCSI_CRS_ DV	H1	NCSI-Out		Pdn[1]	Carrier Sense/Receive Data Valid (CRS/ DV) to MC. Indicates that the data transmitted from the X540 to MC is valid.
	NSCI_RXD0 NCSI_RXD1	H3 G1	NCSI-Out		Pup[2]	MC Receive Data. Data signals from the X540 to the MC.
	NCSI_ARB_ IN	F1	NCSI-In		Pdn[1]	NC-SI Arbitration In.
	NCSI_ARB_ OUT	F2	NCSI-Out			NC-SI Arbitration Out.

1. Pdn value should be considered as 10 KΩ.
2. Pup value should be considered as 10 KΩ.

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

2.1.7 Software Defined Pins (SDPs)

See AC specifications in Section 12.4.5.1.

See Section 3.5 for more details on configurable SDPs.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/ Pdn[1]	Name and Function
	SDP0_0 SDP0_1 SDP0_2 SDP0_3	R4 P3 T4 R3	T/s			General Purpose SDPs. 2.5V I/Os for function 0. Can be used to support IEEE1588 auxiliary devices. Input for external interrupts, PCIe function disablement, etc. See Section 1.6 for possible usages of the pins.
	SDP1_0[2] SDP1_1[2] SDP1_2[2] SDP1_3[2]	T21 T22 U21 U22	T/s			General Purpose SDPs. 2.5V I/Os for function 1. Can be used to support IEEE1588 auxiliary devices. Input for external interrupts, PCIe function disablement, etc. See Section 1.6 for possible usages of the pins.

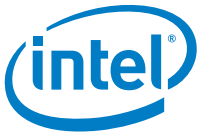
1. SDP pins should have external Pup/Pdn or other board connectivity according to board implementation.
2. These pins are reserved and should be left as No Connect for the the X540 single port configuration.

2.1.8 LEDs

See AC specifications in Section 12.4.5.1.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	LED0_0	H4	Out	Pdn		Port 0 LED0. Programmable LED. By default, indicates link up.
	LED0_1	J3	Out	Pdn		Port 0 LED1. Programmable LED. By default, indicates 10 Gb/s link.
	LED0_2	J4	Out	Pdn		Port 0 LED2. Programmable LED. By default, indicates link/activity.
	LED0_3	K4	Out	Pdn		Port 0 LED3. Programmable LED. By default, indicates 1 Gb/s link.
	LED1_0[1]	J21	Out	Pdn		Port 1 LED0. Programmable LED. By default, indicates link up.
	LED1_1[1]	J22	Out	Pdn		Port 1 LED1. Programmable LED. By default, indicates 10 Gb/s link.
	LED1_2[1]	K21	Out	Pdn		Port 1 LED2. Programmable LED. By default, indicates link/activity.
	LED1_3[1]	K22	Out	Pdn		Port 1 LED3. Programmable LED. By default, indicates 1 Gb/s link.

1. These pins are reserved and should be left as No Connect for the the X540 single port configuration.



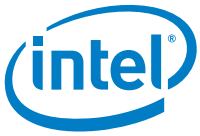
2.1.9 RSVD and No Connect Pins

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.

Reserved	Pin Name	Ball #	Type	Name and Function
	RSVDH22_VSS	H22	In-Only	Reserved/VSS pins.
	RSVDD14_NC	D14	A-Inout	Reserved/no connect pin.
	RSVDG24_VSS	G24	In	Reserved/VSS pins.
	RSVDF4_NC RSVDF3_NC	F4 F3	A-Inout A-Inout	Reserved/no connect pins.
	RSVDD1_NC RSVDE24_NC	D1 E24	A-Inout A-Inout	Reserved/no connect pins.
	RSVDE1_NC RSVDE23_NC	E1 E23	A-Inout A-Inout	Reserved/no connect pins.
	RSVDC1_NC RSVDF24_NC	C1 F24	A-Inout A-Inout	Reserved/no connect pins.
	RSVDV3_NC RSVDV4_NC	V3 V4	A-Inout A-Inout	Reserved/no connect pins.
	RSVDL4_NC RSVDL3_NC RSVDL21_NC RSVDM21_NC RSVDL22_NC RSVDN21_NC RSVDM22_NC RSVDP21_NC RSVDN22_NC RSVDR21_NC RSVDP22_NC RSVDM4_NC RSVDM3_NC RSVDN4_NC RSVDN3_NC RSVDP4_NC	L4 L3 L21 M21 L22 N21 M22 P21 N22 R21 P22 M4 M3 N4 N3 P4	Out Out Out Out Out Out Out Out Out Out Out Out Out Out Out Out	Reserved/no connect pins.
	RSVDR22_NC	R22	Out	Reserved/no connect pin.

Reserved	Pin Name	Ball #	Type	Name and Function
	RSVDAA6_NC RSVDAA8_NC RSVDAA10_NC	AA6 AA8 AA10	PWR	Reserved/no connect pins.
	RSVDAA14_NC RSVDAA16_NC RSVDAA18_NC	AA14 AA16 AA18	PWR	Reserved/no connect pins.
	RSVDU4_NC	U4	PWR	Reserved no connect pin.
	RSVDG11_NC	G11	PWR	Reserved no connect pin.
	RSVDU3_NC RSVDN2_NC RSVDU2_NC RSVDV24_NC RSVDU24_NC RSVDU23_NC RSVDR1_NC RSVDP1_NC RSVDU1_NC RSVDT1_NC	U3 N2 U2 V24 U24 U23 R1 P1 U1 T1	T/s T/s T/s T/s T/s T/s T/s T/s T/s T/s	Reserved no connect pins.
	RSVDT23_VSS	T23	In-Only	Reserved VSS pin.
	RSVDP23_VSS	P23	In-Only	Reserved VSS pin.
	RSVDA24_VSS	A24	In-Only	Reserved VSS pin.
	RSVDAD24_VSS	AD24	In-Only	Reserved VSS pins.
	RSVDN23_VSS RSVDN24_VSS RSVDP24_VSS	N23 N24 P24	In-Only In-Only In-Only	Reserved VSS pins.
	RSVDY21_VSS	Y21	In-Only	Reserved VSS pin.
	RSVDT24_VSS	T24	In-Only	Reserved VSS pin.
	RSVDL23_NC	L23	O/d	Reserved No Connect pin.
	RSVDJ23_VSS	J23	In-Only	Reserved VSS pin.
	RSVDJ24_VSS	J24	In	Reserved VSS pin.
	RSVDW4_VSS	W4	In	Reserved VSS pin.
	RSVDH24_VSS	H24	In	Reserved VSS pin.



Reserved	Pin Name	Ball #	Type	Name and Function
	RSVDG23_VSS	G23	In-Only	Reserved VSS pin.
	RSVDY4_VSS	Y4	In	Reserved VSS pin.
	RSVDAA24_VSS	AA24	In-Only	Reserved VSS pin.
	RSVDAD1_VSS	AD1	In	Reserved VSS pin.
	RSVDD13_NC RSVDC13_NC RSVDC11_NC RSVDD11_NC RSVDA1_NC RSVDV21_VSS RSVDW24_VSS RSVDY20_NC RSVDY5_NC RSVDN13_NC RSVDM12_NC	D13 C13 C11 D11 A1 V21 W24 Y20 Y5 N13 M12	A-Inout A-Inout A-Out A-Out In-Only In-Only A-Inout A-Inout PWR-O PWR-O PWR-O	Reserved no connect/VSS pins.
	RSVDR24_NC RSVDR23_NC RSVDN1_NC	R24 R23 N1	LVDS-O LVDS-O Out-Only	Reserved no connect pins.
	RSVDM24_VSS	M24	In	Reserved VSS pin.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	RSVDT2_VCC2P5	T2	In	Pdn	Pup[1]	Reserved VCC2P5 pin.
	RSVDM23_VCC2P5	M23	In	Pup	Pup[1]	Reserved VCC2P5 pin.
	RSVDK3_VSS	K3	In	Pdn	Pdn[1]	Reserved VSS pin.

1. Pup value should be considered as 3.3 KΩ.

2.1.10 Miscellaneous

See AC/DC specifications in Section 12.4.5.1.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	LAN_PWR_GOOD	L24	In-1p2	Pup	Pup[1]	LAN Power Good. A transition from low-to-high initializes the X540 into operation. If not used (BYPASS_POR = 0b), an internal Power On Reset (POR) circuit triggers the X540 power up sequence.
	BYPASS_POR	H23	In	Pdn	Pdn[2]	Reserved.
	AUX_PWR	P2	In		Note[3]	Auxiliary Power Available. When set, indicates that auxiliary power is available and the X540 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	R2	In		Note[4]	Main Power Good. Indicates that platform main power is up. Must be connected externally.
	LAN1_DIS_N	K24	In	Pup	Pup[1]	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	K23	In	Pup	Pup[1]	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. When LAN 0 port is disabled manageability is not functional and it must not be enabled in the NVM Control Word 1.
	SEC_EN	M1	In	Pup	Pup[1]	Enable/Disable for the internal MACsec/IPSec engines.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	THERM_D1_P THERM_D1_N	G21 G22	A-Inout A-Inout			Thermal Diode Reference. Can be used to measure on-die temperature.
	PHY0_RVSL	T3	T/s	Pup	Note[5]	Pin change order of MDI lanes port 0: 0b = Lane order A, B, C, D. 1b = Lane order D, C, B, A.
	PHY1_RVSL	V23	T/s	Pup	Note[5]	Pin change order of MDI lanes port 1: 0b = Lane order A, B, C, D. 1b = Lane order D, C, B, A.

1. Pup value should be considered as 10 KΩ.
2. Pdn value should be considered as 10 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 10 KΩ.
4. Connect MAIN_PWR_OK signal to Main Power through Pup resistor. Pup value should be considered as 10 KΩ.
5. For pin change order A, B, C, and D, connect PHY_RVSL signal to Pdn. For pin change order D, C, B, and A, connect PHY_RVSL signal to Pup. Pup value should be considered as 10 KΩ. Pdn value should be considered as 3.3 KΩ.

2.1.11 JTAG

See AC specifications in Section 12.4.5.2.

Reserved	Pin Name	Ball #	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
	TCK	Y22	In-Only	Pup	Pdn[1]	JTAG Clock Input.
	TDI	W22	In-Only	Pup	Pup[2]	JTAG Data Input.
	TDO	V22	Out		Pup[3]	JTAG Data Output.
	TMS	W21	In-Only	Pup	Pup[2]	JTAG TMS Input.
	TRST_N	W23	In-Only	Pup	Pdn[1]	JTAG Reset Input. Active low reset for the JTAG port.

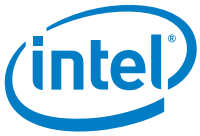
1. Pdn value should be considered as 470 Ω.
2. Pup value should be considered as 10 KΩ.
3. Pup value should be considered as 3.3 KΩ

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

2.1.12 Power Supplies

See AC specifications in Section 12.3.1.

Reserved	Pin Name	Ball #	Type	Name and Function
	RSVDH21_ VSS	H21	PWR	Reserved power pin.
	VSS	B1, B10, B12, B13, B15, B17, B19,B2, B21, B23, B24, B4, B6, B8, C14, C16, C18, C20, C22, C3, C5, C7, C9, D15, D17, D19, D2, D21, D22, D3, D5, D7, D9, E10, E12, E14, E16, E18, E2, E20, E22, E4, E6, E8, F21, F22, F23	PWR _ALG	Ground
	VSS	AA1, AA11, AA12, AA13, AA15, AA17, AA19, AA2, AA20, AA21, AA22, AA23, AA3, AA4, AA5, AA7, AA9, AB10, AB12, AB13, AB15, AB16, AB18, AB19, AB21, AB4, AB6, AB7, AB9, AC1, AC11, AC14, AC17, AC2, AC20, AC23, AC24, AC5, AC8, AD11, AD14, AD17, AD2, AD20, AD23, AD5, AD8, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F5, F6, F7, F8, F9, G10, G12, G14, G16, G18, G20, G6, G8, H11, H13, H15, H17, H19, H5, H7, H9, J10, J12, J14, J16, J18, J20, J6, J8, K11, K13, K15, K17, K19, K5, K7, K9, L10, L12, L14, L16, L18, L20, L6, L8, M11, M13, M15, M17, M19, M5, M7, M9, N10, N12, N14, N16, N18, N20, N6, N8, P11, P13, P15, P17, P19, P5, P7, P9, R10, R12, R14, R16, R18, R20, R6, R8, T11, T13, T15, T17, T19, T5, T7, T9,U10, U12, U14, U16, U18, U20, U6, U8, V11, V13, V15, V17, V19, V5, V7, V9, W10, W12, W14, W16, W18, W20, W3, W6, W8, Y11, Y13, Y15, Y17, Y19, Y23, Y24, Y3, Y7, Y9	PWR	Ground
	VCC0P67	G5, G7, G9, H10, H12, H6, H8, J11, J5, J7, J9, K10, K12, K6, K8, L11, L7, L9, M10, M8, N11, N7, N9, P10, P12, P8, R11, R7, R9, T10, T12, T8, U11, U7, U9, V10, V12, V8, W11, W7, W9	PWR	0.67V
	VCC0P8	G13, G15, G17, G19, H14, H16, H18, H20, J13, J15, J17, J19, K14, K16, K18, K20, L13, L15, L17, L19, M14, M16, M18, N15, N17, P14, P16, P18, R13, R15, R17, T14, T16, T18, U13, U15, U17, V14, V16, V18, W13, W15, W17	PWR	0.8V
	VCC1P2	D10, D4, D8, E11, E7, E9, D6, E5	PWR _ALG	1.2V
	VCC1P2	E19, D20, D16,D18, E13,E15, E17	PWR _ALG	1.2V



Reserved	Pin Name	Ball #	Type	Name and Function
	VCC2P5	A2, C2, A4, C4, A6, C6, A8, C8, A10, C10, A12	PWR_ALG	2.5V
	VCC2P5	A13, A15, C15, A17, C17, A19, C19, A21, C21, A23, C23	PWR_ALG	2.5V
	VCC1P2	E21	PWR_ALG	1.2V
	VCC1P2	E3	PWR_ALG	1.2V
	VCC3P3	L5, M6, N5	PWR	3.3V
	VCC1P2	Y10, Y12, Y14, Y16, Y18, Y8, Y6	PWR	1.2V
	VCC2P5	AB11, AB14, AB17, AB20, AB3 AB5 AB8,AB22	PWR	2.5V
	VCC2P5	C24	PWR_ALG	2.5V
	VCC2P5	M20, N19, V20, P20, P6, R19, R5, T20, T6, U19, U5, V6, W19, W5	PWR	2.5V

2.2 Ball Out — Top View Through Package

	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	VCC2P5	MDI0_p_0	VCC2P5	MDI0_p_1	VCC2P5	MDI0_p_2	VCC2P5	MDI0_p_3	VCC2P5	MDI0_p_4	VCC2P5	VCC2P5	MDI1_p_4	VCC2P5	MDI1_p_3	VCC2P5	MDI1_p_2	VCC2P5	MDI1_p_1	VCC2P5	MDI1_p_0	VCC2P5	RSVDA24_VSS	A
B	VSS	VSS	MDI0_n_0	VSS	MDI0_n_1	VSS	MDI0_n_2	VSS	MDI0_n_3	VSS	MDI0_n_4	VSS	VSS	MDI1_n_4	VSS	MDI1_n_3	VSS	MDI1_n_2	VSS	MDI1_n_1	VSS	MDI1_n_0	VSS	VSS	B
C	RSVDC1_NC	VCC2P5	VSS	VCC2P5	VSS	VCC2P5	VSS	VCC2P5	VSS	VCC2P5	RSVDC11_NC	TM_REXT	RSVDC13_NC	VSS	VCC2P5	VSS	VCC2P5	VSS	VCC2P5	VSS	VCC2P5	VSS	VCC2P5	VCC2P5	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	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	RSVDE23_NC	RSVDE24_NC	E
F	NCSI_ARB_IN	NCSI_ARB_OUT	RSVDF3_NC	RSVDF4_NC	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	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	RSVDG11_NC	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	Therm_D1_P	Therm_D1_N	RSVDG23_VSS	RSVDG24_VSS	G
H	NCSI_CRS_DV	NCSI_TXD0	NCSI_RXD0	LED0_0	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	RSVDH21_VSS	RSVDH22_VSS	RSVDH23	RSVDH24_VSS	H
J	FLSH_SCK	FLSH_CE_N	LED0_1	LED0_2	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	LED1_0	LED1_1	RSVDAJ23_VSS	RSVDJ24_VSS	J
K	FLSH_SO	FLSH_SI	RSVDK3_VSS	LED0_3	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	LED1_2	LED1_3	LAN0_DIS_N	LAN1_DIS_N	K
L	SMBD	SMBCLK	RSVDL3_NC	RSVDL4_NC	VCC3P3	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	RSVDL21_NC	RSVDL22_NC	RSVDL23_NC	LAN_PWR_GOOD	L
M	SEC_EN	SMBALRT_N	RSVDM3_NC	RSVDM4_NC	VSS	VCC3P3	VSS	VCC0P67	VSS	VCC0P67	VSS	RSVDM12_NC	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC2P5	RSVDM21_NC	RSVDM22_NC	RSVDM23_VCC2P5	RSVDM24_VSS	M
N	RSVDN1_NC	RSVDN2_NC	RSVDN3_NC	RSVDN4_NC	VCC3P3	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	RSVDN13_NC	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC2P5	VSS	RSVDN21_NC	RSVDN22_NC	RSVDN23_VSS	RSVDN24_VSS	N
P	RSVDP1_NC	AUX_PWR	SDP0_1	RSVDP4_NC	VSS	VCC2P5	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC2P5	RSVDP21_NC	RSVDP22_NC	RSVDP23_VSS	RSVDP24_VSS	P
R	RSVDR1_NC	MAIN_PWR_OK	SDP0_3	SDP0_0	VCC2P5	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC2P5	VSS	RSVDR21_NC	RSVDR22_NC	RSVDR23_NC	RSVDR24_NC	R
T	RSVDT1_NC	RSVDT2_VCC2P5	PHY0_RVSL	SDP0_2	VSS	VCC2P5	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC2P5	SDP1_0	SDP1_1	RSVDT23_VSS	RSVDT24_VSS	T
U	RSVDU1_NC	RSVDU2_NC	RSVDU3_NC	RSVDU4_NC	VCC2P5	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC2P5	VSS	SDP1_2	SDP1_3	RSVDU23_NC	RSVDU24_NC	U
V	PE_RBIAS0	PE_RBIAS1	RSVDV3_NC	RSVDV4_NC	VSS	VCC2P5	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC2P5	RSVDV21_VSS	TDO	PHY1_RVSL	RSVDV24_NC	V
W	PE_WAKE_N	PE_RST_N	VSS	RSVDW4_VSS	VCC2P5	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P67	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC0P8	VSS	VCC2P5	VSS	TMS	TDI	TRST_N	RSVDW24_VSS	W
Y	PE_CLK_n	PE_CLK_p	VSS	RSVDY4_VSS	RSVDY5_NC	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	VCC1P2	VSS	RSVDY20_NC	RSVDY21_VSS	TCK	VSS	VSS	Y
AA	VSS	VSS	VSS	VSS	VSS	RSVDAA8_NC	VSS	RSVDAA8_NC	VSS	RSVDAA10_NC	VSS	VSS	VSS	RSVDAA14_NC	VSS	RSVDAA16_NC	VSS	RSVDAA18_NC	VSS	VSS	VSS	VSS	VSS	RSVDAA24_VSS	AA
AB	PER_0_n	PER_0_p	VCC2P5	VSS	VCC2P5	VSS	VSS	VCC2P5	VSS	VSS	VCC2P5	VSS	VSS	VCC2P5	VSS	VSS	VCC2P5	VSS	VSS	VCC2P5	VSS	VCC2P5	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_VSS	AD
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	

Figure 2-1 X540 Package Layout

3.0 Interconnects

3.1 PCI Express* (PCIe*)

3.1.1 Overview

PCIe is an I/O architecture that enables cost competitive solutions as well as provide industry leading price/performance and feature richness. It is an industry-driven specification.

PCIe defines a basic set of requirements that addresses the majority of the targeted application classes. Higher-end applications' requirements (Enterprise class servers and high-end communication platforms) are addressed by a set of advanced extensions that compliment the baseline requirements.

To guarantee headroom for future applications, PCIe provides a software-managed mechanism for introducing new, enhanced capabilities.

Figure 3-1 shows the PCIe architecture.

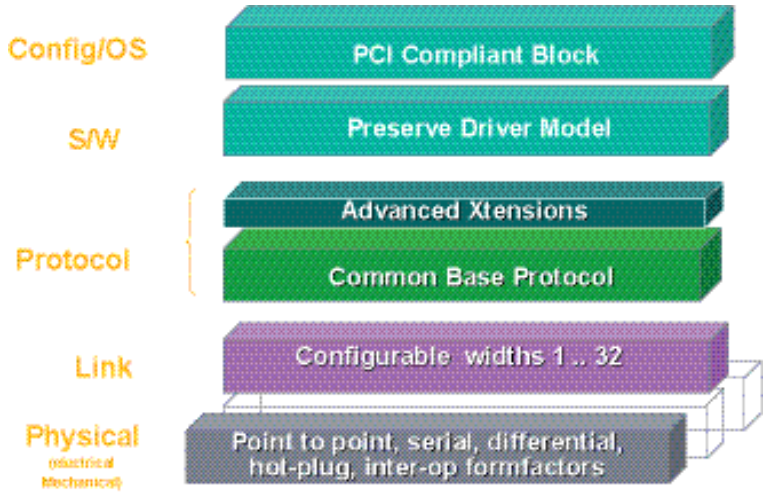


Figure 3-1 PCIe Stack Structure

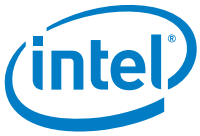
The PCIe physical layer consists of a differential transmit pair and a differential receive pair. Full-duplex data on these two point-to-point connections is self-clocked such that no dedicated clock signals are required. The bandwidth of this interface increases in direct proportion with frequency increases.

The packet is the fundamental unit of information exchange and the protocol includes a message space to replace a variety of side-band signals found on previous interconnects. This movement of hard-wired signals from the physical layer to messages within the transaction layer enables easy and linear physical layer width expansion for increased bandwidth.

The common base protocol uses split transactions along with several mechanisms to eliminate wait states and to optimize the re-ordering of transactions to further improve system performance.

3.1.1.1 Architecture, Transaction and Link Layer Properties

- Split transaction, packet-based protocol
- Common flat address space for load/store access (for example, PCI addressing model)
  - 32-bit memory address space to enable a compact packet header (must be used to access addresses below 4 GB)
  - 64-bit memory address space using an extended packet header
- Transaction layer mechanisms:
  - PCI-X style relaxed ordering
  - Optimizations for no-snoop transactions
- Credit-based flow control
- Packet sizes/formats:
  - Maximum packet size: 512 bytes
  - Maximum read request size: 2 KB
- Reset/initialization:
  - Frequency/width/profile negotiation performed by hardware
- Data integrity support
  - Using CRC-32 for Transaction layer Packets (TLP)
- Link Layer Retry (LLR) for recovery following error detection
  - Using CRC-16 for Link Layer (LL) messages
- No retry following error detection
  - 8b/10b encoding with running disparity



- Software configuration mechanism:
  - Uses PCI configuration and bus enumeration model
  - PCIe-specific configuration registers mapped via PCI extended capability mechanism
- Baseline messaging:
  - In-band messaging of formerly side-band legacy signals (interrupts, etc.)
  - System-level power management supported via messages
- Power management:
  - Full support for PCIm
  - Wake capability from D3cold state
  - Compliant with ACPI, PCIm software model
  - Active state power management
- Support for PCIe Gen 1 v2.0 (2.5GT/s) or PCIe Gen2 v1.0 (5GT/s)
  - Support for completion time out control

3.1.1.2 Physical Interface Properties

- Point-to-point interconnect
  - Full-duplex; no arbitration
- Signaling technology:
  - Low Voltage Differential (LVD)
  - Embedded clock signaling using 8b/10b encoding scheme
- Serial frequency of operation: PCIe Gen 1 v2.0 (2.5GT/s) or PCIe Gen2 v1.0 (5GT/s)
- Interface width of 1, 2, 4, or 8 PCIe lanes
- DFT and DFM support for high-volume manufacturing

3.1.1.3 Advanced Extensions

PCIe defines a set of optional features to enhance platform capabilities for specific usage modes. The X540 supports the following optional features:

- Advanced Error Reporting (AER) — Messaging support to communicate multiple types/severity of errors
- Device Serial Number — Allows exposure of a unique serial number for each device
- Alternative RID Interpretation (ARI) — allows support of more than eight functions per device
- Single Root I/O Virtualization (SR-IOV) — allows exposure of virtual functions controlling a subset of the resources to Virtual Machines (VMs)

3.1.2 General Functionality

3.1.2.1 Native/Legacy

All the X540 PCI functions are native PCIe functions.

3.1.2.2 Locked Transactions

The X540 does not support locked requests as a target or a master.

3.1.3 Host Interface

PCIe device numbers identify logical devices within the physical device (the X540 is a physical device). The X540 implements a single logical device with two separate PCI functions: LAN 0 and LAN 1. The device number is captured from each type 0 configuration write transaction.

Each of the PCIe functions interfaces with the PCIe unit through one or more clients. A client ID identifies the client and is included in the *Tag* field of the PCIe packet header. Completions always carry the tag value included in the request to enable routing of the completion to the appropriate client.

3.1.3.1 TAG ID Allocation

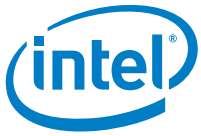
Tag IDs are allocated differently for read and write as detailed in the following sections.

3.1.3.1.1 TAG ID Allocation for Read Transactions

Table 3-1 lists the Tag ID allocation for read accesses. The Tag ID is used by hardware in order to be able to forward the read data to the required internal client.

Table 3-1 TAG ID Allocation Table for Read Transactions

TAG ID	Description	TAG ID	Description
0x0	Data Request 0x0	0x10	Tx Descriptor 0
0x1	Data Request 0x1	0x11	Tx Descriptor 1
0x2	Data Request 0x2	0x12	Tx Descriptor 2
0x3	Data Request 0x3	0x13	Tx Descriptor 3
0x4	Data Request 0x4	0x14	Tx Descriptor 4
0x5	Data Request 0x5	0x15	Tx Descriptor 5
0x6	Data Request 0x6	0x16	Tx Descriptor 6
0x7	Data Request 0x7	0x17	Tx Descriptor 7
0x8	Data Request 0x8	0x18	Rx Descriptor 0



TAG ID	Description	TAG ID	Description
0x9	Data Request 0x9	0x19	Rx Descriptor 1
0xA	Data Request 0xA	0x1A	Rx Descriptor 2
0xB	Data Request 0xB	0x1B	Rx Descriptor 3
0xC	Data Request 0xC	0x1C	Rx Descriptor 4
0xD	Data Request 0xD	0x1D	Rx Descriptor 5
0xE	Data Request 0xE	0x1E	Rx Descriptor 6
0xF	Data Request 0xF	0x1F	Rx Descriptor 7

3.1.3.1.2 TAG ID Allocation for Write Transactions

Request tag allocation depends on these system parameters:

- DCA supported or not supported in the system (DCA_CTRL.DCA_DIS)
- DCA enabled or disabled (DCA_TXCTRL.TX Descriptor DCA EN, DCA_RXCTRL.RX Descriptor DCA EN, DCA_RXCTRL.RX Header DCA EN, DCA_RXCTRL.Rx Payload DCA EN)
- System type: Legacy DCA versus DCA 1.0 (DCA_CTRL.DCA_MODE)
- CPU ID (DCA_RXCTRL.CPUID or DCA_TXCTRL.CPUID)

Case 1 — DCA Disabled in the System:

The following table lists the write requests tags:

Tag ID	Description
2	Write-back descriptor Tx /write-back head.
4	Write-back descriptor Rx.
6	Write data.

Case 2 — DCA Enabled in the System, but Disabled for the Request:

- Legacy DCA platforms — If DCA is disabled for the request, the tags allocation is identical to the case where DCA is disabled in the system (refer to the previous table).
- DCA 1.0 platforms — All write requests have the tag of 0x00.

Case 3 — DCA Enabled in the System, DCA Enabled for the Request:

- Legacy DCA Platforms: the request tag is constructed as follows:
  - Bit[0] — DCA Enable = 1b
  - Bits[3:1] — The *CPU ID* field taken from the CPUID[2:0] bits of the DCA_RXCTRL or DCA_TXCTRL registers
  - Bits[7:4] — Reserved
- DCA 1.0 Platforms: the request tag (all eight bits) is taken from the *CPU ID* field of the DCA_RXCTRL or DCA_TXCTRL registers

3.1.3.2 Completion Timeout Mechanism

In any split transaction protocol, there is a risk associated with the failure of a requester to receive an expected completion. To enable requesters to attempt recovery from this situation in a standard manner, the completion timeout mechanism is defined.

The completion timeout mechanism is activated for each request that requires one or more completions when the request is transmitted. The X540 provides a programmable range for the completion timeout, as well as the ability to disable the completion timeout altogether. The completion timeout is programmed through an extension of the PCIe capability structure.

The X540's reaction to a completion timeout is listed in Table 3-9.

The X540 controls the following aspects of completion timeout:

- Disabling or enabling completion timeout
- Disabling or enabling resending a request on completion timeout
- A programmable range of timeout values
- Programming the behavior of completion timeout is listed in Table 3-2. Note that system software can configure a completion timeout independently per each LAN function.

Table 3-2 Completion Timeout Programming

Capability	Programming Capability
Completion Timeout Enabling	Controlled through PCI configuration. Visible through a read-only CSR bit.
Resend Request Enable	Loaded from the NVM into a R/W CSR bit.
Completion Timeout Period	Controlled through PCI configuration.

Completion Timeout Enable — Programmed through the PCI configuration space. The default is: Completion Timeout Enabled.

Resend Request Enable — The *Completion Timeout Resend* NVM bit (loaded to the *Completion_Timeout_Resend* bit in the PCIe Control Register (GCR) enables resending the request (applies only when completion timeout is enabled). The default is to resend a request that timed out.

3.1.3.2.1 Completion Timeout Period

Programmed through the PCI configuration. Visible through the *Completion_Timeout_Value* bits in the GCR. The X540 supports all four ranges defined by PCIe Gen 1 v2.0 (2.5GT/s) or PCIe Gen2 v1.0 (5GT/s):

- 50 µs to 10 ms
- 10 ms to 250 ms
- 250 ms to 4 s
- 4 s to 64 s

System software programs a range (one of nine possible ranges that sub-divide the four previous ranges) into the PCI configuration register. The supported sub-ranges are:

- 50 µs to 50 ms (default).
- 50 µs to 100 µs
- 1 ms to 10 ms
- 16 ms to 55 ms
- 65 ms to 210 ms
- 260 ms to 900 ms
- 1 s to 3.5 s
- 4 s to 13 s
- 17 s to 64 s

A memory read request for which there are multiple completions are considered completed only when all completions have been received by the requester. If some, but not all, requested data is returned before the completion timeout timer expires, the requestor is permitted to keep or to discard the data that was returned prior to timer expiration.

3.1.4 Transaction Layer

The upper layer of the PCIe architecture is the transaction layer. The transaction layer connects to the X540's core using an implementation-specific protocol. Through this core-to-transaction-layer protocol, the application-specific parts of the X540 interact with the PCIe subsystem and transmits and receives requests to or from the remote PCIe agent, respectively.

3.1.4.1 Transaction Types Accepted by the X540

Table 3-3 Transaction Types Accepted by the Transaction Layer

Transaction Type	FC Type	Tx Layer Reaction	Hardware Should Keep Data From Original Packet	For Client
Configuration Read Request	NPH	CPLH + CPLD	Requester ID, TAG, attribute	Configuration space
Configuration Write Request	NPH + NPD	CPLH	Requester ID, TAG, attribute	Configuration space
Memory Read Request	NPH	CPLH + CPLD	Requester ID, TAG, attribute	CSR space
Memory Write Request	PH + PD	-	-	CSR space
IO Read Request	NPH	CPLH + CPLD	Requester ID, TAG, attribute	CSR space
IO Write Request	NPH + NPD	CPLH	Requester ID, TAG, attribute	CSR space
Read Completions	CPLH + CPLD	-	-	DMA
Message	PH	-	-	Message unit (PM)

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

3.1.4.2 Transaction Types Initiated by the X540

Table 3-4 Transaction Types Initiated by the Transaction Layer

Transaction Type	Payload Size	FC Type	From Client
Configuration Read Request Completion	Dword	CPLH + CPLD	Configuration space
Configuration Write Request Completion	-	CPLH	Configuration space
IO Read Request Completion	Dword	CPLH + CPLD	CSR
IO Write Request Completion	-	CPLH	CSR
Read Request Completion	Dword/Qword	CPLH + CPLD	CSR
Memory Read Request	-	NPH	DMA
Memory Write Request	<= MAX_PAYLOAD_SIZE	PH + PD	DMA
Message	-	PH	Message unit/INT/PM/ error unit

Note: MAX_PAYLOAD_SIZE is loaded from the NVM (up to 512 bytes). Effective MAX_PAYLOAD_SIZE is defined for each PCI function according to the configuration space register for that function.

3.1.4.2.1 Data Alignment

Note: Requests must never specify an address/length combination that causes a memory space access to cross a 4 KB boundary.

The X540 breaks requests into 4 KB-aligned requests (if needed). This does not pose any requirement on software. However, if software allocates a buffer across a 4 KB boundary, hardware issues multiple requests for the buffer. Software should consider aligning buffers to a 4 KB boundary in cases where it improves performance.

The general rules for packet alignment are as follows. Note that these apply to all the X540 requests (read/write, snoop and no snoop):

- 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 X540 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, then it is not broken at the cache line boundary but rather sent as a

single packet (the intent is that the chipset can break the request along cache line boundaries, but the X540 should still benefit from better PCIe use). However, if any of the three general rules require that the request is broken into two or more packets, then the request is broken at the cache line boundary.

3.1.4.2.2 Multiple Tx Data Read Requests (MULR)

The X540 supports 16 multiple pipelined requests for transmit data. In general, requests can belong to the same packet or to consecutive packets. However, the following restrictions apply:

- All requests for a packet must be issued before a request is issued for a consecutive packet.
- Read requests can be issued from any of the supported queues, as long as the previous restriction is met. Pipelined requests can belong to the same queue or to separate queues. However, as previously noted, all requests for a certain packet are issued (from the same queue) before a request is issued for a different packet (potentially from a different queue).
- The PCIe specification does not insure that completions for separate requests return in-order. Read completions for concurrent requests are not required to return in the order issued. The X540 handles completions that arrive in any order. Once all completions arrive for a given request, it can issue the next pending read data request.
- The X540 incorporates a reorder buffer to support re-ordering of completions for all issued requests. Each request/completion can be up to 512 bytes long. The maximum size of a read request is defined as the minimum {2 KB bytes, MAX_READ_REQ}.
- In addition to the transmit data requests, the X540 can issue eight pipelined read requests for Tx descriptors and eight pipelined read requests for Rx descriptors. The requests for Tx data, Tx descriptors, and Rx descriptors are independently issued.

3.1.4.3 Messages

3.1.4.3.1 Received Messages

- Message packets are special packets that carry a message code. The upstream device transmits special messages to the X540 by using this mechanism. The transaction layer decodes the message code and responds to the message accordingly.

Table 3-5 Supported Message in the X540 (as a Receiver)

Message Code [7:0]	Routing r2r1r0	Message	X540 Later Response
0x14	100b	PM_Active_State_NAK	Internal signal set.
0x19	011b	PME_Turn_Off	Internal signal set.
0x50	100b	Slot power limit support (has one Dword data)	Silently drop.
0x7E	010b, 011b, 100b	Vendor_defined type 0 No data	Unsupported request.

Message Code [7:0]	Routing r2r1r0	Message	X540 Later Response
0x7E	010b, 011b, 100b	Vendor_defined type 0 data	Unsupported request.
0x7F	010b, 011b, 100b	Vendor_defined type 1 no data	Silently drop.
0x7F	010b, 011b, 100b	Vendor_defined type 1 data	Silently drop.
0x00	011b	Unlock	Silently drop.

3.1.4.3.2 Transmitted Messages

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

Table 3-6 Supported Message in X540 (as a Transmitter)

Message code [7:0]	Routing r2r1r0	Message
0x20	100b	Assert INT A
0x21	100b	Assert INT B
0x22	100b	Assert INT C
0x23	100b	Assert INT D
0x24	100b	De- Assert INT A
0x25	100b	De- Assert INT B
0x26	100b	De- Assert INT C
0x27	100b	De- Assert INT D
0x30	000b	ERR_COR
0x31	000b	ERR_NONFATAL
0x33	000b	ERR_FATAL
0x18	000b	PM_PME
0x1B	101b	PME_TO_Ack

3.1.4.4 Ordering Rules

The X540 meets the PCIe ordering rules by following the PCI simple device model:

1. Deadlock Avoidance – The X540 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 in order 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 X540 are allowed to proceed.

2. Descriptor/Data Ordering — The X540 insures that a Rx descriptor is written back on PCIe only after the data that the descriptor relates to is written to the PCIe link.
3. 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 doesn't know when the tables are actually updated in the X540, 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.
4. The X540 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.

3.1.4.4.1 Out of Order Completion Handling

In a split transaction protocol, when using multiple read requests in a multi-processor environment, there is a risk that completions for separate requests arrive from the host memory out of order and interleaved. In this case, the X540 sorts the completions and transfers them to the network in the correct order.

Note: Completions for separate read requests are not guaranteed to return in order. Completions for the same read request are guaranteed to return in address order.

3.1.4.5 Transaction Definition and Attributes

3.1.4.5.1 Max Payload Size

The X540's policy for determining Max Payload Size (MPS) is as follows:

1. Master requests initiated by the X540 (including completions) limit MPS to the value defined for the function issuing the request.
2. Target write accesses to the X540 are accepted only with a size of one Dword or two Dwords. Write accesses in the range from three Dwords to MPS are flagged as UR (Unsupported Request) Write accesses above MPS are flagged as malformed.

3.1.4.5.2 Traffic Class (TC) and Virtual Channels (VCs)

The X540 only supports TC = 0 and VC = 0 (default).

3.1.4.5.3 Relaxed Ordering

The X540 takes advantage of the relaxed ordering rules in PCIe. By setting the relaxed ordering bit in the packet header, the X540 enables the system to optimize performance in the following cases:

1. Relaxed ordering for descriptor and data reads — When the X540 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 X540 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 X540 cannot relax ordering for descriptor writes or an MSI write.

Relaxed ordering can be used in conjunction with the no-snoop attribute to enable the memory controller to advance no-snoop writes ahead of earlier snooped writes.

Relaxed ordering is enabled in the X540 by clearing the CTRL_EXT.RO_DIS bit. The actual setting of relaxed ordering is done for LAN traffic by the host through the DCA registers.

3.1.4.5.4 No Snoop

Note: The X540 enables the No Snoop feature by default after power on. The No Snoop feature must be disabled during Rx flow software initialization if there is no intention to use it. To disable No Snoop, the CTRL_EXT.NS_DIS bit should be set to 1b.

The X540 sets the *Snoop Not Required* attribute for master data writes. System logic can provide a separate path into system memory for non-coherent traffic. The non-coherent path to system memory provides a higher, more uniform, bandwidth for write requests.

Note: The *Snoop Not Required* attribute does not alter transaction ordering. Therefore, to achieve the maximum benefit from *Snoop Not Required* transactions, it is advisable to set the relaxed ordering attribute as well (assuming that system logic supports both attributes). In fact, some chipsets require that relaxed ordering is set for no-snoop to take effect.

No snoop is enabled in the X540 by clearing the CTRL_EXT.NS_DIS bit. The actual setting of no snoop is done for LAN traffic by the host through the DCA registers.

3.1.4.5.5 No Snoop and Relaxed Ordering for LAN Traffic

Software can configure no-snoop and relax order attributes for each queue and each type of transaction by setting the respective bits in the DCA_RXCTRL and TCA_TXCTRL registers.

Table 3-7 lists the default behavior for the *No-Snoop* and *Relaxed Ordering* bits for LAN traffic when I/OAT 2 is enabled.

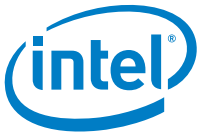


Table 3-7 LAN Traffic Attributes

Transaction	No Snoop Default	Relaxed Ordering Default	Comments
Rx Descriptor Read	N	Y	
Rx Descriptor Write-Back	N	N	Read-only. Must never be used for this traffic.
Rx Data Write	Y	Y	See note and the section that follows.
Tx Descriptor Read	N	Y	
Tx Descriptor Write-Back	N	Y	
Tx Data Read	N	Y	

Note: RX payload no-snoop is also conditioned by the *NSE* bit in the receive descriptor (RDESC.NSE).

No-Snoop Option for Payload

Under certain conditions, which occur when I/OAT 2 is enabled, software knows that it is safe to transfer a new packet into a certain buffer without snooping on the FSB. This scenario occurs when software is posting a receive buffer to hardware that the CPU has not accessed since the last time it was owned by hardware. This might happen if the data was transferred to an application buffer by the data movement engine. In this case, software should be able to set a bit in the receive descriptor indicating that the X540 should perform a no-snoop transfer when it eventually writes a packet to this buffer. When a no-snoop transaction is activated, the TLP header has a no-snoop attribute in the *Transaction Descriptor* field. This is triggered by the *NSE* bit in the receive descriptor.

3.1.4.6 Flow Control

3.1.4.6.1 Flow Control Rules

The X540 only implements the default Virtual Channel (VC0). A single set of credits is maintained for VC0.

Table 3-8 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:

- The X540 maintains two credits for NPD at any given time. It increments the credit by one after the credit is consumed, and sends an UpdateFC packet as soon as possible. UpdateFC packets are scheduled immediately after a resource is available.
- The X540 provides 16 credits for PH (such as for concurrent target writes) and four credits for NPH (such as for four concurrent target reads). UpdateFC packets are scheduled immediately after a resource is available.
- The X540 follows the PCIe recommendations for frequency of UpdateFC FCPs.

3.1.4.6.2 Upstream Flow Control Tracking

The X540 issues a master transaction only when the required flow control credits are available. Credits are tracked for posted, non-posted, and completions (the later to operate against a switch).

3.1.4.6.3 Flow Control Update Frequency

In all cases, UpdateFC packets are scheduled immediately after a resource is available.

When the link is in the L0 or L0s link state, Update FCPs for each enabled type of non-infinite flow control credit must be scheduled for transmission at least once every 30 µs (-0% /+50%), except when the *Extended Sync* bit of the Control Link register is set, in which case the limit is 120 µs (-0% /+50%).

3.1.4.6.4 Flow Control Timeout Mechanism

The X540 implements the optional flow control update timeout mechanism.

The mechanism is active when the link is in L0 or L0s link state. It uses a timer with a limit of 200 µs (-0% /+50%), where the timer is reset by the receipt of any Init or Update FCP. Alternately, the timer can be reset by the receipt of any DLLP.

Upon timer expiration, the mechanism instructs the PHY to retrain the link (via the LTSSM recovery state).

3.1.5 Link Layer

3.1.5.1 ACK / NAK Scheme

The X540 supports two alternative schemes for ACK/NAK rate:

- ACK/NAK is scheduled for transmission following any TLP.
- ACK/NAK is scheduled for transmission according to timeouts specified in the PCIe specification.

3.1.5.2 Supported DLLPs

The following DLLPs are supported by the X540 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 X540 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.

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

If retrain is necessary, there is a need to guarantee that no abrupt termination of the Tx packet happens. For this reason, early termination of the transmitted packet is possible. This is done by appending the EDB to the packet.

3.1.6 Physical Layer

3.1.6.1 Link Speed

The X540 supports PCIe Gen 1 v2.0 (2.5GT/s) or PCIe Gen2 v1.0 (5GT/s). The following configuration controls link speed:

- PCIe *Supported Link Speeds* bit — Indicates the link speeds supported by the X540. Loaded from the PCIe Analog Configuration Module in the NVM, and could be set as follows.

NVM Word Offset (Starting at Odd Word)	Allow Gen 1 and Gen 2 (Default)	Force Gen 1 Setting	Description
2*N+1	0x094		MORIA6 register OFFSET (lower word).
2*N+2	0x0000	0x0100	Disabling gen2 is controlled by setting bit[8] in this register. When the bit is set, the X540 does not advertise gen 2 link-speed support.

- 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 X540 does not initiate a hardware autonomous speed change.

The X540 supports entering compliance mode at the speed indicated in the *Target Link Speed* field in the PCIe Link Control 2 register. Compliance mode functionality is controlled via the PCIe Link Control 2 register.

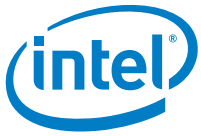
3.1.6.2 Link Width

- The X540 supports a maximum link width of x8, x4, x2, or x1 as determined by the "PCIe Analog Configuration" Module in the NVM and could be set as follow. Note that these setting are not likely being needed in nominal operation:

NVM Word Offset (starting at odd word)	Enable x8 setting (Default)	Limit to x4 setting	Limit to x2 setting	Limit to x1 setting	Description
2*N+1	0x094				MORIA6 register OFFSET (lower word)
2*N+2	0x0000	0x00F0	0x00FC	0x00FE	Lanes can be disabled, by setting bits[7:0] in this offset. Having bit[X] set will cause laneX to be disabled, resulting in narrower link widths (bit per lane)

The maximum link width is loaded into the *Max Link Width* field of the PCIe Capability register (LCAP[11:6]). Hardware default is the x8 link.

During link configuration, the platform and the X540 negotiate on a common link width. The link width must be one of the supported PCIe link widths (x1, 2x, x4, x8), such that:



- If Maximum Link Width = x8, then the X540 negotiates to either x8, x4, x2 or x1[1]
- If Maximum Link Width = x4, then the X540 negotiates to either x4 or x1
- If Maximum Link Width = x1, then the X540 only negotiates to x1

When negotiating for x4, x2, or x1 link, the X540 may negotiate the link to reside starting from physical lane 0 or starting from physical lane 4.

The X540 does not initiate a hardware autonomous link width change. However, it will move to recovery if it detects a low reliability link, and will finally form a degraded link.

3.1.6.3 Polarity Inversion

If polarity inversion is detected, the receiver must invert the received data.

During the training sequence, the receiver looks at symbols 6-15 of TS1 and TS2 as the indicators of lane polarity inversion (D+ and D- are swapped). If lane polarity inversion occurs, the TS1 symbols 6-15 received are D21.5 as opposed to the expected D10.2. Similarly, if lane polarity inversion occurs, symbols 6-15 of the TS2 ordered set are D26.5 as opposed to the expected 5 D5.2. This provides the clear indication of lane polarity inversion.

3.1.6.4 L0s Exit Latency

The number of Fast Training Sequence (FTS) sequences (N_FTS) sent during L0s exit is loaded from the NVM into an 8-bit read-only register.

3.1.6.5 Lane-to-Lane De-Skew

A multi-lane link can have many sources of lane-to-lane skew. Although symbols are transmitted simultaneously on all lanes, they cannot be expected to arrive at the receiver without lane-to-lane skew. The lane-to-lane skew can include components, which are less than one bit time, bit time units (400/200 ps for 2.5/5 Gb), or full symbol time units (4/2 ns). This type of skew is caused by the retiming repeaters' insert/delete operations. Receivers use TS1 or TS2 or Skip Ordered Sets (SOS) to perform link de-skew functions.

The X540 supports de-skew of up to 12 symbols time — 48 ns for PCIe Gen 1 v2.0 (2.5GT/s) and 24 ns for PCIe Gen2 v1.0 (5GT/s).

3.1.6.6 Lane Reversal

Auto lane reversal is supported by the X540 at its hardware default setting. The following lane reversal modes are supported:

- Lane configurations x8, x4, x2, and x1
- Lane reversal in x8, x4, x2, and in x1
- Degraded mode (downshift) from x8 to x4 to x2 to x1 and from x4 to x2 to x1.

1. See restriction in Section 3.1.6.6.

Figure 3-2 through Figure 3-5 shows the lane downshift examples in both regular and reversal connections as well as lane connectivity from a system level perspective.

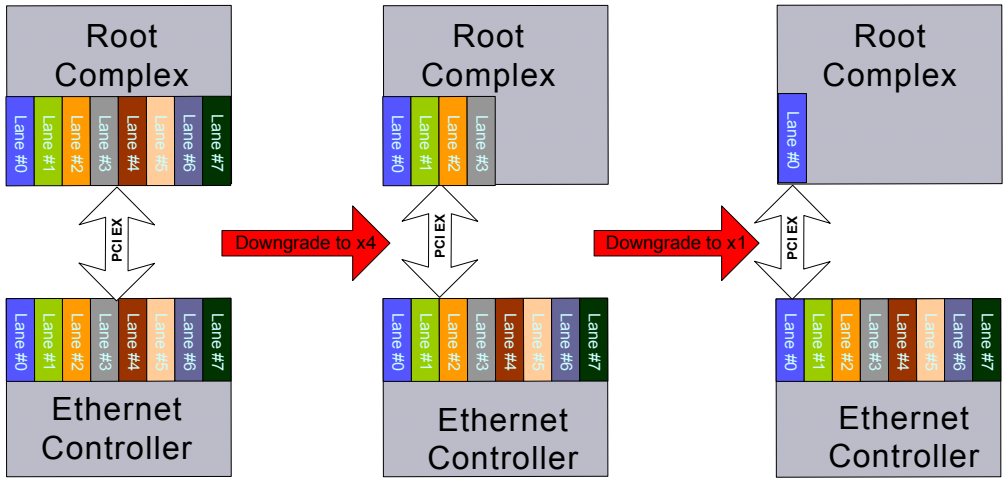


Figure 3-2 Lane Downshift in an x8 Configuration

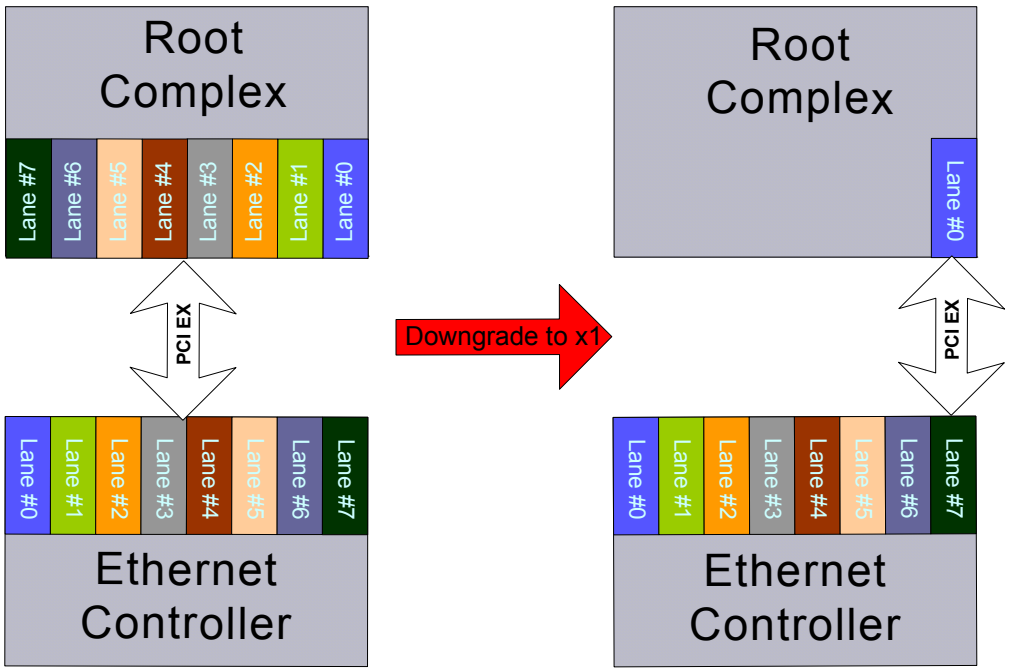


Figure 3-3 Lane Downshift in a Reversal x8 Configuration

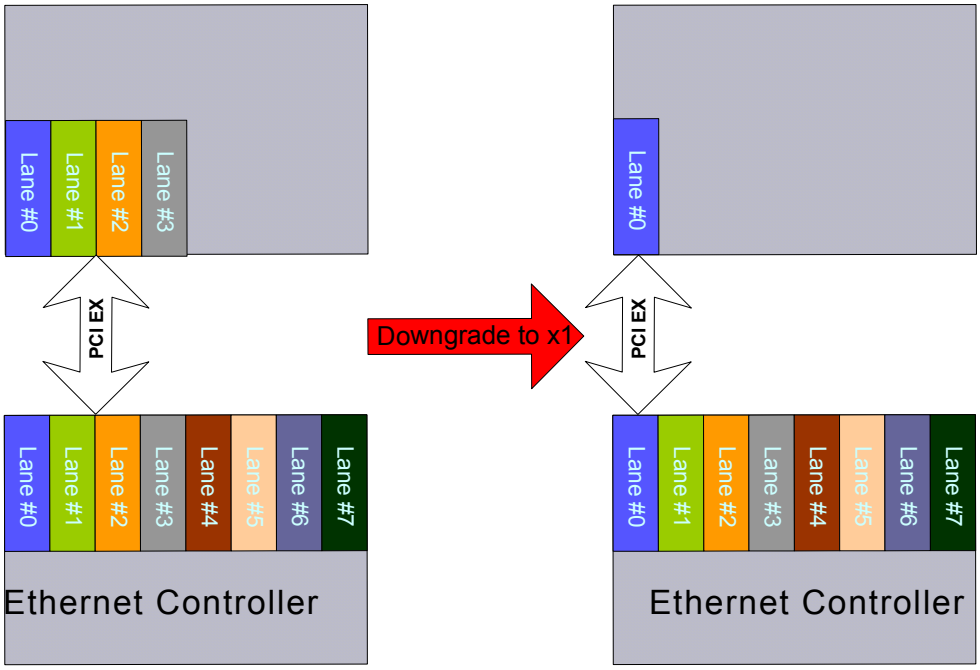


Figure 3-4 Lane Downshift in a x4 Configuration

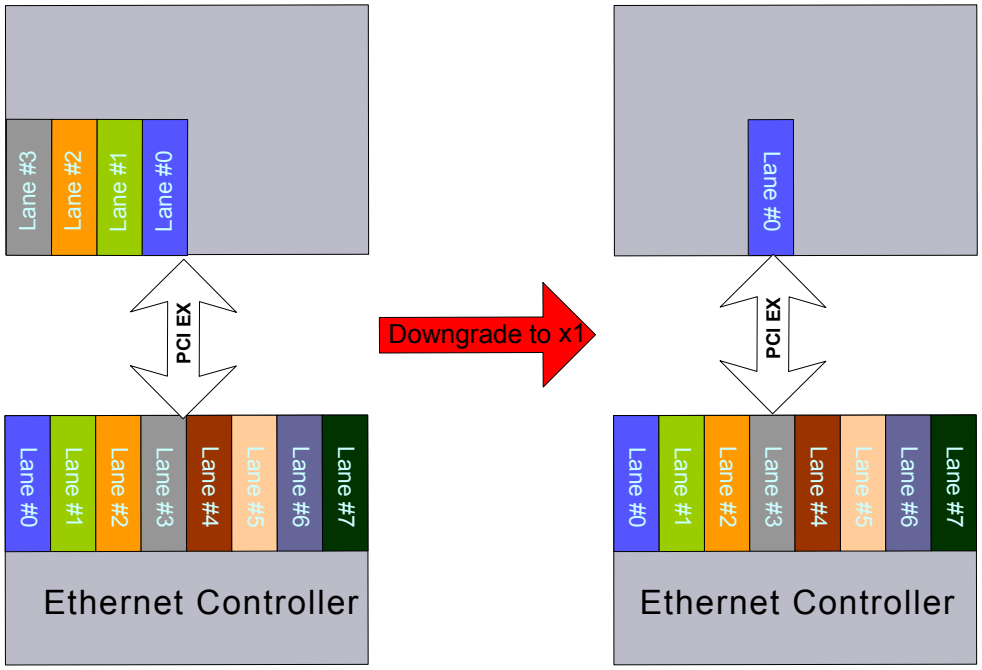


Figure 3-5 Lane Downshift in an x4 Reversal Configuration

3.1.6.7 Reset

The PCIe PHY supplies the core reset to the X540. The reset can be caused by the following events:

- Upstream move to hot reset — Inband Mechanism (LTSSM).
- Recovery failure (LTSSM returns to detect)
- Upstream component moves to disable.

3.1.6.8 Scrambler Disable

- The scrambler/de-scrambler functionality in the X540 can be eliminated by three mechanisms:
- Upstream according to the PCIe specification
- NVM bit — Scram_dis

3.1.7 Error Events and Error Reporting

3.1.7.1 General Description

PCIe defines two error reporting paradigms: the baseline capability and the Advanced Error Reporting (AER) capability. The baseline error reporting capabilities are required of all PCIe devices and define the minimum error reporting requirements. The AER capability is defined for more robust error reporting and is implemented with a specific PCIe capability structure. Both mechanisms are supported by the X540.

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 associated to any specific function within the device are logged in the corresponding status and logging registers of all functions in that device. These include the following cases of Unsupported Request (UR):

- A memory or I/O access that does not match any Base Address Register (BAR) for any function
- Messages
- Configuration accesses to a non-existent function

Figure 3-6 shows, in detail, the flow of error reporting in the X540.

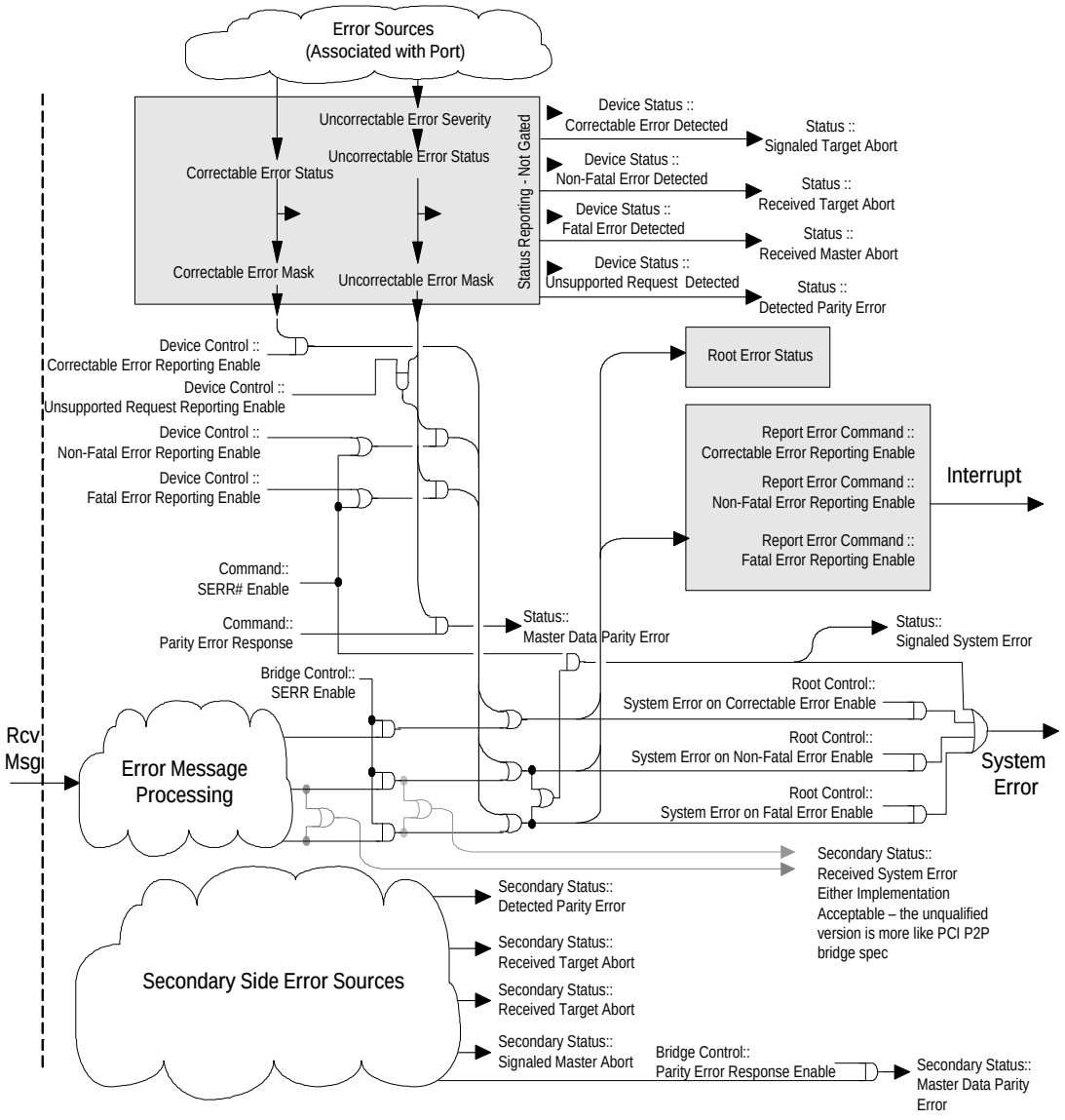


Figure 3-6 Error Reporting Mechanism

3.1.7.2 Error Events

Table 3-9 lists the error events identified by the X540 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 3-9 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
Unsupported Request (UR)	Wrong Config Access MRdLk Config Request Type1 Unsupported Vendor Defined Type 0 Message Not Valid MSG Code Not Supported TLP Type Wrong Function Number Received TLP Outside Address Range	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 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
Completer Abort	Received Target Access With Data Size >64 bits	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
Error Name	Error Events	Default Severity	Action
Flow Control Protocol Error	Minimum Initial Flow Control Advertisements Flow Control Update for Infinite Credit Advertisement	Uncorrectable. ERR_FATAL	Receiver Behavior is Undefined
Malformed TLP (MP)	Data Payload Exceed Max_Payload_Size Received TLP Data Size Does Not Match Length Field TD field value does not correspond with the observed size PM Messages That Don't Use TC0. Usage of Unsupported VC	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

3.1.7.3 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 X540 then reacts as listed in Table 3-9.

The X540 does not initiate any additional master requests for that PCI function until it detects an internal software reset for the associated LAN port. Software is able to access device registers after such a fault.

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.

3.1.7.4 End-to-End CRC (ECRC)

The X540 supports ECRC as defined in the PCIe specification. The following functionality is provided:

- Inserting ECRC in all transmitted TLPs:
  - The X540 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.
- 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), which results in re-issuing the request.
  - The X540 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 errors are reported
- System software can configure ECRC independently per each LAN function

3.1.7.5 Partial Read and Write Requests

Partial memory accesses

The X540 has limited support of read and write requests with only part of the byte enable bits set:

- Partial writes with at least one byte enabled are silently dropped.
- 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).
- Partial reads with at least one byte enabled are handled as a full read. Any side effect of the full read (such as clear by read) is also applicable to partial reads.
- Zero-length reads generate a completion, but the register is not accessed and undefined data is returned.

Note: The X540 does not generate an error indication in response to any of the previous events.

Partial I/O accesses

- Partial access on address
  - A write access is discarded
  - A read access returns 0xFFFF
- Partial access on data, where the address access was correct
  - A write access is discarded
  - A read access performs the read

3.1.7.6 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.

3.1.7.7 Completion With Unsuccessful Completion Status

A completion with unsuccessful completion status is dropped and not delivered to its destination. The request that corresponds to the unsuccessful completion is retried by sending a new request for undeliverable data.

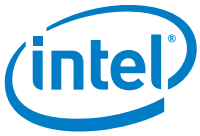
3.1.7.8 Error Reporting Changes

The PCIe Rev. 1.0 specification defines two changes to advanced error reporting. A (new) *Role Based Error Reporting* bit in the Device Capabilities register is set to 1b to indicate that these changes are supported by the X540.

1. Setting the *SERR# Enable* bit in the PCI Command register also enables UR reporting (in the same manner that the *SERR# Enable* bit enables reporting of correctable and uncorrectable errors). In other words, the *SERR# Enable* bit overrides the *Unsupported Request Error Reporting Enable* bit in the PCIe Device Control register.
2. Changes in the response to some uncorrectable non-fatal errors detected in non-posted requests to the X540. These are called Advisory Non-Fatal Error cases. For each of the errors listed, the following behavior is defined:
   - The *Advisory Non-Fatal Error Status* bit is set in the Correctable Error Status register to indicate the occurrence of the advisory error and the *Advisory Non-Fatal Error Mask* corresponding bit in the Correctable Error Mask register is checked to determine whether to proceed further with logging and signaling.
   - If the *Advisory Non-Fatal Error Mask* bit is clear, logging proceeds by setting the corresponding bit in the Uncorrectable Error Status register, based upon the specific uncorrectable error that's being reported as an advisory error. If the corresponding *Uncorrectable Error* bit in the Uncorrectable Error Mask register is clear, the First Error Pointer and Header Log registers are updated to log the error, assuming they are not still occupied by a previous unserviced error.
   - An ERR_COR Message is sent if the *Correctable Error Reporting Enable* bit is set in the Device Control register. An ERROR_NONFATAL message is not sent for this error.

The following uncorrectable non-fatal errors are considered as advisory non-fatal errors:

- A completion with an Unsupported Request or Completer Abort (UR/CA) status that signals an uncorrectable error for a non-posted request. If the severity of the UR/CA error is non-fatal, the completer must handle this case as an advisory non-fatal error.



- When the requester of a non-posted request times out while waiting for the associated completion, the requester is permitted to attempt to recover from the error by issuing a separate subsequent request or to signal the error without attempting recovery. The requester is permitted to attempt recovery zero, one, or multiple (finite) times, but must signal the error (if enabled) with an uncorrectable error message if no further recovery attempt is made. If the severity of the completion timeout is non-fatal, and the requester elects to attempt recovery by issuing a new request, the requester must first handle the current error case as an advisory non-fatal error.
- Receiving a poisoned TLP. See Section 3.1.7.3.
- When a receiver receives an unexpected completion and the severity of the unexpected completion error is non-fatal, the receiver must handle this case as an advisory non-fatal error.

3.1.8 Performance Monitoring

The X540 incorporates PCIe performance monitoring counters to provide common capabilities to evaluate performance. The X540 implements four 32-bit counters to correlate between concurrent measurements of events as well as the sample delay and interval timers. The four 32-bit counters can also operate in a two 64-bit mode to count long intervals or payloads. Software can reset, stop, or start the counters (all at the same time).

Some counters operate with a threshold — the counter increments only when the monitored event crossed a configurable threshold (such as the number of available credits is below a threshold).

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 3.1.8.1.

The list of events supported by the X540 and the counters *Control* bits are described in the PCIe Registers section.

3.1.8.1 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 1 µs as defined in the *LBC Timer* field in the PCIe Statistic Control registers.

- When an event occurs and the value of the LBC meets or exceeds the threshold defined in the *LBC Threshold* field in the PCIe Statistic Control registers, the respective statistics counter increments, and the LBC counter is cleared to zero.

3.2 SMBus

SMBus is a management interface for pass-through and/or configuration traffic between an external Management Controller (MC) and the X540.

3.2.1 Channel Behavior

The SMBus specification defines the maximum frequency of the SMBus as 100 KHz. However, the SMBus interface can be activated up to 400 KHz without violating any hold and setup time.

SMBus connection speed bits define the SMBus mode. Also, SMBus frequency support can be defined only from the NVM.

3.2.2 SMBus Addressing

The SMBus is presented as two SMBus devices on the SMBus (two SMBus addresses). All pass-through functionality is duplicated on the SMBus address, where each SMBus address is connected to a different LAN port.

Note: Designers are not allowed to configure both ports to the same address. When a LAN function is disabled, the corresponding SMBus address is not presented to the MC.

The SMBus addresses are set using the *SMBus 0 Slave Address* and *SMBus 1 Slave Address* fields in the NVM.

Note: For the X540 single port configuration, the SMBus *Single Port Mode* bit should be set in the NVM, and only the *SMBus 0 Slave Address* field is valid.

The SMBus addresses (those that are enabled from the NVM) can be re-assigned using the SMBus ARP protocol.

Besides the SMBus address values, all the previously listed parameters of the SMBus (SMBus channel selection, single port mode, and address enable) can be set only through the NVM.

All SMBus addresses should be in Network Byte Order (NBO) with the most significant byte first.

3.2.3 SMBus Notification Methods

The X540 supports three methods of signaling the external MC that it has information that needs to be read by the external MC:

- SMBus alert — Refer to Section 3.2.3.1.
- Asynchronous notify — Refer to Section 3.2.3.2.
- Direct receive — Refer to section Section 3.2.3.3.

The notification method that is used by the X540 can be configured from the SMBus using the Receive Enable command. The default method is set from the *Notification Method* field in NVM word LRXEN1.

The following events cause the X540 to send a notification event to the external MC:

- Receiving a LAN packet, designated for the MC.
- Receiving a Request Status command from the MC that initiates a status response.
- The X540 is configured to notify the external MC upon status changes (by setting the EN_STA bit in the Receive Enable Command) and one of the following events happen:
- TCO Command Aborted
- Link Status changed
- Power state change
- MACsec indication.

There can be cases where the external MC is hung and cannot not respond to the SMBus notification. The X540 has a timeout value defined in the NVM (refer to Section 6.5.4.3) to avoid hanging while waiting for the notification response. If the MC does not respond until the timeout expires, the notification is de-asserted.

3.2.3.1 SMBus Alert and Alert Response Method

SMBALRT_N (SMBus Alert) is an additional SMBus signal that acts as an asynchronous interrupt signal to an external SMBus master. The X540 asserts this signal each time it has a message that it needs the external MC to read and if the chosen notification method is the SMBus alert method.

Note: SMBALRT_N is an open-drain signal, which means that devices other than the X540 can be connected to the same alert pin. The external MC requires a mechanism to distinguish between the alert sources as follows:

The external MC responds to the alert by issuing an Alert Response Address (ARA) cycle to detect the alert source device. The X540 responds to the ARA cycle (if it was the SMBus alert source) and de-asserts the alert when the ARA cycle completes. Following the ARA cycle, the MC issues a Read command to retrieve the the X540 message.

Note: Some MCs do not implement the ARA cycle transaction. These MCs respond to an alert by issuing a Read command to the X540 (0xC0/0xD0 or 0xDE). The X540 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 X540 is the source of the SMBus alert, it replies to the read transaction.

The ARA cycle is an SMBus receive byte transaction to SMBus Address 0x18.

Note: The ARA transaction does not support PEC.

The alert response address transaction format is as follows:

1	7	1	1	8	1	1
S	ARA	Rd	A	Slave Device Address	A	P
	0001 100	0	0		1	

Figure 3-7 SMBus ARA Cycle Format

3.2.3.2 Asynchronous Notify Method

When configured using the asynchronous notify method, the X540 acts as an SMBus master and notifies the external MC by issuing a modified form of the write word transaction. The asynchronous notify transaction SMBus address and data payload are configured using the Receive Enable command or by using the NVM defaults (see Section 6.5.3.20).

Note: The asynchronous notify is not protected by a PEC byte.

1	7	1	1	7	1	1	
S	Target Address	Wr	A	Sending Device Address		A	• • •
	MC Slave Address	0	0	Manageability Slave SMBus Address	0	0	

8	1	8	1	1
Data Byte Low	A	Data Byte High	A	P
Interface	0	Alert Value	0	

Figure 3-8 Asynchronous Notify Command Format

3.2.3.3 Direct Receive Method

If configured, the X540 has the capability to send the message it needs to transfer to the external MC, as a master over the SMBus instead of alerting the MC and waiting for it to read the message.

The message format is shown in Figure 3-9. Note that the command that should be used is the same command that should be used by the MC in the Block Read command and the opcode that the X540 puts in the data is the same as it would have put in the Block Read command of the same functionality. The rules for the F and L flags 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	• • •
	MC 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	

Figure 3-9 Direct Receive Transaction Format

3.2.4 Receive TCO Flow

The X540 is used as a channel for receiving packets from the network link and passing them to an external MC. The MC can configure the X540 to pass specific packets to the MC (see Section 11.2). Once a full packet is received from the link and identified as a manageability packet that should be transferred to the MC, the X540 starts the receive TCO transaction flow to the MC.

The maximum SMBus fragment length is defined in the NVM (see Section 6.5.4.2). The X540 uses the SMBus notification method to notify the MC that it has data to deliver. The packet is divided into fragments, where the X540 uses the maximum fragment size allowed in each fragment. 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 in the receive TCO LAN packet transaction.

When SMBus Alert is selected as the MC notification method, the X540 notifies the MC on each fragment of a multi-fragment packet.

When asynchronous notify is selected as the MC notification method, the X540 notifies the MC only on the first fragment of a received packet. It is the MC's responsibility to read the full packet including all the fragments.

Any timeout on the SMBus notification results in discarding of the entire packet. Any NACK by the MC on one of the X540's receive bytes also causes the packet to be silently discarded.

Since SMBus throughput is lower than the network link throughput, the X540 uses an 8 KB internal buffer per LAN port, which stores incoming packets prior to being sent over the SMBus interface. The X540 services back-to-back management packets as long as the buffer does not overflow.

The maximum size of the received packet is limited by the X540 hardware to 1536 bytes. Packets larger then 1536 bytes are silently discarded. Any packet smaller than 1536 bytes is processed by the X540.

Note: When the *RCV_EN* bit is cleared, all receive TCO functionality is disabled including packets directed to the MC as well as auto ARP processing.

3.2.5 Transmit TCO Flow

The X540 is used as a channel for transmitting packets from the external MC to the network link. The network packet is transferred from the external MC over the SMBus, and then, when fully received by the X540, is transmitted over the network link.

In dual-address mode, each SMBus address is connected to a different LAN port. When a packet received in SMBus transactions using SMBus 0 Slave Address, it is transmitted to the network using LAN port 0 and is transmitted through LAN port 1 if received on SMB address 1. In single-address mode, the transmitted port is chosen according to the fail-over algorithm (see Section 11.2.2.2).

The X540 supports packets up to an Ethernet packet length of 1536 bytes. SMBus transactions can be up to 240 bytes in length, which means that packets can be transferred over the SMBus in more than one fragment. In each command byte there are the *F* and *L* bits. When the *F* bit is set, it means that this is the first fragment of the packet and *L* means that it is the last fragment of the packet (when both are set, it means that the entire packet is in one fragment). The packet is sent over the network link only after all its fragments have been received correctly over the SMBus.

The X540 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) must be calculated and inserted by the external MC (the X540 does not change any field in the transmitted packet other than adding padding and CRC bytes). If the packet sent by the MC is bigger than 1536 bytes, then the packet is silently discard by the X540.

The minimum packet length defined by the 802.3 specification is 64 bytes. The X540 pads packets that are less than 64 bytes to meet the specification requirements (no need for the external MC to do it). There is one exception, that is if the packet sent over the SMBus is less than 32 bytes, the MC must pad it for at least 32 bytes. The passing bytes value should be zero. Packets that are smaller then 32 bytes (including padding) are silently discarded by the X540.

If the network link is down when the X540 has received the last fragment of the packet, it silently discards the packet.

Note: Any link down event while the packet is being transferred over the SMBus does not stop the operation, since the X540 waits for the last fragment to end to see whether the network link is up again.

The transmit SMBus transaction is described in Section 11.7.2.1.

3.2.5.1 Transmit Errors in Sequence Handling

Once a packet is transferred over the SMBus from the MC to the X540 the *F* and *L* flags should follow specific rules. The *F* flag defines that this is the first fragment of the packet, and the *L* flag defines that the transaction contains the last fragment of the packet.

Table 3-10 lists the different option of the flags in transmit packet transactions.

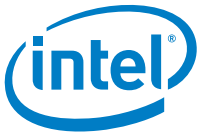


Table 3-10 SMBus Transmit Sequencing

Previous	Current	Action / Notes
Last	First	Accept both.
Last	Not First	Error for current transaction. Current transaction is discarded and an abort status is asserted.
Not Last	First	Error for previous transaction. The previous transaction (until the previous first) is discarded. The current packet is processed. No abort status is asserted.
Not Last	Not First	The X540 can process the current transaction.

Note: Since every other Block Write command in the TCO protocol has both the First (F) and Last (L) flags on, they cause flushing any pending transmit fragments that were previously received. As such, when running the TCO transmit flow, no other Block Write transactions are allowed in between the fragments.

3.2.5.2 TCO Command Aborted Flow

Bit 6 in first byte of the status returned from the X540 to the external MC indicates that there was a problem with previous SMBus transactions or with the completion of the operation requested in previous transaction.

The abort can be asserted due to any of the following reasons:

- Any error in the SMBus protocol (NACK, SMBus time-outs).
- Any error in compatibility due to required protocols to specific functionality (Rx Enable command with byte count not 1/14 as defined in the command specification).
- If the X540 does not have space to store the transmit packet from the MC (in an internal buffer) before sending it to the link. In this case, all transactions are completed but the packet is discarded and the BMC is notified through the *Abort* bit.
- Error in *First/Last* bit sequence during multi-fragment transactions.
- The *Abort* bit is asserted after an internal reset to the X540 manageability unit.

Note: The abort in the status does not always imply that the last transaction of the sequence was bad. There is a time delay between the time the status is read from the X540 and the time the transaction has occurred.

3.2.6 Concurrent SMBus Transactions

Concurrent SMBus write transactions are not permitted. Once a transaction is started, it must be completed before additional transaction can be initiated.

3.2.7 SMBus ARP Functionality

The X540 supports the SMBus ARP protocol as defined in the SMBus 2.0 specification. The X540 is a persistent slave address device when its SMBus address is valid after power-up and loaded from the NVM. The X540 also supports all SMBus ARP commands defined in the SMBus specification, both general and directed.

Note: SMBus ARP can be disabled through NVM configuration (See Section 6.5.4.3).

3.2.7.1 SMBus ARP

the X540 responds as two SMBus devices, in which it has two sets of *AR*/*AV* flags — one for each port. The X540 should respond twice to the SMBus-ARP master, one time for each port. Both SMBus addresses are taken from the SMBus-ARP addresses word of the NVM. The UDID is different between the two ports in the version ID field, which represents the Ethernet MAC address, which is different between the two ports. It is recommended for the X540 to first answer as port 0, and only when the address is assigned, to answer as port 1 to the Get UDID command.

3.2.7.2 SMBus-ARP Flow

SMBus-ARP flow is based on the status of two *AV*s and *AR*s:

- Address Valid — This flag is set when the X540 has a valid SMBus address.
- Address Resolved — This flag is set when the X540 SMBus address is resolved: SMBus address was assigned by the SMBus-ARP process.

Note: These flags are internal the X540 flags and not shown to external SMBus devices.

Since the X540 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, it means that the X540 always has a valid SMBus address. The entire SMBus ARP Flow is described in Figure 3-10.

When the SMBus master needs to start the SMBus-ARP process, it resets (in terms of ARP functionality) all the devices on the SMBus, by issuing either Prepare to ARP or Reset Device commands. When the X540 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).

A cleared *AR* flag means that the X540 answers the following SMBus ARP transactions that are issued by the master. The SMBus master then issues a Get UDID command (General or Directed), to identify the devices on the SMBus. The X540 responds to the Directed command all the time, and to the General command only if its *AR* flag is not set. After the Get UDID, the master assigns the X540 SMBus address, by issuing Assign Address command. The X540 checks whether the UDID matches its own UDID, and if there is a match 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 X540 does not respond to the Get UDID General command, while all other commands should be processed even if the *AR* flag is set.

After SMBus ARP is successfully carried out, the new address is stored in the NVM, and will thus be the address used at the next power up.

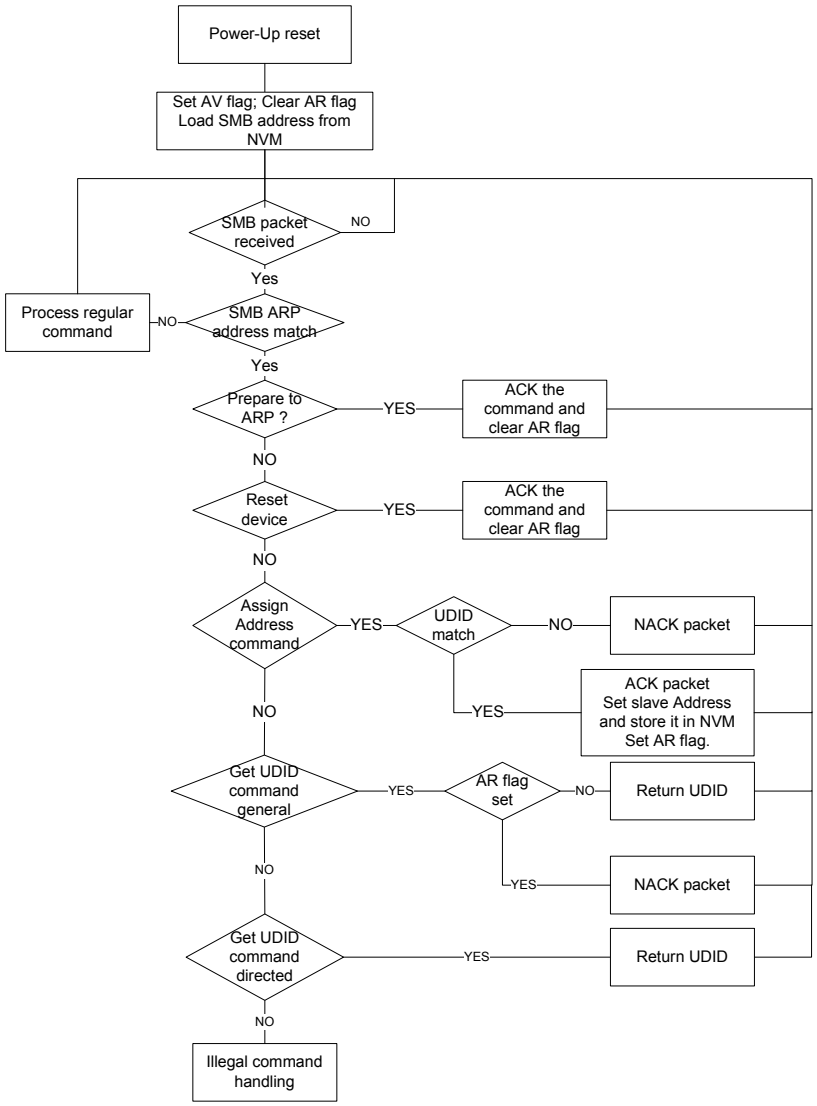


Figure 3-10 SMBus-ARP Flow

3.2.8 Fairness Arbitration

When sending MCTP messages over SMBus and when fairness arbitration is enabled (see Section 6.5.4.3), the X540 should respect the fairness arbitration as defined in section 5.13 of DSP0237 when sending MCTP messages.

3.3 Network Controller — Sideband Interface (NC-SI)

The NC-SI interface in the X540 is a connection to an external MC.

The X540 NC-SI interface meets the NC-SI version 1.0.0 specification as a PHY-side device.

3.3.1 Electrical Characteristics

The X540 complies with the electrical characteristics defined in the NC-SI specification.

3.3.2 NC-SI Transactions

Compatible with the NC-SI specification.

3.4 Non-Volatile Memory (NVM)

3.4.1 General Overview

The X540 uses a Flash device for storing product configuration information. The Flash is divided into three general regions:

- Hardware Accessed — Loaded by the X540 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 4.0.
- Firmware Area — Includes 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.

3.4.2 Flash Device Requirements

The X540 merges the 82599 legacy EEPROM and Flash content in a single Flash device. Flash devices require a sector erase instruction in case a cell is modified from 0b to 1b. As a result, in order to update a single byte (or block of data) it is required to erase it first. The X540 supports Flash devices with a sector erase size of 4 KB. Note that many Flash vendors are using the term sector differently. The X540 Datasheet uses the term Flash sector for a logic section of 4 KB.

The X540 supports Flash devices that are either write-protected by default after power-up or not. The X540 is responsible to remove the protection by sending the write-protection removal OpCode to the Flash after power up.

The following OpCodes are supported by the X540 as they are common to all supported Flash devices:

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

3.4.3 Shadow RAM

The X540 maintains the first two 4 KB sectors, Sector 0 and Sector 1, for the configuration content. At least one of these two sectors must be valid at any given time or else the X540 is set by hardware default. Following a Power On Reset (POR) the X540 copies the valid lower 4 KB sector of the Flash device into an internal shadow RAM. Any further accesses of the software or firmware to this section of the NVM are directed to the internal shadow RAM. Modifications made to the shadow RAM content are then copied by the X540 into the other 4KB sector of the NVM, flipping circularly the valid sector between sector 0 and 1 of NVM.

This mechanism provides the following advantage:

1. A seamless backward compatible read/write interface for software/firmware to the first 4 KB of the NVM as if an external EEPROM device were connected. This interface is referred as EEPROM-Mode access to the Flash.
2. A way for software to protect image-update procedure from power down events by establishing a double-image policy. See Section 6.2.1.1 for a description of the double-image policy. It relies on having pointers to all the other NVM modules mapped in the NVM sector which is mirrored in the internal shadow RAM.

Figure 3-11 shows the shadow RAM mapping and interface.

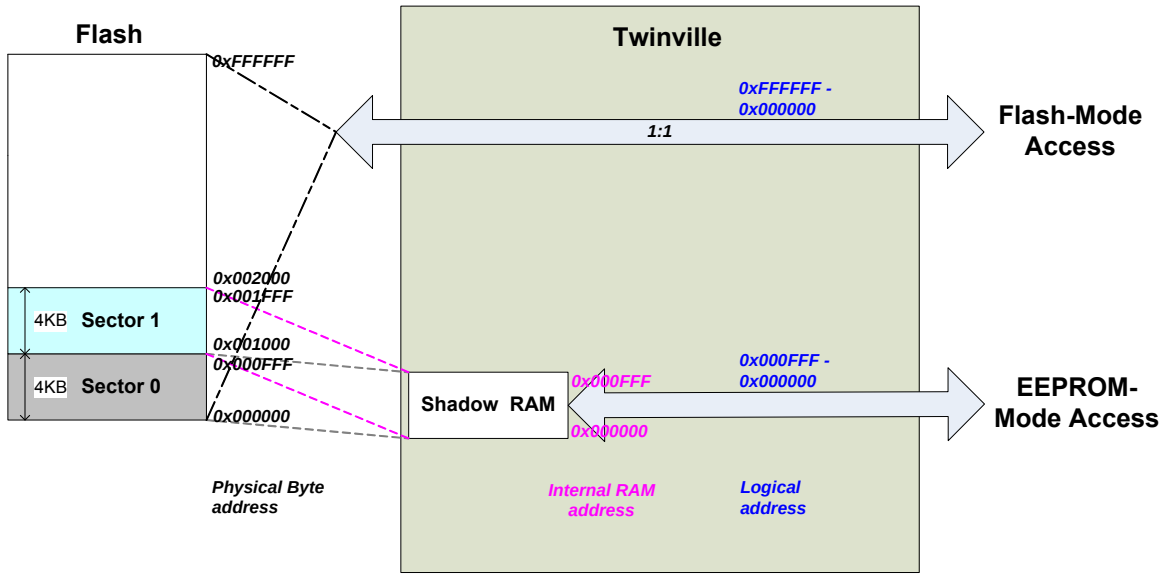


Figure 3-11 NVM Shadow RAM

Following a write access by software or firmware to the shadow RAM, the data should finally be updated in the Flash as well. The X540 updates the Flash from the shadow RAM when software/firmware requests explicitly to update the Flash by setting the *FLUPD* bit in the EEC register. For saving Flash updates, it is expected that software/firmware set the *FLUPD* bit only once it has completed their last write access to the Flash. The X540 then copies the content of the shadow RAM to the non-valid configuration sector and makes it the valid one. The Flash update sequence handled by the device is listed in the steps that follow:

1. Initiate sector erase instruction(s) to the non-valid sector, either sector 0 or sector 1 (the non-valid sector is defined by the inverse value of the *SEC1VAL* bit in the EEC register).
2. Copy the shadow RAM to the non-valid sector, with the signature field present in NVM Control Word 1 copied last.
3. Toggle the state of the *SEC1VAL* bit in the EEC register to indicate that the non-valid sector became the valid one and visa versa.
4. Clear the signature field in the valid sector to make it invalid. Since a valid signature is 01b, it is enough to program the bits to 00b, without issuing a sector erase command to the Flash.

Note: Software should be aware that programming the Flash might require a long latency due to the Flash update sequence handled by hardware. The sector erase command by itself can last tens of µs. Software must poll the *FLUDONE* bit in the EEC register to check whether or not the Flash programming completed.

Note: The X540 always effectively updates the Flash after any VPD write access (no use of the EEC.FLUPD bit is required in this case).

Note: Contents of the shadow RAM is reset only at LAN_PWR_GOOD events. It is protected against an ECC error at the shell level in such a way that the probability of an error is close to zero.

Note: Each time the Flash content is not valid (blank configuration sectors or wrong signature on both sector 0 and 1) EEPROM access mode is turned off. Software should rather use either the bit banging interface to the Flash through FLA register or the memory mapped Flash BAR access.

3.4.4 NVM Clients and Interfaces

Note: Access to the NVM should be done exactly according to the flows described in this section. Any read or write access to the NVM that does not follow exactly to the rules and steps listed in this section might lead to unexpected results.

There are several clients that can access the NVM to different address ranges via different access modes, methods, and interfaces. The various clients to the NVM are Software Tools (BIOS, etc.), Drivers, MC (via Firmware), and VPD Software.

Table 3-11 lists the different accesses to the NVM.

Table 3-11 Clients and Access Types to the NVM

Client	NVM Access Method	NVM Access Mode	Logical Byte Address Range	NVM Access Interface (CSRs or Other)
VPD Software	Parallel (32-bits)	EEPROM	0x000000 - 0x000FFF	VPD Address and Data registers, via shadow RAM logic. Any write access is immediately pushed by the X540 into the Flash. VPD module must be located in the first valid Flash sector.
Software	Parallel (16-bits)	EEPROM	0x000000 - 0x000FFF	EERD, EEWR, via shadow RAM logic.
	Parallel (32-bits read, 8-bits write)	Flash	0x000000 - 0x001FFF	Memory mapped via BARs. Accessing this range via Flash BAR should be avoided during normal operation as it might cause non-coherency between the Flash and the shadow RAM.
		Flash	0x002000 - 0xFFFFFF	Memory mapped via BARs.
Software	Bit Banging (1-bit)	Flash	0x000000 - 0x001FFF	FLA. Accessing this range via bit-banging should be avoided during normal operation as it might cause non-coherency between the Flash and the shadow RAM.
			0x002000 - 0xFFFFFFF	FLA

Note: Firmware saves words like SMBus Slave Addresses or Signature, which are saved into the NVM at the firmware's initiative. Note that the VPD module must be mapped to the first valid 4 KB sector.

3.4.4.1 Memory Mapped Host Interface

Using the legacy Flash transactions the Flash is read from, or written to, by the X540 each time the host CPU performs a read or a write operation to a memory location that is within the Flash address mapping or upon boot via accesses in the space indicated by the Expansion ROM Base Address register. Accesses to the Flash are based on a direct decode of CPU accesses to a memory window defined in either:

- Memory CSR + Flash Base Address Register (PCIe Control Register at offset 0x10).
- The Expansion ROM Base Address Register (PCIe Control Register at offset 0x30).
- The X540 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 (Flash word address 0x05). This pointer is loaded by the X540 from the Flash before enabling any access to the Expansion ROM memory space.
  - When modifying the PXE Driver Section Pointer in the NVM, it is required to issue a PCIe reset on which the updated offset is sampled by the hardware.
  - In case there is no valid NVM signature in the two first 4 KB sectors, then expansion ROM BAR is disabled.

Note: The X540 controls accesses to the Flash when it decodes a valid access. Attempt to out of range write access the PCIe Expansion/Option ROM module (according to NVM size field in NVM Control Word 1) is ignored, while read access would return value of 0xDEADBEAF. The X540 supports only byte writes to the Flash.

Note: Flash read accesses are assembled by the X540 each time the access is greater than a byte-wide access.

Note: The X540 byte reads or writes to the Flash take about 2-30 µs time. The device continues to issue retry accesses during this time.

Note: During normal operation, the host should avoid memory mapped accesses to the first two 4 KB sectors of the Flash because it might be non-coherent with the shadow RAM contents.

Caution: Flash BAR access while FLA.FL_REQ is asserted (and granted) is forbidden. It can lead to a PCIe hang as a bit-banging access requires several PCIe accesses.

3.4.4.2 CSR Mapped Host Interface

Software has bit banging or parallel accesses to the NVM or to the shadow RAM (refer to Table 3-11) via registers in the CSR space. The X540 supports the following cycles on the parallel interface: posted write, posted read, sector erase and device erase.

3.4.4.2.1 EEPROM-Mode Host Interface

EEPROM-Mode provides a parallel interface to the first valid 4KB sector of the NVM, aka base sector, which is agnostic to the Flash device type. It also minimizes excessive sector erase cycles to the Flash device by coalescing an update of the whole base sector to a single programming cycle.

3.4.4.2.2 Bit Banging Host Interface

Software can access the Flash directly by using the Flash's 4-wire interface through the Flash Access (FLA) register. It can use this for reads, writes, or other Flash operations (accessing the Flash status register, erase, etc.).

3.4.4.3 MC Interface

The MC can access several fields in the NVM and/or shadow RAM via dedicated NC-SI commands.

3.4.5 Flash Access Contention

Flash accesses initiated through the LAN "A" device and those initiated through the LAN "B" device may occur during the same approximate size window. The X540 does not synchronize between the two entities accessing the Flash so contentions caused from one entity reading and the other modifying the same locations is possible.

To avoid such a contention between software LANs or between software and firmware accesses, these entities are required to make use of the semaphore registers. Refer to Section 11.7.5. 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 timeout of the PCIe transaction made to a memory mapped Flash address while the Flash is currently busy with a long sector erase operation.

Two software entities could however not use the semaphore mechanism: BIOS and VPD software.

- Since VPD software accesses only the VPD module, which is located in the first valid sector of the NVM, VPD accesses are always performed against the shadow RAM first. In this case, hardware must take/release ownership over the NVM as if it was the originator of the Flash access. It is then hardware's responsibility to update the NVM according to the Flash update sequence described in Shadow RAM.
- No contention can occur between BIOS and any other software entity (VPD included) as it accesses the NVM while the operating system is down.
- Contention 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 pushing on 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 poll whether the NVM write has completed before sending the response to the MC NC-SI command.
  - If a system reboot is issued by a local user on the host, there is no technical way to avoid NVM access contention between BIOS and the MC to occur.

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

Note: 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 PHY auto-load while updating the PHY NVM modules.

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

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

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

3.4.6 NVM Read, Write, and Erase Sequences

Refer to Section 6.2.1.1 to establish the required double-image policy prior to updating any Flash module.

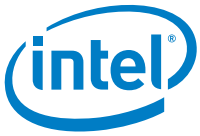
Any software or firmware flow described in this section (excepted for VPD and BIOS) shall be preceded by taking NVM ownership via semaphores as described in Section 11.7.5.

3.4.6.1 Flash Erase Flow by the Host

1. Erase access to the Flash must first be enabled by clearing the *FWE* field in the EEC register.
2. Poll the *FL_BUSY* flag in the FLA register until cleared.
3. Set the *Flash Device Erase* bit (FL_DER) in the FLA register or the *Flash Sector Erase* bit (FL_SER) together with the Flash sector index to be erased (FL_SADDR).
4. Clear the erase enable by setting the *FWE* field to 01b in the EEC register to protect the Flash device.

Note: Trying to erase a sector in the Flash device when writes are disable (FWE=01b) cannot be performed by the X540.

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



3.4.6.2 Software Flow to the Bit Banging Interface

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

1. Write a 1b to the *Flash Request* bit (FLA.FL_REQ).
2. Read 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 and can be found in the appropriate datasheet.
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 datasheet for the opcode to be used for reading the STATUS register.

Note: Bit Banging Interface is not expected to be used during nominal operation. Software/firmware should rather use the EEPROM-Mode when accessing the base sector and the Flash-Mode for other sectors.

Note: If software must use the Bit Banging Interface in nominal operation it should adhere to the following rules:

- Gain access first to the Flash using the flow described in Section 11.7.5
- Minimize FLA.FL_REQ setting for a single byte/word/dword access or other method that guarantee fast enough release of the FLA.FL_REQ.

3.4.6.3 Software Word Program Flow to the EEPROM-Mode Interface

Read Interface:

Software initiates a read cycle to the NVM via the EEPROM-mode by writing the address to be read and the *Start* bit to the EERD register.

As a response, hardware executes the following steps:

1. The X540 reads the data from the shadow RAM.
2. Puts the data in *Data* field of the EERD register.
3. Sets the *Done* bit in the EERD register.

Note: Any word read this way is not loaded into the X540's internal registers. This happens only at an hardware auto-load event.

Write Interface:

Software initiates a write cycle to the NVM via the EEPROM-mode as follows:

1. Poll the *Done* bit in the EEWR register until its set.
2. Write the data word, its address, and the *Start* bit to the EEWR register.

As a response, hardware executes the following steps:

1. The X540 writes the data to the shadow RAM.

2. The X540 sets the *Done* bit in the EEWR register.

Note: In addition, the VPD area of the NVM can be accessed via the PCIe VPD capability structure.

Note: EEPROM-Mode writes are performed into the internal shadow RAM. Section 6.2.1.1 describes the procedure for copying the internal shadow RAM content into the base sector of the Flash device.

3.4.6.4 Flash Program Flow via the Memory Mapped Interface

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

1. Enable Flash BAR writes by setting EEC.FWE to 10b.
2. Poll the *FL_BUSY* flag in the FLA register until cleared.
3. Write the data byte to the Flash through the Flash BAR.
4. Repeat the steps 2 and 3 if multiple bytes should be programmed.
5. Disable Flash BAR writes by setting EEC.FWE to 01b.

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.

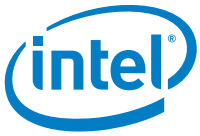
Note: Software must erase the sector prior to programming it.

3.4.7 Signature Field

The only way The X540 can tell if a Flash is present is by trying to read the Flash. The X540 first reads the Control word at word address 0x000000 and at word address 0x000800. 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 and programs its internal registers based on the values read. Otherwise, it ignores the values it reads from that location and does not read any other words.

If the signature bits are valid at both addresses the X540 assumes that the base sector starts at address zero.



3.4.8 Flash Recovery

The first two sectors of the Flash contains fields that if programmed incorrectly might affect the functionality of the X540. The impact might range from an incorrect setting of some function (like LED programming), via disabling of entire features (such as no manageability) and link disconnection, to the inability to access the device via the regular PCIe interface.

The X540 implements a mechanism that enables recovery from a faulty Flash no matter what the impact is, using an SMBus message that instructs the firmware to invalidate the first two sectors of the Flash.

This mechanism uses an SMBus message that the firmware is able to receive in all modes, no matter what is in the content of the first two sectors of the Flash. After receiving this message, firmware erases the first two sectors of the Flash that sets word 0x0 to 0xFF invalidating the signature BIOS or the operating system initiates a power event to force a Flash auto-load process that fails and enables access to the device.

The firmware is programmed to receive such a command only from PCIe reset until one of the functions changes its status from D0u to D0a. Once one of the functions moves to D0a it can be safely assumed that the device is accessible to the host and there is no further need for this function. This reduces the possibility of malicious software to use this command as a back door and limits the time the firmware must be active in non-manageability mode.The command is sent on a fixed SMBus address of 0xC8. The format of the command is SMBus Block Write is as follows:

Function	Command	Data Byte
Release Flash	0xC7	0x12

Note: This solution requires a controllable SMBus connection to the X540.

Note: In case more than one the X540 is in a state to accept this solution, all of the X540 devices connected to the same SMBus accept the command. The devices in D0u state erase the first two sectors of the Flash.

After receiving a Release Flash command, firmware should keep its current state. It is the responsibility of the user updating the Flash to send a firmware reset if required after the entire Flash update process is done.

Data byte 0x12 is the LSB of the X540's default Device ID. The 82575, for example, uses the same command but the data byte there is 0xAA.

An additional command is introduced to enable the write from the SMBus interface directly into any MAC CSR register. The same rules as for the Release Flash command that determine when the firmware accepts this command apply to this command as well.

The command is sent on a fixed SMBus address of 0xC8. The format of the command is SMBus Block Write is as follows:

Function	Command	Byte Count	Data 1	Data 2	Data 3	Data 4	...	Data 7
CSR Write	0xC8	7	Config Address 2	Config Address 1	Config Address 0	Config Data MSB	...	Config Data LSB

The MSB in Configuration Address 2 indicates which port is the target of the access (0 or 1).

The X540 always enables the manageability block after power up. The manageability clock is stopped only if the manageability function is disabled in the Flash and one of the functions had transitioned to D0a; otherwise, the manageability block gets the clock and is able to wait for the new command.

This command allows writing to any MAC or PHY CSR register as part of the Flash recovery process. This command can be used to write to the Flash and update different sections in it.

3.4.9 Flash Deadlock Avoidance

The Flash is a shared resource between the following clients:

1. Hardware auto-read.
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, on which hardware implements the actual access to the Flash. Hardware 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 X540 implements a timeout mechanism, which releases the grant from a client that holds 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.

Note: The bit banging interface does not guarantee fairness between the clients, therefore it should be avoided in nominal operation as much as possible. When write accesses to the Flash are required the software or manageability should access the Flash one word at a time releasing the interface after each word. Software and firmware should avoid holding the Flash bit-bang interface for more than 500 ms.

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

3.4.10 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 VPD capability does not appear in the configuration space.

The maximal area size is 256 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 in the following tables.

Table 3-12 Small Resource Structure

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

Table 3-13 Large Resource Structure

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

The X540 parses the VPD structure during the auto-load process following PCIe reset in order to detect the read only and read/write area boundaries. The X540 assumes the following VPD fields with the limitations listed:

Table 3-14 VPD Structure

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

VPD structure limitations:

- The structure must start with a Tag = 0x82. If the X540 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 3-14, it assumes the area is not programmed and the entire 256 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.
- Both read and write sections on the VPD area must be Dword aligned. For example, each tag must start on Dword boundaries and each data field must end on Dword boundary. Write accesses to Dwords that are only partially in the read/write area are ignored. VPD software is responsible to make the right alignment to allow a write to the entire area.
- The structure must end with a Tag = 0x78. The tag must be word aligned.

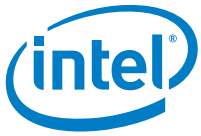
- The VPD area is accessible for read and write via the EEPROM-mode access only. The VPD area can be accessed through the PCIe configuration space VPD capability structure listed in Table 3-14. Write accesses to a read only area or any accesses outside of the VPD area via this structure are ignored.
- VPD area must be mapped to the first valid 4 KB sector of the Flash.
- 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 directed 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 modification is written into the Flash device at the hardware's initiative, once the other entity releases Flash ownership, which may take up to several seconds.

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

The X540 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.



- 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 pins of *both* ports are 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 2 (SDP_FUNC_OFF_EN) in PCIe Control 3 Word (offset 0x07) of the NVM.
- The lowest SDP pins (SDP0_0 and SDP1_0) of the two ports can be combined to encode the NC-SI package ID of the X540. This ability is enabled by setting bit 15 (NC-SI Package ID from SDP) in NC-SI Configuration 2 word (offset 0x07) of the NVM. The 3-bit package ID is encoded as follows: Package ID = [0, SDP1_0, SDP0_0], where SDP0_0 is used for the least significant bit.
- 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 7.9.4.

All SDP pins can be allocated to hardware functions. See more details on IEEE1588 auxiliary functionality in Section 7.9.4 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 then 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 3-15 lists the setup required to achieve each of the possible SDP configurations.

Table 3-15 SDP Settings

SDP	Usage	NVM Settings	GPI Register Settings	ESDP Register Settings		
				SDPx_NATIVE	SDPx_IODIR	SDP1_Function
0	SDP	NC-SI Package ID from SDP = 0	SDP0_GPIEN= 0	0	Input/Output	N/A
	GPI (EICR bit 25)	NC-SI Package ID from SDP = 0	SDP0_GPIEN= 1	0	Input	
	NC-SI package ID	NC-SI Package ID from SDP = 1	SDP0_GPIEN= 0	0	Input	
	1588 functionality: Drive Target Time 0 /Clock Out	NC-SI Package ID from SDP = 0	SDP0_GPIEN= 0	1	Output	
1	SDP	SDP_FUNC_OFF_EN = 0	SDP1_GPIEN= 0	0	Input/Output	
	GPI (EICR bit 26)	SDP_FUNC_OFF_EN = 0	SDP1_GPIEN= 1	0	Input	
	PCI disable	SDP_FUNC_OFF_EN = 1	SDP1_GPIEN= 0	0	N/A	
	1588 functionality: Drive Target Time 1	SDP_FUNC_OFF_EN = 0	SDP1_GPIEN= 0	1	Output	0
	Thermal sensor hot indication	SDP_FUNC_OFF_EN = 0	SDP1_GPIEN= 0	1	Output	1
2	SDP	N/A	SDP2_GPIEN= 0	0	Input/Output	N/A
	GPI (EICR bit 27)		SDP2_GPIEN= 1	0	Input	
	1588 functionality: Sample time in Auxiliary Time Stamp 0 register		SDP2_GPIEN= 0	1	Input	
3	SDP	N/A	N/A	0	Input/Output	
	1588 functionality: Sample time in Auxiliary Time Stamp 1 register			1	Input	

3.6 Network Interface

3.6.1 Overview

The X540 provides dual-port network connectivity with copper media. Each port includes integrated MAC-PHY functionalities and can be operated at either 10 GbE, 1 GbE, or 100 BASE-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.

Note: Designers are 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.

3.6.2 Internal MDIO Interface

The X540 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.

Note: 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.

Note: 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 3.6.3.4.3).

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

To execute 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 bit MSCA.MDICMD set to 1b.
5. Wait for bit MSCA.MDICMD 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 bit MSCA.MDICMD set to 1b.
4. Wait for bit MSCA.MDICMD 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.

Note: A read-increment-address flow is performed if the *OPCODE* field is set to 10b in step 2. The address is incremented internally once data is read at step 5 so that no address cycle is needed to perform a data read from the next address.

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

3.6.3 Integrated Copper PHY Functionality

3.6.3.1 PHY Performance

3.6.3.1.1 Reach

Table 3-16 BER and Ranges vs. Link Speed and Cable Types

Speed	Cable	Committed Reach	Committed BER
10GBASE-T	CAT-7	Full reach: 100 m	$< 10^{-16}/10^{-12}$
	CAT-6a	Full reach: 100 m	
	CAT-6	55 m	
1000BASE-T	CAT-5e	Full reach: 130m/100 m	$< 10^{-15}/10^{-10}$
100BASE-TX	CAT-5e	Full reach: 130m/100 m	$< 10^{-14}/10^{-8}$

Note: Reaches specified in Table 3-16 refer to real cable lengths and not to the IEEE standard model.

3.6.3.1.2 MDI / Magnetics Spacing

The X540 supports a variable distance of 0 to 4 inches with the magnetics.

3.6.3.1.3 Cable Discharge

The X540 is capable passing the Intel cable discharge test.

3.6.3.2 Auto-Negotiation and Link Setup

Link configuration is determined by PHY auto-negotiation with the link partner. The software device driver must change auto-negotiation settings in cases where a successful link is not negotiated or the designer desires to change link properties. Note that the link partner should always have auto-negotiation enabled.

3.6.3.2.1 Automatic MDI Cross-Over and Lane Inversion

Note: The X540 uses an automatic MDI/MDI-X configuration. Intel recommends using straight through cables. Where crossover cables are used, all four pairs must be crossed. Using crossover cables where only some pairs are crossed is not supported and might result in link failure or slow links.

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. The default setting is ABCD on PHY0 and DCBA to PHY1. One dedicated pin per PHY (PHY0_RVSL / PHY1_RVSL) is controlling the MDI configuration for MDI reversal, such as ABCD to DCBA pair inversion. It is also configurable via provisional PHY register 1.E400.

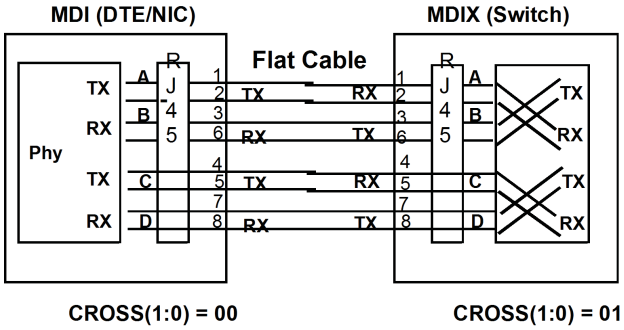


Figure 3-12 Cross-Over Function

3.6.3.2.2 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 in order 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)

Note: 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 3.6.5.

Note: 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.

Note: The X540 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 X540, 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.

3.6.3.2.2.1 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, then 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.

3.6.3.2.2.2 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 3-17 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 3-17 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.

3.6.3.2.3 Fast Retrain

In 10GBASE-T mode, the X540 PHY supports the Cisco Fast Retrain mode. If enabled, the PHY upon losing frame can inject a programmable ordered set onto the line that tells the far-end PHY to implement a very short resynchronization sequence to enable the near-end PHY to re-acquire frame synchronization. This saves roughly four seconds off of the link-reconnection time on simple link breaks, as the two second link break time-out and re-auto-negotiating.

This X540 feature requires that the far-end PHY support this proprietary mode as well. Fast Retrain capability Exchange is done during the auto-negotiation flow.

Fast Retrain mode is enabled via PHY registers 1E.C475 and 1.E400.

3.6.3.3 PHY Initialization

3.6.3.3.1 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 X540 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.

3.6.3.3.2 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. If running in 10GBASE-T mode, and power minimization mode is enabled, shut down unused taps.
8. Verify error-free operation.
9. Enter steady state.

3.6.3.3.3 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 3.6.3.3.1).

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 Section 4-4.

At PHY reset events, all the PHY functionalities 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.

3.6.3.4 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

3.6.3.4.1 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).

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

3.6.3.4.2 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 3-18 PHY Link State Registers

Register Bits	Name	Description
7.C800 2:1	Connect Rate	0x3 = 10GBASE-T. 0x2 = 1000BASE-T. 0x1 = 100BASE-TX. 0x0 = 10BASE-T.
7.C800 0	Connect Type (Duplex)	1b = Full. 0b = Half.
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 auto-negotiation). 0x06 = Test mode. 0x07 = Loopback mode. 0x08 = Reserved. 0x09 = Reserved. 0x0A = Reserved. 0x0B:0x10 = Reserved.

3.6.3.4.3 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.

3.6.3.4.4 PHY Interrupt Handling Flow

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

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

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 shall 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 11.7.5 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 shall 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.
9. Prior to do a PHY reconfiguration that might drop the link (e.g. restart auto-negotiation), the host must wait until the VETO bit is read as 0b
10. Host releases PHY semaphore.

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

Note: 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.

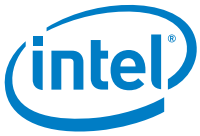
Note: PHYINT_STATUS registers and EEMNGCTL.CFG_DONE bits are reset by hardware only at power-up events.

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

3.6.3.5 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 3-20 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 3-19 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 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).

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.

3.6.3.6 Low Power Operation and Power Management

The PHY incorporates numerous features to maintain the lowest power possible. Refer to Section 5.2.3.2.

3.6.4 Loopback Support

In external PHY loopback mode, data is sent through the MDI interface and looped back using an external loopback plug. The X540 enters this mode by setting bit F in the PMA Receive Reserved Vendor Provisioning 1 register (1.E400.F).

3.6.5 Ethernet Flow Control (FC)

The X540 supports flow control as defined in 802.3x, as well as the specific operation of asymmetrical flow control defined by 802.3z. The X540 also supports Priority Flow Control (PFC), known as Class Based Flow Control, as part of the DCB architecture.

Note: The X540 can either be configured to receive regular FC packets or PFC packets. The X540 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 then Tx CRC by hardware should be enabled as well (HLREG0.TXCRCEN = 1b).
- Flow Control Address Low, High (RAL[0], RAH[0]) — 6-byte FC multicast address.
- Priority Flow Control Type Opcode (PFCTOP) — Contains the type and OpCode values for PFC.
- Flow Control Receive Threshold High (FCRTH[7:0]) — A set of 13-bit high watermarks indicating receive buffer fullness. A single watermark is used in link FC mode and up to eight watermarks are used in PFC mode.
- Flow Control Receive Threshold Low (FCRTL[7:0]) — A set of 13-bit low watermarks indicating receive buffer emptiness. A single watermark is used in link FC mode and up to eight watermarks are used in PFC mode.

- Flow Control Transmit Timer Value (FCTTV[3:0]) — A set of 16-bit timer values to include in transmitted PAUSE frame. A single timer is used in link FC mode and up to eight timers are used in PFC mode.
- Flow Control Refresh Threshold Value (FCRTV) — 16-bit PAUSE refresh threshold value (in legacy FC FCRTV[0] must be smaller than FCTTV[0])

3.6.5.1 MAC Control Frames and Reception of Flow Control Packets

3.6.5.1.1 MAC Control Frame — Other than FC

The IEEE specification reserved the Ethertype value of 0x8808 for MAC control frames, which are listed in Table 3-20.

Table 3-20 MAC Control Frame Format

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

3.6.5.1.2 Structure of 802.3X FC Packets

802.3X FC packets are defined by the following three fields (see Table 3-21):

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 3-21 802.3X Packet Format

DA	01_80_C2_00_00_01 (6 bytes).
SA	Port Ethernet MAC Address (6 bytes).
Type	0x8808 (two bytes).
Opcode	0x0001 (two bytes).
Time	XXXX (two bytes).
Pad	42 bytes.
CRC	Four bytes.

3.6.5.1.3 PFC

DCB introduces support for multiple TCs assigning different priorities and bandwidth per TC. Link-level FC (PAUSE) stops all the TCs. PFC, known as Class Based Flow Control or CBFC, allows more granular FC 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 X540. 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 X540 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.

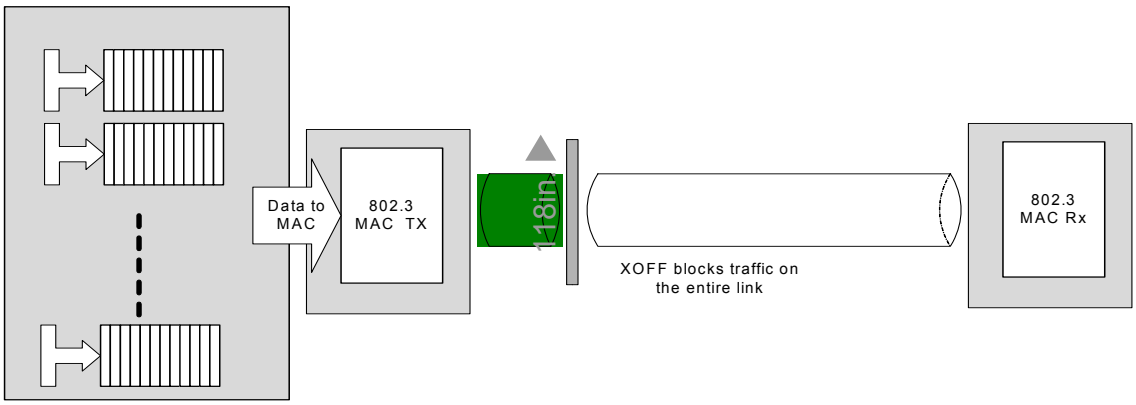


Figure 3-13 802.3X Link Flow Control (PAUSE)

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 3-14.

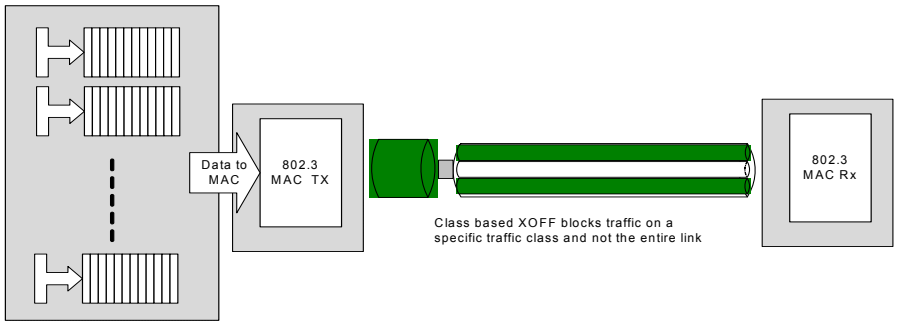


Figure 3-14 PFC

Table 3-22 Packet Format for PFC

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

Table 3-23 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 X540 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 X540, the same values are replicated into every *Timer* field and priority enable vector bit of all the UPs bound to the associated TC. These values are configured in the RTRUP2TC register.

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

3.6.5.1.4 Operation and Rules

The X540 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, then the device sets the timer to the maximum value among all enabled *Timer* fields associated with the TC.
- Disable packet transmission or schedule the disabling of transmission after the current packet completes.
  - For PFC, this is done per paused TC
  - Tx manageability traffic is bound to a specific TC as defined in the MNGTXMAP register, and is thus paused when its TC is paused

Resumption of transmission can occur under the following conditions:

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

— For PFC, this is done per TC

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

3.6.5.1.5 Timing Considerations

When operating at 10 GbE line speed, the X540 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 X540 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 X540 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 2.3, proposes that the tolerated response time for Priority XOFF frames are the same as Link XOFF frames with extra budget of 19360 bit times if MACSec is used, or of 2 pause quanta otherwise. This extra budget is aimed to compensate the fact that decision to stop new transmissions from a specific TC must be taken earlier in the transmit data path than for the Link Flow Control case.

3.6.5.2 PAUSE and MAC Control Frames Forwarding

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

- The *DPF* bit controls the transfer of PAUSE packets to the host. The same policy applies to both link FC and PFC packets as listed in Table 3-24. 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) then link FC frames are considered as MAC control frames for this case. Similarly, when PFC frames are not enabled (RPFCM = 0b) then 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 7.10.3.4.

Table 3-24 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.

3.6.5.3 Transmitting PAUSE Frames

The X540 generates PAUSE packets to ensure there is enough space in its receive packet buffers to avoid packet drop. The X540 monitors the fullness of its receive FIFOs and compares it with the contents of a programmable threshold. When the threshold is reached, the X540 sends a PAUSE frame. The X540 supports both link FC and PFC — but not both concurrently. When DCB is enabled, it only sends PFC, and when DCB is disabled, it only sends link FC.

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

3.6.5.3.1 PFC Mode

The X540 FC operates in either a link 802.3X compliant mode or in a PFC mode, but not in both at the same time.

The same FC 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 PFC.

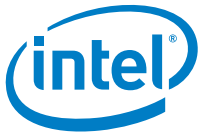
Specific considerations for generating PFC packets:

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

3.6.5.3.2 Operation and Rules

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

The content of the Flow Control Receive Threshold High (FCRTH) register determines at what point the X540 transmits the first PAUSE frame. The X540 monitors the fullness of the receive FIFO and compares it with the contents of FCRTH. When the threshold is reached, the X540 sends a PAUSE frame with its pause time field equal to FCTTV.



At this time, the X540 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 X540 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 X540 sends a PAUSE frame if it has previously sent one and the FIFO overflows. This is intended to minimize the amount of packets dropped if the first PAUSE frame did not reach its target.

3.6.5.3.3 Flow Control High Threshold — FCRTH

The X540 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 three 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)
3. Whether or not MACsec is enabled or disabled over the 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 + (MACSec enabled = yes) 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 + (MACSec enabled = yes) 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 shall 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 X540.

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 then 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 X540:

The X540 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 + (MACSec enabled = yes) x (3,200)] + 2 x MaxFrame(TC) + (MACSec enabled = yes) x MaxFrame(link)

FCRTH must be set to the size of the Rx packet buffer allocated to the traffic class minus the X540 DV.

Table 3-25 X540 Delay Values (DV) used for FCRTH

MACsec Enabled	9.5 KB Jumbo Enabled	FCoE Traffic Class	X540 DV
No	No	No	24 KB
No	No	Yes	25 KB
No	Yes	No	50 KB
No	Yes	Yes	35 KB
Yes	No	No	27 KB
Yes	No	Yes	27 KB
Yes	Yes	No	60 KB
Yes	Yes	Yes	45 KB

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

3.6.5.3.4 FC Low Threshold — FCRTL

The low threshold value is aimed to protect against wasted available host bandwidth. There is some latency from the time that the low threshold is crossed until the XON frame is sent and packets are received from the link partner. The low threshold shall 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, then 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 shall 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.

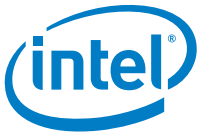


Table 3-26 X540 FCRTL

9.5 KB Jumbo Enabled	FCoE Traffic Class	X540 DV
No	No	6 KB
No	Yes	7 KB
Yes	No	21 KB
Yes	Yes	7 KB

3.6.5.3.5 Packet Buffer Size

When PFC is enabled, the total size of a TC packet buffer must be large enough for the Low and high thresholds. In order 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 X540 DV *(see* Section 3.6.5.3.3*)*
- For a best effort TC:
  - Rx Packet Buffer size = FCRTL, as the same considerations than described in Section 3.6.5.3.4 play here to avoid bubbles over PCIe

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

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

Table 3-27 Some Recommended Rx Packet Buffer Settings

Flow Director Table Size	MACsec Enabled	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	No	No	33 KB	62 KB FCRTL = 7 KB FCRTH = 37 KB	61 KB FCRTL = 6 KB FCRTH = 37 KB
No	No	Yes	27 KB	52 KB FCRTL = 7 KB FCRTH = 17 KB	86 KB FCRTL = 21 KB FCRTH = 36 KB
No	Yes	No	32 KB	64 KB FCRTL = 7 KB FCRTH = 37 KB	63 KB FCRTL = 6 KB FCRTH = 36 KB
No	Yes	Yes	22 KB	57 KB FCRTL = 7 KB FCRTH = 12 KB	91 KB FCRTL = 21 KB FCRTH = 31 KB
64 KB	No	No	25 KB	54 KB FCRTL = 7 KB FCRTH = 29 KB	53 KB FCRTL = 6 KB FCRTH = 29 KB
64 KB	No	Yes	19 KB	44 KB FCRTL = 6 KB FCRTH = 9 KB	78 KB FCRTL = 20 KB FCRTH = 28 KB
64 KB	Yes	No	24 KB	56 KB FCRTL = 7 KB FCRTH = 29 KB	55 KB FCRTL = 6 KB FCRTH = 28 KB
64 KB	Yes	Yes	14 KB	49 KB FCRTL = 3 KB FCRTH = 4 KB	83 KB FCRTL = 18 KB FCRTH = 23 KB
128 KB	No	No	17 KB	46 KB FCRTL = 7 KB FCRTH = 21 KB	45 KB FCRTL = 6 KB FCRTH = 21 KB
128 KB	Yes	No	16 KB	48 KB FCRTL = 7 KB FCRTH = 21 KB	47 KB FCRTL = 6 KB FCRTH = 20 KB

Note: In some of the cases, it has been necessary to get compromised on the rules for hysteresis and FCRTL in order to fit the size available for Rx packet buffer.

Note: 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.

Note: In all cases, FCRTH rule has been applied as is, since compromising on it is not allowed and extending it provides no performance benefits.

3.6.5.4 Link FC in DCB Mode

When operating in DCB mode, PFC is the preferred method of getting the best use of the link for all TCs. When connecting to switches that do no support (or enable) PFC, the X540 can also throttle the traffic according to incoming link FC notifications. Following is the required device setting and functionality.

- The X540 should be set to legacy link FC by setting MFLCN.RFCE.
- Receive XOFF pauses transmission in all TCs.
- Crossing the Rx buffer high threshold on any TC generates XOFF transmission. Each TC can have its own threshold configured by the FCRTH[n] registers.
- Crossing the Rx buffer Low threshold on any TC generates XON transmission. This behavior is undesired. Therefore, software should not enable XON in this mode by clearing FCRTL[n].XONE bits in all TC.
- The FCTTV of all TCs must be set to the same value.

3.6.6 Inter Packet Gap (IPG) Control and Pacing

The X540 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 Pacing Speeds at 10 GbE Link Speed, 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 3-28 lists the pacing configurations supported by The X540 at link rates of 10 GbE. When operating at lower link speeds the pacing speed is proportional to the link speed.

Table 3-28 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.

4.0 Initialization

4.1 Power Up

4.1.1 Power-Up Sequence

Figure 4-1 shows the X540's power-up sequence from power ramp up until it is ready to accept host commands.

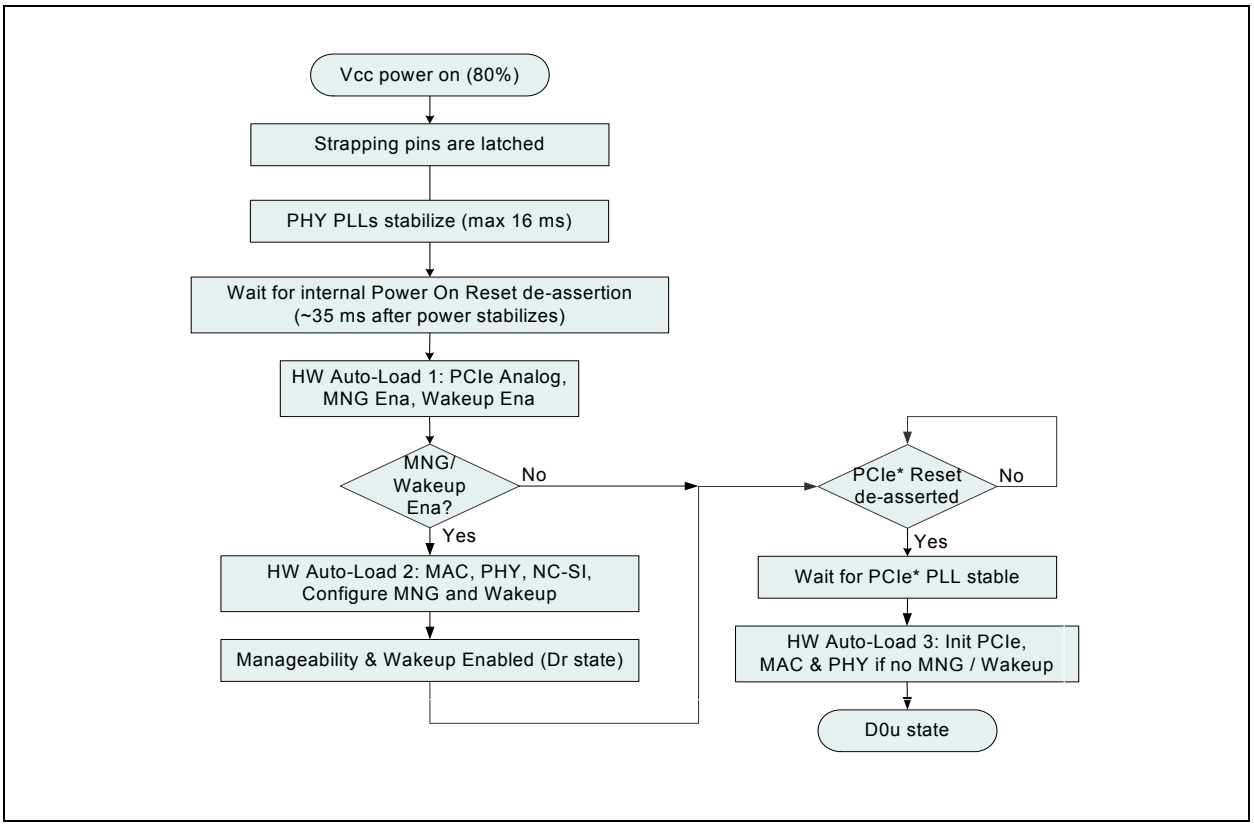


Figure 4-1 X540 Power-Up Sequence

4.1.2 Power-Up Timing Diagram

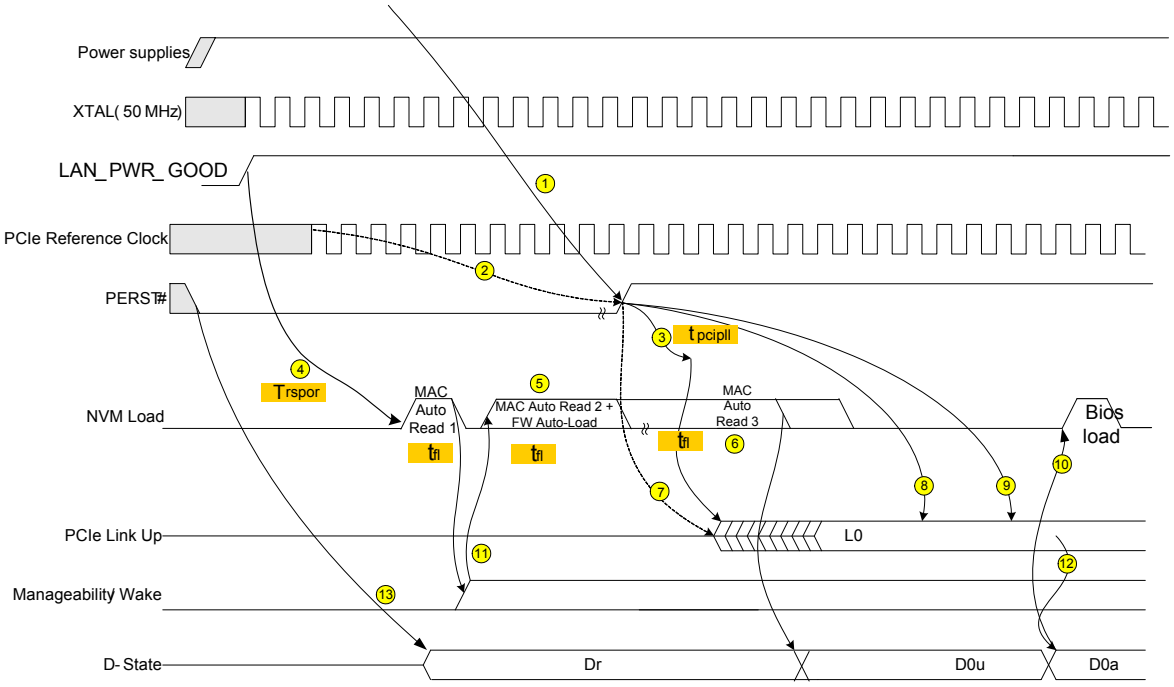


Figure 4-2 Power-Up Timing Diagram

Table 4-1 Notes for Power-Up Timing Diagram

Note	
1	PCIe reset is de-asserted by minimum t_{PVPGL} after power is stable (PCIe specification).
2	The PCIe reference clock is valid $t_{PWRGD-CLK}$ before the de-assertion of PCIe reset (PCIe specification).
3	Deassertion of PCIe reset to PCIe PLL stable t_{PCIPLL}.
4	NVM read starts following the rising edge of the internal LAN_PWR_Good signal. First Flash auto-read sequence is owned by the MAC to load MNG enable, APM enable, PCIe Analog, and other configuration words from legacy NVM initialization section.
5	Interleaved auto-read sequences between MAC and PHY. In this second MAC auto-load sequence, MAC manageability and wake-up modules are loaded (if manageability / wake up enabled).
6	MAC auto-load 3: PCIe general configuration; PCIe configuration space; LAN core modules, and MAC module if manageability is not enabled.
7	PCIe link training starts after t_{pgtrn} from PCIe reset de-assertion (PCIe specification).
8	A first PCIe configuration access might arrive after t_{pgcfg} from PCIe reset de-assertion (PCIe specification).

Note	
9	A first PCI configuration response can be sent after t_{pgres} from PCIe reset de-assertion (PCIe specification).
10	BIOS software reads the PCIe driver and iSCSI/FCoE boot code from Flash, via expansion ROM.
11	APM wake up and/or manageability active, based on NVM contents (if enabled).
12	Setting the *Memory Access Enable* or *Bus Master Enable* bits in the PCI Command register transitions the X540 from D0u to D0 state.
13	NVM read starts following the rising edge of the internal LAN_PWR_Good signal. First Flash auto-read sequence is owned by the MAC to load MNG enable, APM enable, PCIe Analog, and other configuration words from legacy NVM initialization section.

4.1.2.1 Timing Requirements

The X540 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 X540 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 us	21,000 us	Uses an internal timer based on 50MHz clock.
T_{bgg}	Bandgap good indication is checked.	5,280 us	10,560 us	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 us	5,200 us	worst case timing is when this step needs to iterate.
T_{pll}	Lock detection of one PLL is checked.	5,280 us	10,560 us	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 us	1.1 us	
T_{efuse}	Read EFUSE content.	500 us	500 us	
t_{ppg}	Internal MAC power good delay from valid power rail.	37,423 us	63,581 us	Typically the time will be no worse than 42 ms.
t_{fl}	Flash auto-read duration.		3 ms	
topll	PCIe reset to start of link training.		10 ms	
tpcipll	PCIe reset to PCIe PLL stable.		5 ms	
tpgtrn	PCIe reset to start of link training.		20 ms	According to PCIe specification.
tpgres	PCIe reset to first configuration response cycle.	100 ms		According to PCIe specification.

4.1.3 Main-Power / Aux-Power Operation

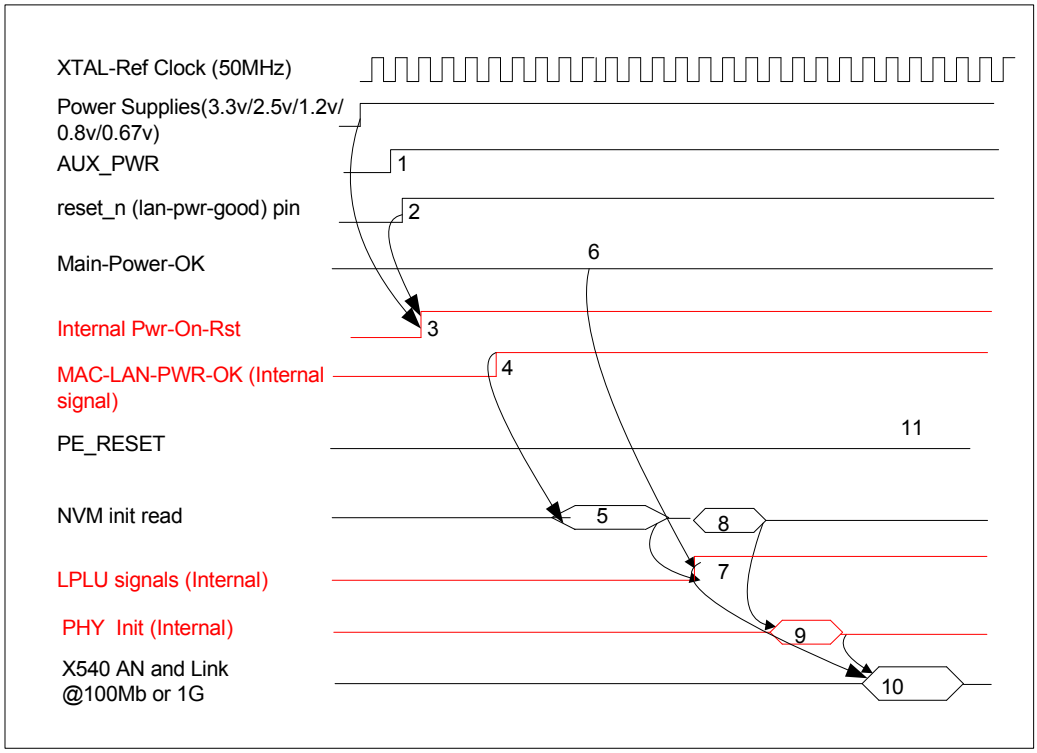


Figure 4-3 Low-Power Modes Timing Diagram

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

Note	
1	AUX_PWR set to 1b, indicates that the X540 should support Aux-Power Mode.
2	RESET_N signal deasserted.
3	Internal Power-On-Reset deasserted, indicates to all the X540 blocks that all power-rails are OK and the external Reset signal is deasserted (reset_n).
4	MAC_LAN_PWR_OK deasserted (internal signal), indicates that all clocks are stable.
5	MAC reads configuration from NVM.
6	Main-Power-OK still deasserted, 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.

Note	
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 deasserted, indicates that the system is still down, keep PCIe at reset, and the MAC at low-power modes (WoL or MNG).

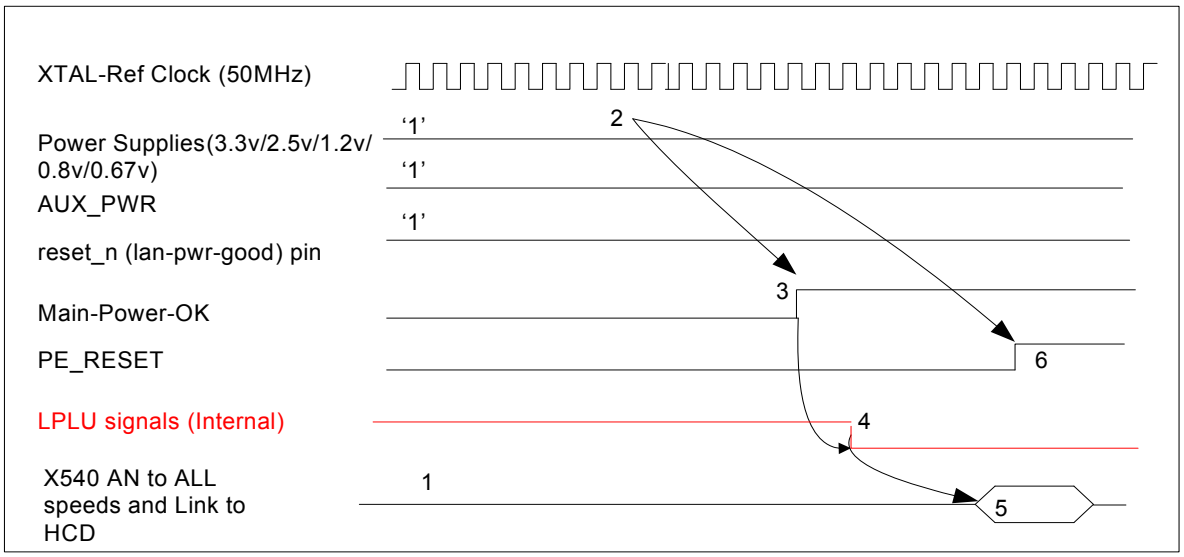


Figure 4-4 Auxiliary-to-Main Power

Note	
1	Aux-Power Mode: LPLU-En, PHY link at 100 Mb/s or GbE, system is down (PE-RESET = 0b), or Main-Power-OK is deasserted.
2	Power-Supplies moved from Aux-Power to Main-Power.
3	Main-Power-OK is asserted, indicates the X540 needs to move to the highest possible speed.
4	LPLU Deasserted (Internal signal), indicates the PHY needs to restart auto-negotiation at the highest possible link.
5	PHY starts auto-negotiation.
6	System turned on, PE_RESET deasserted and PCIe established. Note that this is orthogonal to the PHY link.

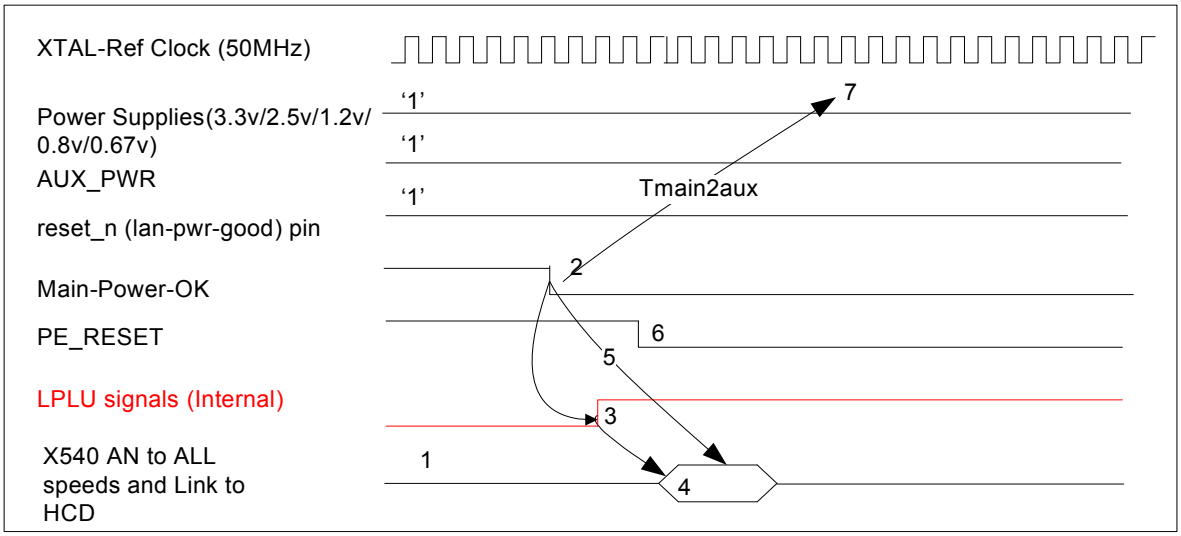


Figure 4-5 Main-to-Auxiliary Power

Note	
1	Main-Power Mode: PHY Link is at maximum speed, system is up, and Main-Power-OK is asserted.
2	System starts the flow of moving Power-Supplies from Main-Power to Aux-Power. Deasserted the Main-Power-OK signal.
3	LPLU asserted (internal signal), indicates the PHY needs to restart auto-negotiation to the lowest possible link.
4	PHY starts auto-negotiation.
5	PHY has 5 ms to move from a high-speed link to auto-negotiation.
6	System turned off, PE_RESET asserted. Note that this is orthogonal to the PHY link.
7	Power supplies actually moved from Main-Power to Aux-Power. $T_{main2aux}$ must be at least 10 ms.

4.2 Reset Operation

4.2.1 Reset Sources

The X540 reset sources are described in the sections that follow.

4.2.1.1 LAN_PWR_GOOD

The X540 has an internal mechanism for sensing power pins. Once power is up and stable, the X540 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)

Deasserting 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 X540 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 X540 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 transitioned from D3hot to D0.

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 doesn't require driver intervention in order to stop the master transactions of this function. FLR affects the device 1 parallel clock cycle from FLR assertion by default setting, or any other value defined by the *FLR Delay Disable* and *FLR Delay* fields in the PCIe Init Configuration 2 — Offset 0x02 word in the NVM.

4.2.1.5.2 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.

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 X540 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 5.2.5.3.2.

If DCB is enabled then 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 map 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 5.2.5.3.2). 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 in order 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.RST bit. In pass-through mode it is generated when receiving a ForceTCO SMB 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 X540 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 5.2.5.3.2.

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 11.7 for more details on *MNG_VETO* bit.
2. Take ownership of the relevant PHY using the flow described in Section 11.7.5.
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 11.7.5.

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 X540 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 X540 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	FW Reset	Force TCO	Notes
NVM read	See Section 6.4.1					
LTSSM (back to detect/polling)	X	X	X			
PCIe link data path	X	X	X			
PCI configuration registers (RO)	X	X	X			9
PCI configuration registers (RW)	X	X	X			9

Table 4-5 Reset Effects — Common Resets

Reset Activation	LAN Power Good	PCIe PE_RST_N	In-Band PCIe Reset	FW Reset	Force TCO	Notes
PCIe local registers	X					
Data path	X	X	X		X	2, 7
MAC, TimeSync, MACsec, and IPsec	X	X 6	X 6		X	
PHY (excluding PLLs)	X	X 6	X 6		X	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	SW Reset	Link Reset or Exit from LAN Disable	PHY Image Reload or PHY Reset	Notes
NVM read	See Section 6.4.1					
LTSSM (back to detect/polling)						
PCIe link data path						
PCI configuration registers (RO)						9
PCI configuration registers (RW)	X	X				9
Data path and memory space, TimeSync, MACsec, and IPsec	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

Table 4-6 Reset Effects — Per Function Resets

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

Table 4-7 Reset Effects -Virtual Function Resets

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

Table 4-5 Through Table 4-7 Notes:

1. If AUX_PWR = 0b the wake up context is reset (*PME_Status* and *PME_En* bits should be 0b at reset if the X540 does not support PME from D3cold).
2. The following register fields do not follow the general rules previously described:
   a. ESDP registers — Reset on LAN Power Good only.
   b. LED configuration registers — Reset on LAN Power Good and on SW Reset events.
   c. The *Aux Power Detected* bit in the PCIe Device Status register is reset on LAN Power Good and PCIe reset only.
   d. FLA — Reset on LAN Power Good only.
   e. RAH/RAL[n, where n>0], MTA[n], VFTA[n], FHFT_*[n], TDBAH/TDBAL, and RDBAH/RDVAL registers have no default value. If the functions associated with these registers are enabled they must be programmed by software. Once programmed, their value is preserved through all resets as long as power is applied.
   f. Statistic registers (physical function)
3. The wake up context is defined in the PCI Bus Power Management Interface specification (sticky bits). It includes:
   a. *PME_En* bit of the Power Management Control/Status Register (PMCSR).
   b. *PME_Status* bit of the PMCSR.
   c. *Aux_En* bit in the PCIe registers.
   d. The device requester ID (since it is required for PM_PME TLP).

The shadow copies of these bits in the Wake Up Control (WUC) register are treated identically.

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:
   a. WUS 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:
   a. The Tx packet buffers.
   b. The Rx packet buffers.
   c. IPsec Tx SA tables.
   d. IPsec Rx SA tables.
8. Sticky bits and hardware init bits (indicated as HwInit) in the PCI Configuration registers are cleared only by LAN Power Good reset.

   The following register fields are not affected by FLR and PFLR:

   - Max_Payload_Size is the Device Control register
   - Active State Power Management (ASPM) Control in the Link Control register
   - Common Clock Configuration in the Link Control register
9. These registers include:
   a. VFEICS.
   b. VFEIMS.
   c. VFEIAC.
   d. VFEIAM.
   e. VFEITR 0-2.
   f. VFIVAR0.
   g. VFIVAR_MISC.
   h. VFPBACL.
   i. VFMailbox.
10. These registers include:
    a. VFEICS.
    b. VFEIMS.
    c. VFEIMC.
    d. VFEIAC.

e. VFEIAM.

f. VFEICR.

g. EITR 0-2.

h. VFIVAR0.

i. VFIVAR_MISC.

j. VFPBACL.

k. VFMailbox.

l. VFMBMEM.

m. RSCINT

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

Note: Unless specified otherwise the X540'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 for details on disabling and enabling an Rx queue.

See Section 4.6.8.1 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 X540 provides support for selectively enabling or disabling one or both LAN device(s) in the system.

Note: The X540 can be configured to have LAN port 1 permanently disabled (X540 single port configuration). In this case, a dedicated single port NVM should be used for power savings.

4.4.2 Overview

Device presence (or non-presence) must be established early during BIOS execution, in order 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 X540's LAN-disable mechanism is implemented in order to be compatible with such a solution.

The X540 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), two different behaviors are possible:

- Dummy function mode — In some systems, it is required to keep all the functions at their respective location, even when other functions are disabled. In dummy function mode, if function #0 (either LAN0 or LAN1) is disabled, then it does not disappear from the PCIe configuration space. Rather, the function presents itself as a dummy function. The device ID and class code of this function changes to other values (dummy function device ID 0x10A6, class code 0xFF0000). In addition, the function does not require any memory or I/O space, and does not require an interrupt line.
- Legacy mode — When function 0 is disabled (either LAN0 or LAN1), then the port residing on function 1 moves to reside on function 0. Function 1 disappears from the PCI configuration space.

Mapping between function and LAN ports is listed in the following tables.

Table 4-8 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	LAN1	Disable
LAN 1 is disabled.	x	LAN 0	Disable
Both LAN functions are disabled.	Both PCI functions are disabled. Device is in low-power mode.		

Table 4-9 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	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 in legacy mode, the LAN port associated originally with function 1 appears in function 0. Function 1 disappears from the PCI configuration space.
- When function 0 is disabled in dummy function mode, 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.
- 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 NVM *LAN PCI Disable* bit.
- The *LAN Disable Select* NVM field indicates which function is disabled.
- The *LAN Function Select* NVM bit defines the correspondence between LAN port and PCI function.

- The *Dummy Function Enable* NVM bit enables the dummy function mode. Default value is disabled.
- The *SDP_FUNC_OFF_EN* NVM 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** LAN 1 can be disabled in the NVM by setting the *LAN PCI Disable* bit in the PCIe Control 2 – Offset 0x05 word. The *LAN Disable Select* in the same word in the NVM selects which LAN is disabled. Furthermore, if the LAN port at function 0 is disabled, the *Dummy Function Enable* in the same word chooses between filling the disabled function by a dummy function, or moving the other LAN port to function 0.

Note: Mapping LAN0 and LAN1 to PCI function 0 and PCI function 1 is controlled by the *LAN Function Select* field in the PCIe Control 2 – Offset 0x05 word in the NVM.

LAN0 **and** LAN 1 can be disabled on the board level by driving the LAN0_Dis_N and LAN1_Dis_N pins 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 both SDP0_1 and SDP1_1 pins 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 *SDP_FUNC_OFF_EN* bit in PCIe Control 3 word (offset 0x07) of the NVM.

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 the LAN Function at Boot Time by the Using 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 X540 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 X540 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 the PCIe Functions at Boot Time by the Using 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 X540 should be disabled.
3. The BIOS modifies the *SDP_FUNC_OFF_EN* bit in PCIe Control 3 word in the NVM and it might eventually issue an in-band PCIe reset that causes the X540 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 X540 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 X540 is no longer a multi-function device. The X540 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 X540 reports 0x0 in this field to signify single-function capability.

4.4.4.4 Interrupt Use

When both LAN devices are enabled, the X540 uses the PCI legacy interrupts of both ports for interrupt reporting. The NVM configuration controls the *Interrupt Pin* field of the PCI configuration header to be advertised for each LAN device to comply with PCI specification requirements.

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

4.4.4.5 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 X540 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

When both LAN ports are disabled following a power on reset / LAN Power Good/ PCIe reset/ in-band reset, the LANx_DIS_N signals should be tied statically to low. At this state, the X540 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_OFF_EN bit was set in NVM. As an example, digital I/O pins are in an electrical off state where pull-up/pull-down resistors are at their defined values.

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 X540 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 X540 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 X540, especially initialization steps. These include:

- General hardware power-up state
- Basic device configuration
- Initialization of transmit
- Receive operation
- Link configuration
- Software reset capability
- Statistics
- Diagnostic hints

4.6.2 Power-Up State

When the X540 powers up, it automatically reads the NVM. The NVM contains sufficient information to bring the link up and configure the X540 for manageability and/or APM wake up. However, software initialization is required for normal operation.

4.6.3 Initialization Sequence

The following sequence of commands is typically issued to the device by the software device driver in order to initialize the X540 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 3.6.3.2 and Section 3.6.3.3.
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. 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 IMS register.

4.6.3.2 Global Reset and General Configuration

Global reset = software reset + link reset.

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

To enable flow control, program the FCTTV, FCRTL, FCRTH, FCRTV and FCCFG registers. If flow control is not enabled, these registers should be written with 0x0. If Tx flow control is enabled, then Tx CRC by hardware should be enabled as well (HLREG0.TXCRCEN = 1b). Refer to Section 3.6.5.3.2 through Section 3.6.5.3.5 for recommended settings of the Rx packet buffer sizes and flow control thresholds. Note that FCRTH[n].RTH fields must be set by default regardless if flow control is enabled or not. Typically, the FCRTH[n] default value should be equal to RXPBSIZE[n]-0x6000. 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 3.6.3.3 and Section 3.6.3.2 for the initialization and link setup steps. The device driver uses the MDIO register to initialize the PHY and setup the link. Section 3.6.2 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.LinkStatus Reflects the PHY internal indication to the MAC.

LINKS.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 RQSMR register section for RQSMR setup, and to 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 registers according to the preferred mode of operation.
- Software clears EICR by writing all 1s to clear old interrupt causes.
- Software enables the required interrupt causes by setting the EIMS[n] registers.

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 enables the required interrupt causes by setting the EIMS[n] registers.

4.6.7 Receive Initialization

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

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

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

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

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

Initialize the flexible filters 0…5 — 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, SAQF, DAQF, SDPQF, FTQF, SYNQF, ETQF, ETQS, RDRXCTL, RSCDBU.

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 one of the following two cases:

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

2. MACsec encapsulation is expected. In these cases, set MAXFRS.MFS to the expected maximum packet size plus 32 bytes for the MACsec encapsulation. See MAXFRS.MFS register/bit description for the correct handling of VLAN and double VLAN headers.

Enable receive coalescing if required as described in Section 4.6.7.2.

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 RDBAL, 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 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:

- Disable the routing of packets to this queue by reconfiguring the Rx filters. In order to ensure that the receive packet buffer does not contain any packets to the specific queue it is required to follow the Flushing the Packet Buffers procedure described later.

- If RSC is enabled on the specific queue and VLAN strip is enabled as well then wait two ITR expiration times (ensure all open RSCs completed).

- Disable the queue by clearing the RXDCTL.ENABLE bit. The X540 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 X540 completes the current buffer write. If the packet spreads over more than one data buffer, all subsequent buffers are not written.
- The X540 clears the RXDCTL.ENABLE bit only after all pending memory accesses to the descriptor ring are done. The driver should poll this bit before releasing the memory allocated to this queue.
- Once the RXDCTL.ENABLE bit is cleared the driver should wait an additional amount of time
(~100 µs) before releasing the memory allocated to this queue.
- Software can re-configure the Rx filters back to the original setting.

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

Flushing the Packet Buffers

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 X540 provides a mechanism for software to identify when the packet buffers were drained of such stale packets. The RXMEMWRAP register contains a set of counters (one per-packet buffer) that increments each time a buffer is overtaken by the tail pointer. Software must read a counter repeatedly until its count is incremented at least by two, to insure that the buffer made at least one complete wrap-around. Software should also check the *Empty* bit for the counter. If the bit is set, the buffer is empty and there is no further need to sample the buffer counter.

4.6.7.2 RSC Enablement

RSC enablement as well and 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.

Global Setting

- Enable global CRC stripping via the HLREG0 register (hardware default setting).
- Software 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.
- NFS packet handling:
  - NFS header filtering should be disabled if NFS packets coalescing are required (at the TCP layer). The RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits should be set to 1b. Furthermore, the *PSR_type1* bit in the PSRTYPE[n] registers (header split on NFS) must be turned off in all queues.

— Both RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits should be cleared to 0b if NFS coalescing is not required. The *PSR_type1* bit can be set (per queue) according to the required header split.

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.
- 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), then the *RSC Delay* field might need to be increased.
  - Map the relevant Rx queues to an interrupt by setting the relevant IVAR registers.

4.6.8 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 DCA parameters via DCA_TXCTRL.
- Refer to the TIPG description in Section 3.6.6 for more details.
- Set RTTDCS.ARBDIS to 1b.
- 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 0b.

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. Program the TXDCTL register with the desired TX descriptor write-back policy (see the recommended values in the register description).
5. If needed, set TDWBAL/TWDBAH to enable head write back.
6. Enable the transmit path by setting the DMATXCTL.TE bit.

7. Enable the queue using the TXDCTL.ENABLE bit. Poll the TXDCTL register until the *Enable* bit is set.

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

4.6.8.1 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 in order 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.

- The FCoE DDP context table should be initialized clearing the FCBUFF.Valid bit and the FCFLT.Valid bit of all contexts.
- EType Queue Filter — ETQF[n]: Select a filter by setting the *FCoE* bit. The *EType* field should be set to 0x8906 (FCoE Ethernet Type). If FCoE traffic is expected on multiple VLAN priorities then multiple ETQF filters might be required.
- 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.

- Multiple receive queues can be enabled by setting the FCRECTL.ENA bit and programming the FCRETA[n] registers.
- Low Latency Interrupts (LLIs) for critical FCoE frames can be enabled by setting the FCRXCTRL.FCOELLI bit.
- 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.
- Follow the rules described in Section 7.13.2.1 and Section 7.13.3.1 for Tx and Rx cross-functionality requirements. These sections include requirements on Ethernet CRC and padding handling, MACsec offload, legacy Rx buffers, etc.
- Software is not expected to access the EOF and SOF setting. If there is some error in the implementation these settings might need to be modified. If so, software must program the REOFF and TEOFF registers by the same values. This same rule applies to the RSOFF and TSOFF registers.

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 methods — MAC/VLAN filtering for pool selection and either DCB or RSS for the queue in pool selection. MRQC.Multiple Receive Queues Enable = 1000b, 1010b, 1011b, 1100b, or 1101b.
- DCB should be initiated as described in Section 4.6.11. In RSS mode, the RSS key (RSSRK) and redirection table (RETA) should be programmed. Note that the redirection table is common to all pools and only indicates the queue inside the pool to use once the pool is chosen. Each pool can decide if it uses DCB.
- 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.
- 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 and VLAN anti-spoof features via the relevant bit in the PFVFSPOOF.MACAS and PFVFSPOOF.VLANAS registers, 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 didn't 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 untagged 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 or MACsec decrypting.
- 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 in order 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), setup of the MACsec channel, and setup of the DCB parameters.

In IOV mode, VMDq + RSS mode is not available. RSS mode might be used, but this assumes all the VMs uses the same key, RSS hash algorithms, and redirection table, which is currently not supported by any VMM vendor.

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 X540 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 X540 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 X540 for the duration of the exit time. Given 64 KB Rx packet buffers per TC in the X540 (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

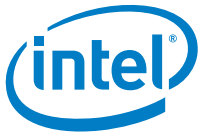
Only the following configuration modes are allowed.

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 the packet buffer size rules in Section 3.6.5.3.5 for a non-symmetrical sizing.
     TXPBSIZE[0-7].SIZE=0x14
     but non-symmetrical sizing is also allowed (see rules in the TXPBSIZE register)
     TXPBTHRESH.THRESH[0-7]=TXPBSIZE[0-7].SIZE — Max expected Tx packet length in this TC
   - 4 TCs mode — Packet buffer size and threshold, typically

     RXPBSIZE[0-3].SIZE=0x60, 0x50, or 0x40, depending on the size of the flow director table. Refer to the packet buffer size rules in Section 3.6.5.3.5 for a non-symmetrical sizing.
     RXPBSIZE[[4-7].SIZE=0x0
     TXPBSIZE[0-3].SIZE=0x28, TXPBSIZE[4-7].SIZE=0x0
     but non-symmetrical sizing among TCs[0-3] is also allowed (see rules in TXPBSIZE register)
     TXPBTHRESH.THRESH[0-3]=TXPBSIZE[0-3].SIZE — Maximum expected Tx packet length in this TC
     TXPBTHRESH.THRESH[4-7]=0x0
   - Multiple Receive and Transmit Queue Control (MRQC and MTQC)
     - Set MRQC.MRQE to 1xxxb, with the 3 LS bits set according to the number of VFs, TCs, and RSS mode as listed in the MRQC register section.
     - Set both *DCB_ena* and *VT_Ena* bits in the MTQC register.
     - Set MTQC.NUM_TC_OR_Q according to the number of TCs/VFs enabled as described in the MTQC register section.



— Set the PFVTCTL.VT_Ena bit (as the MRQC.VT_Ena bit) Queue Drop Enable (PFQDE): In SR-IO 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, in order 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.

— 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 the Flow Control Registers section 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, and BPBFSM=0b

— Tx Packet Plane Control and Status (RTTPCS): TPPAC=1b, TPRM=1b, ARBD=0x004

— 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

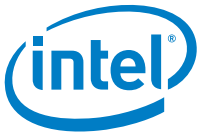
— Clear the PFVTCTL.VT_Ena bit (as the MRQC.VT_Ena bit)

- 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, and BPBFSM=0b
  - — Tx Packet Plane Control and Status (RTTPCS): TPPAC=0b, TPRM=0b, ARBD=0x224
  - — Rx Packet Plane Control and Status (RTRPCS): RAC=0b, RRM=0b

4.6.11.3.4 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_Enabit (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, and 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 then 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 TC's 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. TC's 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 TC's. For example, starting from priority 7 and down; the other TC's 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.

Example 4-10 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 4-11 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	

Table 4-11 Bandwidth Share Example

TCs and VMs		Bandwidth Share %	Note
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 4-12 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 (such as rate-limiting VF traffic). Setting a rate limiter on Tx queue N to a TargetRate requires the following settings:

— 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 RateFactor = LinkSpeed / TargetRate, where LinkSpeed 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

- TargetRate = 4 Gb/s
- LinkSpeed = 10 Gb/s

  RateFactor = 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 (both MACsec and IPsec), and the content of the SA tables must be cleared by software.

Security offload cannot be enabled when either the internal security fuses are blown (set to zero) or the SEC_EN pin is set to 0b. In this case both IPsec and MACsec are 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 1b 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
- LSECTXCTRL bits 1:0 are cleared to 00b and read as 00b
- LSECRXCTRL bits 3:2 are cleared to 00b and read as 00b

Security offload can be used when the security offload internal fuse is enabled and the SEC_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 Enablement 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 or MACsec 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.buff_af_thresh with the value of 0x15.
4. Enable SA lookup:
   a. For IPsec, set the IPSTXIDX.IPS_TX_EN and the IPSRXIDX.IPS_RX_EN bits.
   b. For MACsec, set the enable bits in the LSECTXCTRL and LSECRXCTRL registers.
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:
   a. For IPsec, clear the IPSTXIDX.IPS_TX_EN and the IPSRXIDX.IPS_RX_EN bits.
   b. For MACsec, clear the enable bits in the LSECTXCTRL and LSECRXCTRL registers.
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.buff_af_thresh 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 X540'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.

In order 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.

In order to support this mode, the X540 provides the Alternate Ethernet MAC Address structure in the NVM to store the original MAC addresses. This structure is described in Section 6.3.7.

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.

4.6.13.1 LAN MAC Address Restore

The X540 restores the LAN MAC addresses stored in the Alternate Ethernet MAC Address structure to the regular LAN MAC address location (see Section 6.4.6.2) 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), the regular MAC address is backed up in the Alternate MAC structure.

If the address is modified, the RAH/RAL[0] registers should be set to contain the new address so that APM wake up operates with the correct address.

4.6.13.2 SAN MAC Address Restore

The X540 restores the SAN MAC addresses (word 0x28 - see Section 6.3.4.5) at power on reset by copying the values from the Alternate SAN MAC Address structure (word 0x27; Section 6.3.5.1) 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 6.3.4.5).

3. The addresses stored in the Alternate SAN MAC Address structure (word 0x27 - see Section 6.3.4.4) 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).

If the SAN and LAN Factory MAC addresses where restored by the internal firmware, the *FWSM.Factory MAC address restored* bit is set. This bit is common to all ports.

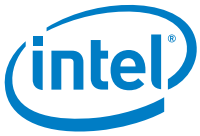
The X540 supports replacement of the MAC address with an alternate MAC address via the NC-SI interface using the Update System MAC address Intel OEM Command or via a BIOS CLP interface as described in the BIOS CLP document.

4.7 Access to Shared Resources

Part of the resources in the X540 are shared between several software entities — namely the drivers of the two ports and internal firmware. In order 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 11.7.5 in order to acquire ownership of this resource, and use the *Releasing a Shared Resource by Software* flow described there in order to relinquish ownership of this resource.

The shared resources are:

- NVM
- PHY 0 and PHY 1 registers
- MAC (LAN controller) shared registers



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

5.0 Power Management and Delivery

This section defines how power management is implemented in the X540.

5.1 Power Targets and Power Delivery

See Section 12.4.1 for the current consumption and Section 12.3.1 for the power supply specification.

5.2 Power Management

5.2.1 Introduction to the X540 Power States

The X540 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 X540 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.

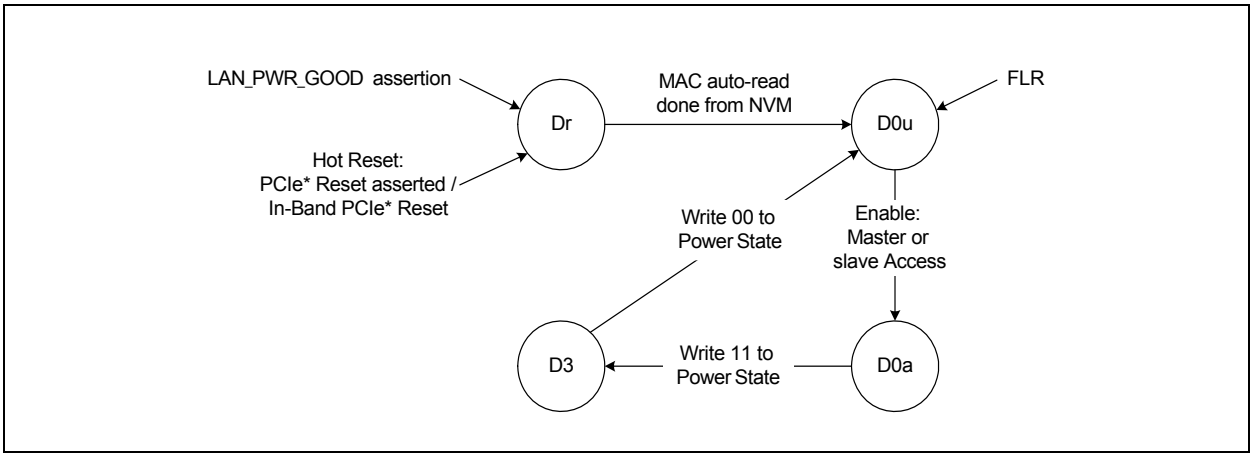


Figure 5-1 Power Management State Diagram

5.2.2 Auxiliary Power Usage

The X540 uses the *AUX_PWR* indication that auxiliary power is available to the controller, and therefore advertises D3cold wake up support. The amount of power required for the function, which includes the entire Network Interface Card (NIC), is advertised in the Power Management Data register, which is loaded from the NVM.

If 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 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 X540.

The X540 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 X540 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 X540 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 Interconnects Power Management

This section described the power reduction techniques employed by the X540 main interconnects.

5.2.3.1 PCIe Link Power Management

The PCIe link state follows the power management state of the device. Since the X540 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 X540 supports all PCIe power management link states:

- L0 state is used in D0u and D0a states.
- The L0s state is used in D0a and D0u states each time link conditions apply.

- The L1 state is used in D0a and D0u states each time link conditions apply, as well as in the D3 state.
- The L2 state is used in the Dr state following a transition from a D3 state if PCI-PM PME is enabled.
- The L3 state is used in the Dr state following power up, on transition from D0a and also if PME is not enabled in other Dr transitions.

The X540 support for active state link power management is reported via the PCIe Active State Link PM Support register loaded from the NVM.

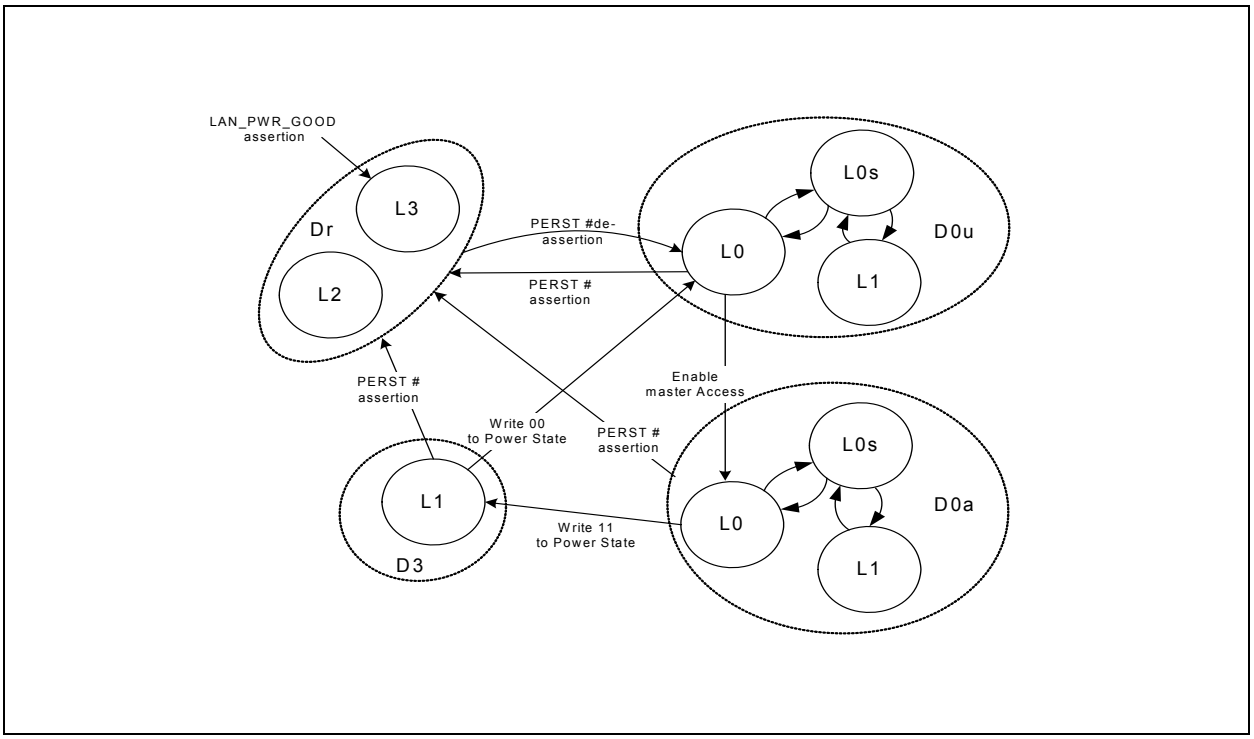


Figure 5-2 Link Power Management State Diagram

While in L0 state, the X540 transitions the transmit lane(s) into L0s state once the idle conditions are met for a period of time defined as follows.

L0s configuration fields are:

- L0s enable — The default value of the *Active State Link PM Control* field in the PCIe Link Control register is set to 00b (both L0s and L1 disabled). System software can later write a different value into the Link Control register. The default value is loaded on any reset of the PCI configuration registers.
- L0s exit latency (as published in the *L0s Exit Latency* field of the Link Capabilities register) is loaded from NVM. Separate values are loaded when the X540 shares the same reference PCIe clock with its partner across the link, and when the X540 uses a different reference clock than its partner across the link. The X540 reports whether it uses the slot clock configuration through the PCIe *Slot Clock Configuration* bit loaded from the *Slot_Clock_Cfg* NVM bit.

- L0s acceptable latency (as published in the *Endpoint L0s Acceptable Latency* field of the Device Capabilities register) is 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)
- PCIe PLL Gate Disable — Controls PCIe PLL gating while in L1 or L2 states
- L1_Act_Ext_Latency — Defines L1 active exit latency
- L1_Act_Acc_Latency — Defines L1 active acceptable exit latency

5.2.3.2 Network Interfaces Power Management

The X540 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 X540 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 (*MNG Enable* bit in the NVM is cleared), then LAN ports are not allocated for manageability.
- If manageability is enabled:
  - Power-up — Following 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 X540 MAC asserts internal signals to notify the PHY that it must either power down as well or re-negotiate to a lower link speed.

5.2.3.2.1 PHY Power-Down State

Each X540 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 doesn't 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 PCIe Control 2 word, and if the port is not used for WoL or for MC traffic, then 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 PCIe Control 2 word 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 entire device is 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.

5.2.3.2.2 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.

5.2.3.2.3 Disable 10GBASE-T and/or 1000MBASE-T Speeds

In order 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).

5.2.3.2.4 Low Power Link Up (LPLU)

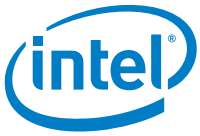
The PHY is aware of power management states. If the PHY is not in a power down state, then PHY behavior regarding several features are different depending on the device or port power state. The internal PHY power state is controlled internally from the MAC.

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 5-1 lists link speed as function of power management state, link speed control, and GbE speed enabling.

Table 5-1 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 (normal operation).
				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.



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	
Non-D0	0b	X	X	0b	0b	PHY advertises 10 GbE, 1 GbE, 100 Mb/s (normal operation).
				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 goes through LPLU procedure, advertising 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 goes through LPLU procedure, advertising 100 Mb/s only.
		1b	1b	X	X	PHY goes through LPLU procedure, advertising 100 Mb/s only.
		0b	1b	X	X	N/A (non supported setting)

X means Don't Care
LPLU bit in NVM - NVM Control Word 3, bit 3, shared by both ports
Disable 10 GbE in LPLU mode - NVM Control Word 3, bits 6 and 8, one bit per port
Disable 1 GbE in LPLU mode - NVM Control Word 3, bits 5 and 7, one bit per port
Disable 10 GbE in any state - Refers to PHY register 7.20: [C]
Disable 1 GbE in any state - Refers to PHY register 7.C400 [F]

Notes: The 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 5-1, when in non-D0 state, are not relevant as the PHY port is disabled.

5.2.3.2.4.1 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 5-1. 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.

5.2.3.2.4.2 Link Speed Change vs. Power Mode

Normal speed negotiation drives to establish a link at the Highest Common Denominator (HCD) link speed. The X540 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 X540 initiates auto negotiation without direct 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 as the manageability session would 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 power management state is enabled by default on LAN Power 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 Power Good (such as the X540 does not revert to the hardware default value on PE_RST#, PCIe reset, or any other reset but LAN Power 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 on the sideband interface (see Section 11.7.1.11.1). 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 on de-assertion of 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 5-3 shows the X540 behavior when entering low power mode:

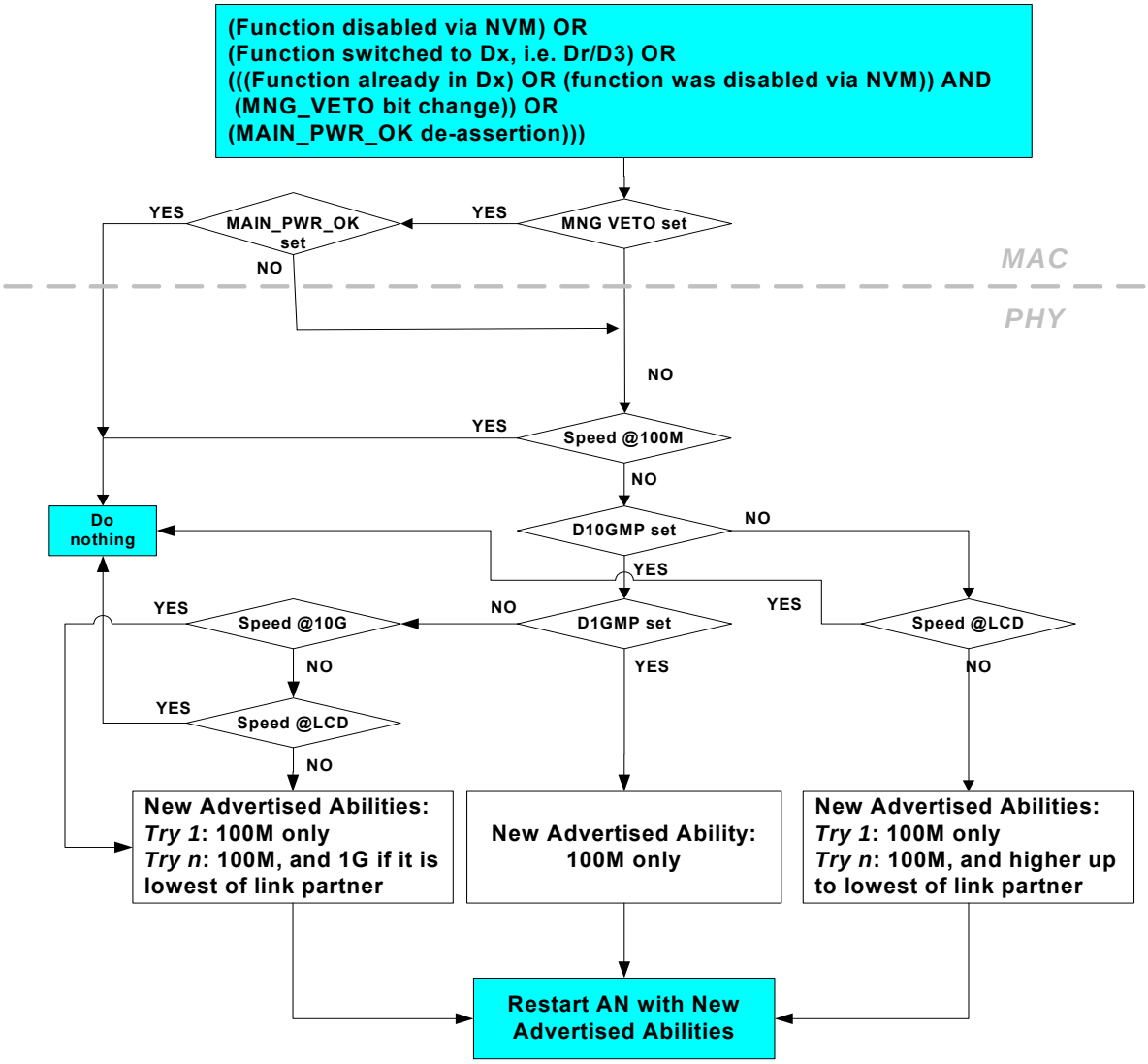


Figure 5-3 Link Speed Change when Entering Power Down Mode

Figure 5-4 describes the X540 behavior when going to power-up mode.

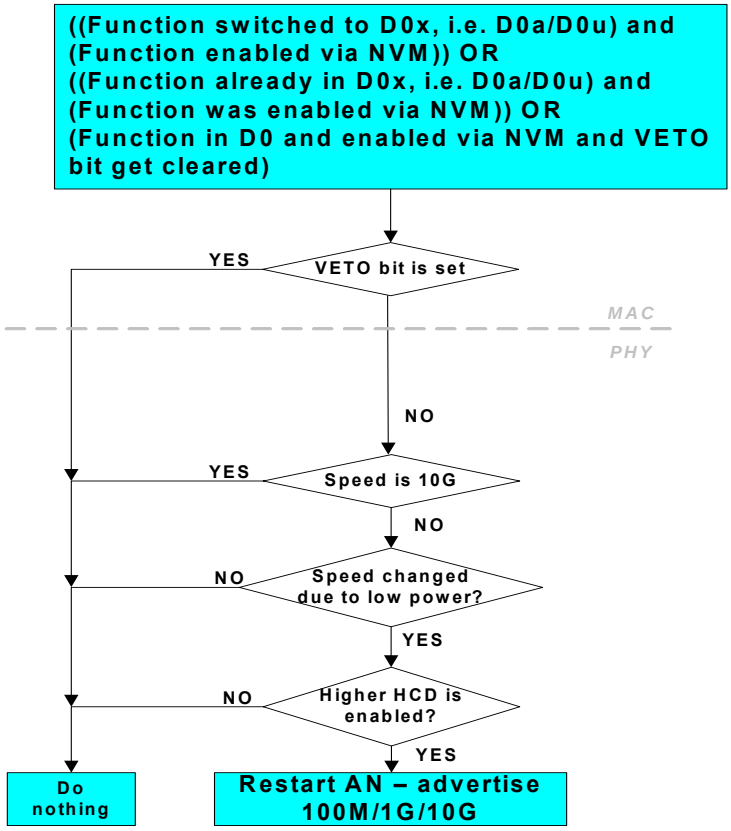


Figure 5-4 Link Speed Change When Entering Power-Up Mode

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, for which 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 5.2.3.2.3, 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.

5.2.3.3 MCTP over SMBus Power Management

MCTP over SMBus is needed only while the system is in S0 mode. In Sx mode, it is expected that any NIC which is not used for WoL will use a minimal power, like if it was configured in no manageability mode.

Therefore, MCTP over SMBus is not active while in Dr State.

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 X540 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 X540 requires that software perform a full re-initialization of the function including its PCI configuration space.

5.2.4.2 D0active State

Once memory space is enabled, the X540 enters an active state. It can transmit and receive packets if properly configured by the 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 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 X540 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 transitioning to D3 state, the X540 clears the *Memory Access Enable* and *I/O Access Enable* bits of the PCI Command register, which disables memory access decode. In D3 the X540 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 device driver disables scheduling of further tasks to the X540; 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 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 X540 brings its PCIe link into the L1 link state. As part of the transition into L1 state, the X540 suspends scheduling of new TLPs and waits for the completion of all previous TLPs it has sent. The X540 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 in order to reduce power consumption, whenever entering D3 state, the device driver might disable one of the LAN ports (LAN disable) and/or the PHY(s) will auto-negotiate network link(s) to a lower speed (if supported by the network interface). See Section 5.2.3.2 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 device must not issue master accesses for this function. Due to the full-duplex nature of PCIe, and the pipelined design in the X540, 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 driver:

- *PCIe Master Disable* bit in the Device Control Register (CTRL) register — When the *PCIe Master Disable* bit is set, the X540 blocks new master requests by this function. The X540 then proceeds to issue any pending requests by this function. This bit is cleared on master reset (LAN Power Good all the way to software reset) to allow master accesses.
- *PCIe Master Enable Status* bits in the Device Status register — Cleared by the X540 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 X540 clears the *PCIe Master Enable Status* bit:
- Master requests by the transmit and receive engines
- All pending completions to the X540 are received.

Note: The device driver disables any reception to the Rx queues as described in Section 4.6.7.1. Then the device driver sets the *PCIe Master Disable* bit when notified of a pending master disable (or D3 entry). The X540 then blocks new requests and proceeds to issue any pending requests by this function. The driver then reads the change made in *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 driver might time out if the PCIe *Master Enable Status* bit is not cleared within a given time. Examples for cases that the X540 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 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 might need to clear internal buffers by setting GCR_EXT.Buffers_Clear_Func bit for 20 µs. To support internal buffers, clearing the HLREG0.LPBK bit should be set during this period, and RXCTRL.RXEN should be cleared. Finally, the driver should initiate two consecutive software resets with a larger delay than 1 µs between the two of them.

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 X540 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 X540 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 device enters Dr state on power-up, the NVM is read. If the NVM contents determine that the conditions to enter Dr Disable are met, the device then enters this mode (assuming that PCIe PE_RST_N is still asserted).

Exit from Dr disable is through de-assertion of PCIe PE_RST_N.

If Dr disable mode is entered from D3 state, the platform might remove X540 power. If the platform removes X540 power, it must remove all power rails from the device if it wishes to use this capability. Exit from this state is through power-up cycle to the X540.

5.2.4.4.2 Entry to Dr State

Dr-entry on platform power-up is as follows:

- Assertion of the internal power detection circuit (LAN_PWR_GOOD). Device power is kept to a minimum by keeping the PHYs in low power.
- The NVM is then read and determines device configuration.
- If the *APM Enable* bit in the NVM's Control Word 3 is set then APM wake up is enabled (for each port independently).
- If the *MNG Enable* bit in the NVM word is set, pass-through manageability is not enabled.
- Each of the LAN ports can be enabled if required for WoL or manageability. See Section 5.2.3.2 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 5.2.3.2.4).
- The PCIe link is not enabled in Dr state following system power up (since PE_RST_N is asserted).

Entry to Dr state from D0a state is through assertion of the PE_RST_N signal. An ACPI transition to the G2/S5 state is reflected in a device transition from D0a to Dr state. The transition can be orderly (such as user selected a shut down operating system option), in which case the device driver might have a chance to intervene. Or, it might be an emergency transition (like power button override), in which case, the 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 5.2.3.2.4).

Transition from D3 state to Dr state is done by assertion of PE_RST_N signal. Prior to that, the system initiates a transition of the PCIe link from L1 state to either the L2 or L3 state (assuming all functions were already in D3 state). The link enters L2 state if PCI-PM PME is enabled.

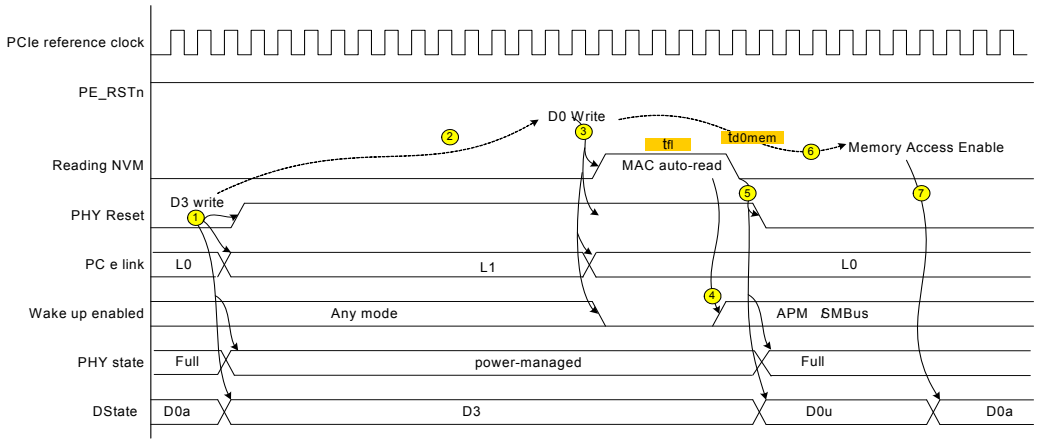
5.2.5 Timing of Power-State Transitions

The following sections give detailed timing for the state transitions. In the diagrams the dotted connecting lines represent the X540 requirements, while the solid connecting lines represent the X540 guarantees.

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.

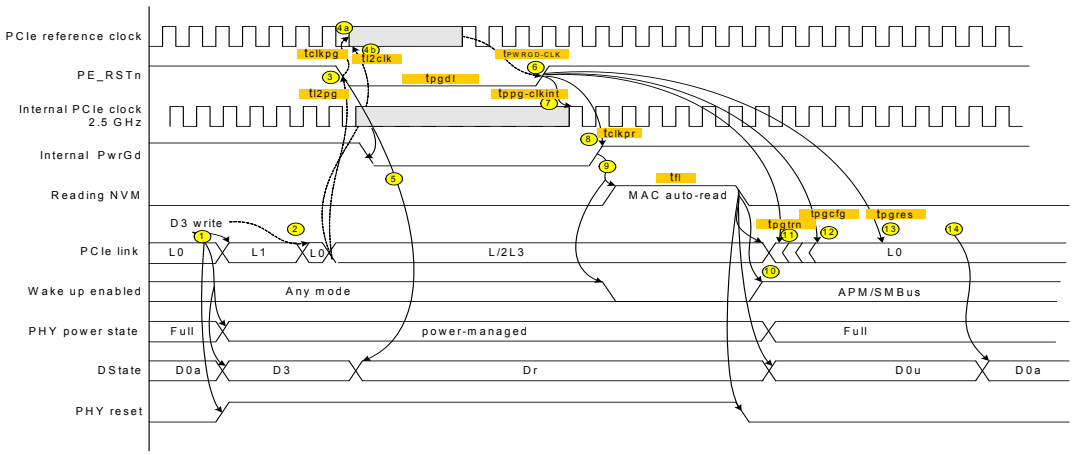
1. ACPI 2.0 specifies that OSPM does not disable wake events before setting the *SLP_EN* bit when entering the S5 sleeping state. This provides support for remote management initiatives by enabling Remote Power On (RPO) capability. This is a change from ACPI 1.0 behavior.

5.2.5.1 Transition From D0a To D3 and Back Without PE_RST_N



Note	
1	Writing 11b to the *Power State* field of the PMCSR transitions the X540 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 X540 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 X540 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 X540 from D0u to D0 state.

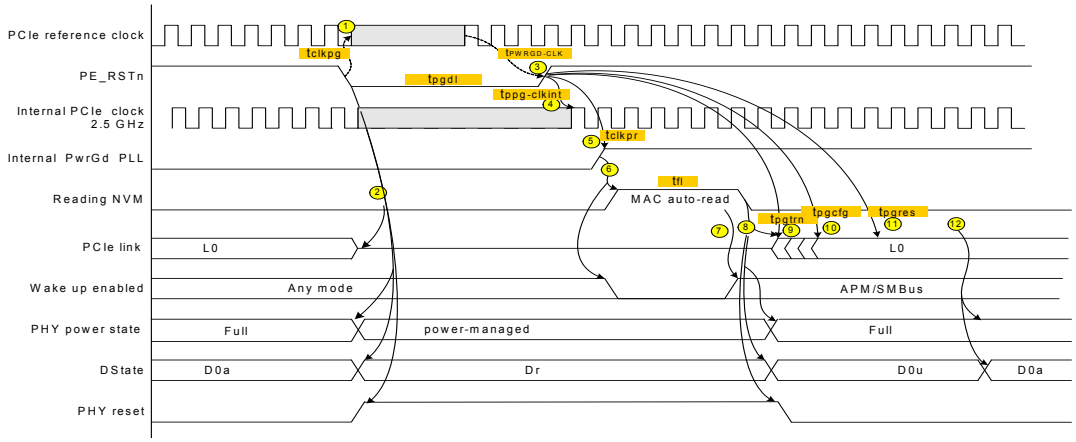
5.2.5.2 Transition From D0a To D3 And Back With PE_RST_N



Note	
1	Writing 11b to the *Power State* field of the PMCSR transitions the X540 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 X540 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.

Note	
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 tpgres 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



Note	
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 X540 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	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.
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.

Note	
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 X540 requires the following start-up and power-state transitions.

Table 5-2 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.
td0mem	Device programmed from D3h to D0 state to next device access	10 ms		According to PCI power management specification.
tl2pg	L2 link transition to PE_RST_N assertion	0 ns		According to PCIe specification.
tl2clk	L2 link transition to removal of PCIe reference clock	100 ns		According to PCIe specification.
tclkpg	PE_RST_N assertion to removal of PCIe reference clock	0 ns		According to PCIe specification.
tpgdl	PE_RST_N assertion time	100 µs		According to PCIe specification.

5.2.5.5 Timing Guarantees

The X540 guarantees the following start-up and power state transition related timing parameters.

Table 5-3 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	37.5 ms	63.6 ms	
t_{fl}	NVM read duration		3 ms	
$t_{ppg\text{-}clkint}$	PCIe PE_RST_N to internal PLL lock		50 µs	
t_{clkpr}	Internal PCIe PWGD from external PCIe PE_RST_N		50 µs	
tpgtrn	PCIe PE_RST_N to start of link training		20 ms	According to PCIe specification.
tpgres	External PE_RST_N to response to first configuration cycle		1 second	According to the PCIe specification: Root-complex or system software must allow at least 1 second after reset of the device before it can determine that a device that fails to return a successful completion status is a broken device.

5.3 Wake Up

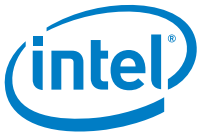
5.3.1 Advanced Power Management Wake Up

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 special signal that ran across a cable to a defined connector on the motherboard. The NIC would assert the signal for approximately 50 ms to signal a wake up. The X540 uses (if configured to) an in-band PM_PME message for this.

At power up, the X540 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 X540 checks all incoming packets for Magic Packets. See Section 18.1.3 for a definition of Magic Packets.

Once the X540 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 WAKE# signal first to resume power and clock to the PCIe interface).
- Sets the Magic Packet *Received* bit in the Wake up Status (WUS) register.

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.

5.3.2 ACPI Power Management Wake Up

The X540 supports ACPI power management based wake up. It can generate system wake-up events from three sources, regardless to the power state:

- Reception of a Magic Packet.
- Reception of a network wake up packet.
- Detection of a link change of state.

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 driver clears all pending wake-up status in the WUS register by writing 1b to all the status bits.
- The 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_FILTER). 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 X540 wakes the system the 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 X540 into low-power mode, and once entering D0, OS disables ACPI wake up.

Once wake up is enabled, the X540 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 X540:

- Sets the *PME_Status* bit in the PMCSR
- If the *PME_En* bit in the PMCSR is set, asserts PE_WAKE_N.
- Sets one or more of the *Received* bits in the WUS register. Note that the X540 sets more than one bit if a packet matches more than one filter.)

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.

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.

5.3.3 Wake-Up Packets

The X540 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.

Note: When VLAN filtering is enabled, a receive packet that passes any of the wake-up filters (that are checked after the L2 filters) causes a wake-up event only if it also passes the VLAN filtering.

5.3.3.1 Pre-Defined Filters

The following packets are supported by the X540'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.

5.3.3.1.1 Directed Exact Packet

The X540 generates a wake-up event upon reception of 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).

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination address		Compare	Match any pre-programmed address

5.3.3.1.2 Directed Multicast Packet

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 then the X540 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).

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination address		Compare	See previous paragraph.

5.3.3.1.3 Broadcast

If the *Broadcast Wake Up Enable* bit in the WUFC register (WUFC.BC) is set the X540 generates a wake-up event when it receives a broadcast packet.

Magic Packet

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination address	FF*6	Compare	

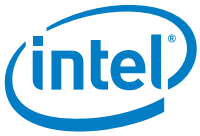
A Magic Packet's destination address must match the address filtering enabled in the configuration registers with the exception that broadcast packets are considered to match even if the *Broadcast Accept* bit of the Receive Control register (FCTRL.BAM) is 0b. If *APM Wakeup* is enabled in the NVM, the X540 starts up with the Receive Address Register 0 (RAH0, RAL0) loaded from the NVM. This is to permit the X540 to accept packets with the matching IEEE address before the driver comes up.

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)

5.3.3.1.4 ARP / IPv4 Request Packet

The X540 supports reception of 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 X540 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



5.3.3.1.5 Directed IPv4 Packet

The X540 supports reception of 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 X540 also handles directed IPv4 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	0x0800	Compare	IP
14+S+D	1	Version/ HDR length	0x4X	Compare	Check IPv4
15+S+D	1	Type of Service	-	Ignore	
16+S+D	2	Packet Length	-	Ignore	
18+S+D	2	Identification	-	Ignore	
20+S+D	2	Fragment Info	-	Ignore	
22+S+D	1	Time to live	-	Ignore	
23+S+D	1	Protocol	-	Ignore	
24+S+D	2	Header Checksum	-	Ignore	
26+S+D	4	Source IP Address	-	Ignore	
30+S+D	4	Destination IP Address	IP4AT	Compare	Can match any of 4 values in IP4AT

5.3.3.1.6 Directed IPv6 Packet

The X540 supports reception of 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 X540 also handles directed IPv6 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	0x0800	Compare	IP
14+S+D	1	Version/ Priority	0x6X	Compare	Check IPv6
15+S+D	3	Flow Label	-	Ignore	
18+S+D	2	Payload Length	-	Ignore	
20+S+D	1	Next Header	-	Ignore	
21+S+D	1	Hop Limit	-	Ignore	
22+S+D	16	Source IP Address	-	Ignore	
38+S+D	16	Destination IP Address	IP6AT	Compare	Match value in IP6AT

5.3.3.2 Flexible Filter

The X540 supports a total of six host flexible filters. Each filter can be configured to recognize any arbitrary pattern within the first 128 byte 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. The 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 — 0x09800 — 0x098FF
- Filter 5 — 0x09900 — 0x099FF

The following table depicts 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 doesn't match the byte programmed in the Flexible Host Filter Table (FHFT_FILTER) then 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 X540 (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.

The following packets are listed for reference purposes only. The flexible filter could be used to filter these packets.

5.3.3.2.1 IPX Diagnostic Responder Request Packet

An IPX diagnostic responder request packet must contain a valid Ethernet MAC address, a Protocol Type of 0x8137, and an IPX diagnostic socket of 0x0456. It can include LLC/SNAP headers and VLAN tags. Since filtering this packet relies on the flexible filters, which use offsets specified by the operating system directly, the operating system must account for the extra offset LLC/SNAP headers and VLAN tags.

Offset	# of bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	
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	0x8137	Compare	IPX
14+S+D	16	Some IPX Stuff	-	Ignore	
30+S+D	2	IPX Diagnostic Socket	0x0456	Compare	

5.3.3.2.2 Directed IPX Packet

A valid directed IPX packet contains the station's Ethernet MAC address, a Protocol Type of 0x8137, and an IPX node address that equals to the station's Ethernet MAC address. It can include LLC/SNAP headers and VLAN tags. Since filtering this packet relies on the flexible filters, which use offsets specified by the operating system directly, the operating system must account for the extra offset LLC/SNAP Headers and VLAN tags.

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	0x8137	Compare	IPX
14+S+D	10	Some IPX Stuff	-	Ignore	
24+S+D	6	IPX Node Address	Receive Address 0	Compare	Must match Receive Address 0

5.3.3.2.3 IPv6 Neighbor Discovery Filter

In IPv6, a neighbor discovery packet is used for address resolution. A flexible filter can be used to check for a neighborhood discovery packet.

5.3.3.3 Wake-Up Packet Storage

Unsupported.

5.3.4 Wake Up and Virtualization

When operating in a virtualized environment, all wake-up capabilities are managed by a single entity (such as the VMM or an IOVM). In an IOV architecture, the physical 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.

6.0 Non-Volatile Memory Map

6.1 NVM General Map

Note: **Intel configures the reserved NVM fields and are not intended to be changed beyond the default image provided by Intel.**

The following table lists the NVM map used by the X540. This table lists common modules for the NVM including: hardware pointers, software and firmware. Blocks are detailed in the following sections. All addresses and pointers in this table are absolute in word units.

Word Address	Used By	Field Name	LAN 0 / 1	End User
0x00	HW	NVM Control Word 1 — Section 6.4.2.1	Shared Logic	Yes
0x01	HW	NVM Control Word 2 — Section 6.4.2.2	Shared Logic	Yes
0x02	HW	Hardware Reserved	Shared Logic	No
0x03	HW	PCIe Analog Configuration Module Pointer — Section 6.4.3	Shared Logic	No
0x04	HW	PHY Module Pointer - Section 6.7	Shared Logic	No
0x05	HW	PCIe Expansion/Option ROM Pointer - Section 6.6	SW + Shared Logic	No
0x06	HW	PCIe General Configuration Module Pointer — Section 6.4.4	Shared Logic	No
0x07	HW	PCIe Configuration Space 0 Module Pointer — Section 6.4.5	Function 0	No
0x08	HW	PCIe Configuration Space 1 Module Pointer — Section 6.4.5	Function 1	No
0x09	HW	LAN Core 0 Module Pointer — Section 6.4.6	Port 0	No
0x0A	HW	LAN Core 1 Module Pointer — Section 6.4.6	Port 1	No
0x0B	HW	MAC 0 Module Pointer — Section 6.4.7	Port 0	No
0x0C	HW	MAC 1 Module Pointer — Section 6.4.7	Port 1	No
0x0D	HW	CSR 0 Auto Configuration Module Pointer — Section 6.4.8	Port 0	No
0x0E	HW	CSR 1 Auto Configuration Module Pointer — Section 6.4.8	Port 1	No

Word Address	Used By	Field Name	LAN 0 / 1	End User
0x0F	FW	Firmware Module Pointer — Section 6.5	FW	No
0x10 – 0x14	SW	SW Compatibility Module — Section 6.3.1	SW	No
0x15 - 0x16	SW	PBA Bytes 1...4 — Section 6.3.2	SW	No
0x17	SW+FW	Boot Configuration Start Address — Section 6.3.3	SW+FW	No
0x18	SW+FW	NVM Image Revision - Section 6.3.4.1	SW+FW	No
0x19 – 0x2E	SW	Software Reserved	SW	No
0x2F	OEM	VPD Pointer — Section 6.3.5	Shared Logic	No
0x30 – 0x36	PXE	PXE Configuration Words — Section 6.3.5	SW	No
0x37	SW+FW	Alternate Ethernet MAC Addresses Pointer — Section 6.3.7	SW+FW	No
0x38	HW	NVM Control Word 3 — Section 6.4.2.3	Shared Logic	Yes
0x39 – 0x3D	HW	Hardware Reserved	Reserved	No
0x3E	SW	Free Space Provisioning Segment Pointer - Section 6.2.1	SW	No
0x3F	SW+FW	Software Checksum, Words 0x00 — 0x3F	Shared Logic	No

Note: All pointers refer to word addresses, excepted to pointers to PCIe Expansion/Option ROM, PHY Image, Firmware Module, and Free Space Provisioning Segment which are expressed in 4 KB sector units. Pointers to 4 KB sectors are recognized by a 1b in their most significant bit.

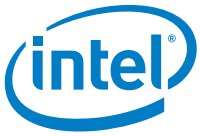
6.2 NVM Modules

NVM modules can vary across platforms and system configurations. Each module family might include several module-children.

6.2.1 NVM Organization

The X540 NVM contains the following four high-level modules:

- **Legacy EEPROM Modules**. These modules are mapped over one of the first two 4 KB sectors of the Flash device, and cannot be extended beyond these sectors. It is made by all the NVM modules used by the MAC hardware (such as PCIe down to MAC blocks, PHYs excluded) or used by the manageability firmware, excepted to firmware patches.
- **Firmware Extension Module**. Must fit within 256 KB and start at a 4 KB boundary. It is mainly destined to firmware patches, but it can also include other firmware sub-



modules. It is pointed by the Firmware Module Pointer located at the NVM word address 0x0F, in either sector 0 or 1. This pointer is used as the base address for the firmware sub-modules that are located in the Firmware Extension Module area. Content of the Firmware Extension Module always starts with the list of firmware sub-modules pointers as listed in Table 6-7.

- **PCIe Expansion/Option ROM**. Must fit within 512 KB and start at a 4 KB boundary.

  It 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 is pointed by the PCIe Expansion/Option ROM Pointer located at the NVM word address 0x05, in either sector 0 or 1.

- **PHY Image**. Must fit within 256 KB and start at a 4 KB boundary.

  It includes a 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.

  The PHY Image Pointer is located at the NVM word address 0x04, in either sector 0 or 1.

  The PHY must validate the PHY image before using it, by checking a *Signature* field enclosed in the image.

The NVM organization must provide a space for the Firmware Extension module to update it via a double image policy. It is referred as a Free Space Provisioning module or segment. It must be large enough to contain this module, at least 256 KB.

It is pointed by the Free Space Provisioning Segment Pointer located at the NVM word address 0x3E. See section 6.2.1.1 for the usage model of this Flash area.

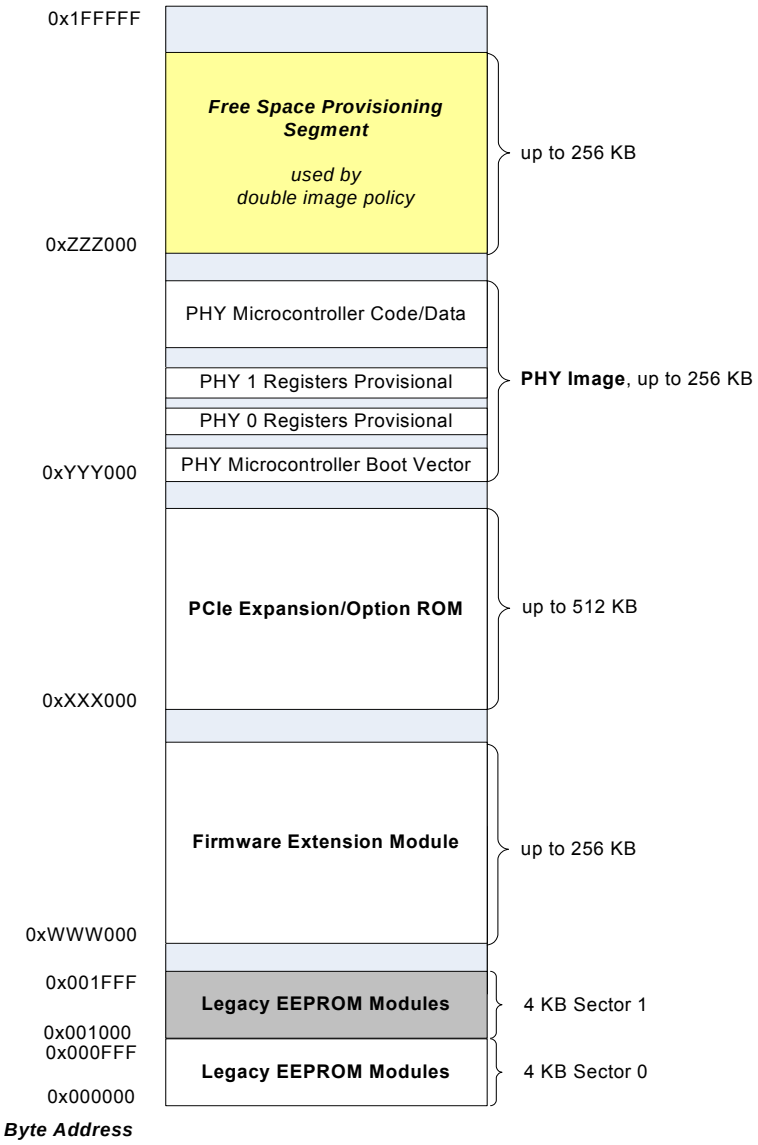


Figure 6-1 Organization of the NVM

Note: Figure 6-1 shows the general NVM organization and not a required order.

6.2.1.1 Double Image Policy - Flow for Updating One of the Big Modules

In order to protect the Flash update procedure from power-down events, a double image policy is required whenever updating the Firmware Extension module. The entire module has to be re-programmed. The updating Software/Firmware should proceed as follows:

1. Take ownership over the NVM via semaphore bits, see Section 11.7.5.
    a. If SW_FW_SYNC.SMBITS[9]:NVM_Update_Started bit is read as clear, then set this bit together with setting NVM semaphore bit. It is used to notify other entities that a long NVM update process has been started, which may take up to several minutes. No write access to the FW Extension is allowed for others during that time, but this module can be read accessed in between update bursts. 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 Firmware Module Pointer.
3. Read the Free Space Provisioning Segment Pointer located at the NVM word address 0x3E.
4. Initiate sector erase instructions of the free space provisioning segment, up to the size needed for replacing the module.
    a. In order to guaranty NVM semaphore ownership time will not exceed the 3 sec / 1 sec timeout, it is recommended to perform at this step no more than four / two 4 KB sector erase operations at once in a burst, releasing semaphore ownership for 10 ms in between. This way, other entities can insert NVM read accesses in between burst without waiting for completion of the whole update process which may take up to minutes.
5. Write the new contents of the module to be replaced at the location pointed by the Free Space Provisioning Segment Pointer via Flash-Mode access.
    a. For the same reason as before, it is recommended to write at this step no more than four / two 4 KB sectors at once in a burst, releasing semaphore ownership for 10 ms in between.
6. Swap between the Free Space Provisioning Segment Pointer and the Firmware Module Pointer. Use the EEPROM-Mode access for this step.
7. Set the *FLUPD* bit in EEC register to require hardware to load the internal shadow RAM into the Flash.
8. Poll the *FLUDONE* bit in EEC register until it is set by hardware.
9. Erase the content of the old module. It is assumed that a firmware image content made by all ones is detected as not valid by the firmware entity.
    a. Same recommendation that in step 4.a. above
10. Poll the *FLUDONE* bit in EEC register until it is set by hardware.
11. Release ownership over the NVM that was taken at step 1, see Section 11.7.5.
    a. Clear SW_FW_SYNC.SMBITS[9]:NVM_Update_Started bit to notify other entities that the long NVM update process completed.

Notes: Pointer swap operation leaves the software checksum word unchanged.

Updating the PCIe Expansion/Option ROM module is affected only by taking/releasing ownership over the NVM. It does not use the double image policy.

6.2.1.2 Flow for Updating One of the Legacy EEPROM Modules

When updating one or several fields from a Legacy EEPROM Module there is a risk that a hardware auto-load event will occur in the middle of the operation (further to a PCIe reset for instance), 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/firmware must use repeatedly the following procedure for each Legacy EEPROM module to be updated:

1. Take ownership over the NVM via semaphore bits, see Section 11.7.5.
2. Invalidate the pointer to the module to be modified by setting it to 0xFFFF via EEPROM-Mode access. 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.
3. Modify the contents of the module via EEPROM-Mode access.
4. Restore the pointers to the modified module(s) via EEPROM-Mode access.
5. Compute and update the software Checksum (word 0x3F).
6. Set the *FLUPD* bit in EEC register to require hardware to load the internal shadow RAM into the Flash.
7. Poll the *FLUDONE* bit in EEC register until it is set by hardware.
8. Release ownership over the NVM that was taken at step 1, see Section 11.7.5.

Note: Refer to Table 6-6 for the NVM hardware module-to-hardware auto-load event mapping.

6.2.1.3 Flow for Updating All the Legacy EEPROM Modules

When updating all the Legacy EEPROM Modules all at once, it is simpler to run the following procedure only once than repeating again the procedure described in Section 6.2.1.2 for each Legacy EEPROM Module:

1. Take ownership over the NVM via semaphore bits, see Section 11.7.5.
2. Read EEC.SEC1VAL to determine which one of the first two 4 KB NVM sector is not valid.
3. Initiate sector erase instruction of the 4 KB sector which is not valid.
4. Write the new contents of the Legacy EEPROM modules into the non-valid 4 KB sector via one of the Flash-Mode accesses. A valid signature field shall be written as well as a valid SW Checksum (word 0x3F).
5. Invalidate the signature field of the valid sector via one of the Flash-mode accesses.

Caution: Initiate a power-up event to synchronize hardware with the new valid 4 KB sector. Otherwise, the new data written into the Flash device is overwritten the next time hardware copies the shadow RAM into the Flash device. Refer to Section 3.4.3. Users must avoid VPD write accesses while in the process of updating all the Legacy EEPROM modules.

6.2.1.4 Flow for Updating the PHY Image Module

1. Take ownership over the NVM via semaphore bits, see Section 11.7.5.
2. Read the PHY FW module pointer at NVM word address 0x4.
3. Enable the Flash for writing.
4. Initiate sector erase instruction of the relevant Flash 4 KB sectors given the starting address of the PHY FW module and the length of the PHY FW module.
5. Write the new PHY FW image to the Flash.
6. Disable the Flash for writing.
7. Verify that the new PHY FW image was written.
8. Update the PHY FW version number at NVM word address 0x19 by the Legacy EEPROM Module Update flow, see Section 6.2.1.2.
9. Read-modify-write SRAMREL register for setting LATCH_IMAGE_VLD bit to 1b.
10. Read-modify-write SRAMREL register for setting LATCH_IMAGE_VLD bit to 0b.
11. Write PHY register bit 1E.C474.0 bit to 0b.
12. Force PHY image reload by setting PHY register bit 1E.C442.0 to 1b.

6.3 Software Sections

6.3.1 Software Compatibility Module — Word Address 0x10-0x14

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.

Note: Unused words are filled with the default value 0xFFFF.

6.3.1.1 Software Compatibility Word 1 — Word Address 0x10

This word is for platform/NIC/LOM specific usage.

Bits	Name	Default	Description
15:12	Reserved	0	Reserved.
11	LOM	0	This bit indicates whether the NVM attached to the X540 contains a dedicated area for storing Option ROM firmware (sometimes called Boot ROM). This bit is used by Option ROM software update utilities and other applications to decide whether or not to allow updating the Option ROM firmware as well as whether or not to allow enabling/disabling the Option ROM by means of the PCI Expansion ROM Base Address Register (BAR). See Section 3.4.4.1 for details on how to enable or disable the PCI Expansion ROM BAR in the X540. 0b = NIC (attached Flash has an area for storing Option ROM firmware). 1b = LOM (attached Flash does not have an area for storing Option ROM firmware).
10	Server	1	Legacy, not currently used. 0b = Client. 1b = Server.
9	Reserved	0	Reserved.
8	OEM/Retail	0	Legacy, not currently used. 1b = OEM 0b = Retail
7:0	Reserved	0	Reserved.

6.3.2 PBA Number Module — Word Address 0x15-0x16

The nine-digit Printed Board Assembly (PBA) number used for Intel manufactured Network Interface Cards (NICs) is stored in the NVM.

Note that 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.

PBA Number	Word 0x15	Word 0x16
G23456-003	FAFA	Pointer to PBA Block

The PBA block is pointed to by word 0x16.

Word Offset	Description	Reserved
0x0	Length in words of the PBA block (default 0x6).	
0x1 ... 0x5	PBA number stored in hexadecimal ASCII values.	

The PBA block contains the complete PBA number including the dash and the first digit of the 3-digit suffix. For example:

PBA Number	Word Offset 0	Word Offset 1	Word Offset 2	Word Offset 3	Word Offset 4	Word Offset 5
G23456-003	0006	4732	3334	3536	2D30	3033

Older PBA numbers starting with (A,B,C,D,E) are stored directly in words 0x15 and 0x16. The dash itself is not stored nor is the first digit of the 3-digit suffix, as it is always 0b for relevant products.

PBA Number	Byte 1	Byte 2	Byte 3	Byte 4
123456-003	12	34	56	03

Note: The PBA module (a length of 12 bytes) must be mapped in the first valid 4 KB sector of the Flash.

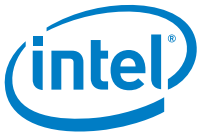
6.3.3 Boot Configuration Block — Word Address 0x17

The Boot Configuration module is located using the Word pointer — *Boot Configuration Address* field in word 0x17. The block length is embedded in the Boot module.

Note:

Configuration Item	Offset (Bytes)	Size in Bytes	Comments
Shared Words			
Boot Signature	0x1:0x0	2	'i', 'S' (0x5369).
Block Size	0x3:0x2	2	Total byte size of the boot configuration block.
Structure Version	0x4	1	Version of this structure. Should be set to 1b.
Reserved	0x5	1	Reserved for future use, should be set to zero.
iSCSI Initiator Name	0x105:0x6	255 + 1	iSCSI initiator name. This field is optional and built by manual input, DHCP host name, or with MAC address.

Shared Words			
iSCSI Configuration Block	0x107:0x106	2	Bits 15:8 (Major) - Combo image major version. Bits 7:0 (Build) - Combo image build number (15:8).
	0x109:0x108	2	Bits 15:8 (Build) - Combo image build number (7:0). Bits 7:0 (Minor) - Combo image minor version.
Reserved	0x127:0x10A	30	Reserved for future use, should be set to zero.
Port 0 Configuration			
iSCSI Flags	0x129:0x128	2	Bit 0x00: Enable DHCP 0b = Use static configurations from this structure. 1b = Overrides configurations retrieved from DHCP. Bit 0x01: Enable DHCP for getting iSCSI target information. 0b = Use static target configuration. 1b = Use DHCP to get target information by the Option 17 Root Path. Bit 0x02 – 0x03: Authentication Type 0x00 = None. 0x01 = One way chap. 0x02 = Mutual chap. Bit 0x04 – 0x05: Ctrl-D setup menu 0x00 = Enabled 0x03 = Disabled, skip Ctrl-D entry Bit 0x06 - 0x07: Reserved Bit 0x08 - 0x09: ARP Retries Retry value Bit 0x0A – 0x0F: ARP Timeout Timeout value for each try
iSCSI Initiator IP	0x12D:0x12A	4	Initiator DHCP flag not set = This field should contain the initiator IP address. Set = This field is ignored.
Subnet Mask	0x131:0x12E	4	Initiator DHCP flag Not set = This field should contain the subnet mask. Set = This field is ignored.
Gateway IP	0x135:0x132	4	Initiator DHCP flag Not set = This field should contain the gateway IP address. Set = If DHCP bit is set this field is ignored.
iSCSI Boot LUN	0x137:0x136	2	Target DHCP flag Not set = iSCSI target LUN number should be specified. Set = This field is ignored.
iSCSI Target IP	0x13B:0x138	4	Target DHCP flag; Not set = IP address of iSCSI target. Set = This field is ignored.
iSCSI Target Port	0x13D:0x13C	2	Target DHCP flag Not set = TCP port used by iSCSI target. Default is 3260. Set = This field is ignored.



Port 0 Configuration			
iSCSI Target Name	0x23D:0x13E	255 + 1	Target DHCP flag Not set = iSCSI target name should be specified. Set = This field is ignored.
CHAP Password	0x24F:0x23E	16 + 2	The minimum CHAP secret must be 12 octets and maximum CHAP secret size is 16. The last 2 bytes are null alignment padding.
CHAP User Name	0x2CF:0x250	127 + 1	The user name must be non-null value and maximum size of user name allowed is 127 characters.
Configuration Item	**Offset (Bytes)**	**Size in Bytes**	**Comments**
VLAN ID	0x2D1:0x2D0	2	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.
Mutual CHAP Password	0x2E3:0x2D2	16 + 2	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.
FCoE Flags	0x2E5:0x2E4	2	Bit 1 used for Disable FCoE Ctrl-D Menu. Default = 0. 0 = FCoE Ctrl-D menu is enabled and user can configure FCoE ports. 1 = FCoE Ctrl-D menu is disabled and user cannot configure FCoE ports. 1 =
Reserved	0x2EB:0x2E6	6	Reserved for future use, should be set to zero.
FCoE Target 0			
Target Worldwide Port Name (WWPN)	0x2F3:0x2EC	8	Byte string of target WWPN
Boot LUN	0x2F5:0x2F4	2	Target LUN
VLAN ID	0x2F7:0x2F6	2	VLAN ID for the Port. Default is 0.
Target Boot Order	0x2F8	1	Target Boot Order. Valid range is 0-4, with 0 meaning no boot order.
Reserved	0x2FB:0x2F9	3	Reserved for future use, should be set to zero.
FCoE Target 1			
Target Worldwide Port Name (WWPN)	0x303:0x2FC	8	Byte string of target WWPN
Boot LUN	0x305:0x304	2	Target LUN
VLAN ID	0x307:0x306	2	VLAN ID for the Port. Default is 0.
Target Boot Order	0x308	1	Target Boot Order. Valid range is 0-4, with 0 meaning no boot order.
Reserved	0x30B:0x309	3	Reserved for future use, should be set to zero.

FCoE Target 2			
Target Worldwide Port Name (WWPN)	0x313:0x30C	8	Byte string of target WWPN
Boot LUN	0x315:0x314	2	Target LUN
VLAN ID	0x317:0x316	2	VLAN ID for the Port. Default is 0.
Target Boot Order	0x318	1	Target Boot Order. Valid range is 0-4, with 0 meaning no boot order.
Reserved	0x31B:0x319	3	Reserved for future use, should be set to zero.
FCoE Target 3			
Target Worldwide Port Name (WWPN)	0x323:0x31C	8	Byte string of target WWPN
Boot LUN	0x325:0x324	2	Target LUN
VLAN ID	0x327:0x326	2	VLAN ID for the Port. Default is 0.
Target Boot Order	0x328	1	Target Boot Order. Valid range is 0-4, with 0 meaning no boot order.
Reserved	0x329	1	Reserved for future use, should be set to zero.
Reserved	0x383:0x32A	90	Reserved for future use, should be set to zero.
Port 1 Configuration			
Same configuration as port 0. Add to each offset 0x25C.			

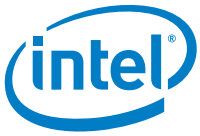
6.3.4 Software Reserved — Words 0x18-0x2E

This area is reserved for software, specific usage reserved for platform/NIC/LOM.

Note: Unused words are filled with the default value 0xFFFF.

6.3.4.1 Software Reserved Word 1 — NVM Image Revision — Word 0x18

Bits	Name	Default	Description
15:12	Major		NVM major version.
11:4	Minor		NVM minor version.
3:0	ID		NVM image ID.



6.3.4.2 Software Reserved Word 2 — PHY FW Revision — Word Address 0x19

Word 0x19 points to the PHY firmware revision.

Note: Default value 0xFFFF means this word is not used.

Bits	Name	Default	Description
15:12	Major Version		Major PHY firmware revision.
11:4	Minor Version		Minor PHY firmware revision.
3:0	Point Release Version		PHY firmware image ID.

6.3.4.3 Software Reserved Word — PXE VLAN Configuration Pointer — Word 0x20

Bits	Name	Default	Description
15:0	PXE VLAN Configuration Pointer	0x0	The pointer contains offset of the first NVM word of the PXE VLAN config block.

Table 6-1 PXE VLAN Configuration Section Summary Table

Word Offset	Word Name	Description
0x0000	VLAN Block Signature	ASCII 'V', 'L'.
0x0001	Version and Size	Contains version and size of structure.
0x0002	Port 0 VLAN Tag	VLAN tag value for the first port of the X540. Contains PCP, CFI and VID fields. A value of 0 means no VLAN is configured for this port.
0x0003	Port 1 VLAN Tag	VLAN tag value for the second port of the X540. Contains PCP, CFI and VID fields. A value of 0 means no VLAN is configured for this port.

Table 6-2 VLAN Block Signature - 0x0000

Bits	Field Name	Default	Description
15:0	VLAN Block Signature	0x4C56	ASCII 'V', 'L'

Table 6-3 Version and Size - 0x0001

Bits	Field Name	Default	Description
15:8	Size		total size in bytes of section
7:0	Version	0x01	Version of this structure. Should be set to 1

Table 6-4 Port 0 VLAN Tag - 0x0002

Bits	Field Name	Default	Description
15:13	Priority (0-7)	0x0	Priority 0-7
12	Reserved	0x0	Always 0
11:0	VLAN ID (1- 4095)	0x0	VLAN ID (1-4095)

Table 6-5 Port 1 VLAN Tag - 0x0003

Bits	Field Name	Default	Description
15:13	Priority (0-7)	0x0	Priority 0-7
12	Reserved	0x0	Always 0
11:0	VLAN ID (1- 4095)	0x0	VLAN ID (1-4095)

6.3.4.4 Software Reserved Word 16 — Word Address 0x27

Word 0x27 points to the Alternate SAN MAC Address block used by FCoE.

Note: Default value 0xFFFF means this word is not used.

Bits	Name	Default	Description
15:0	Alternate SAN MAC Address Pointer	0xFFFF	Pointer to the Alternate SAN MAC Address block. Used by both software and firmware. 0xFFFF indicates an invalid pointer.

6.3.4.4.1 Alternate SAN MAC Address Block

Word 0x27 points to the Alternate SAN MAC Address block used to store the alternate SAN MAC addresses and alternate WWN prefixes. Offsets to the SAN MAC addresses are defined as follows:

Word Offset	Default	Description
0x0	0x0003	Capabilities
0x1 ... 0x3	0xFFFF	Alternate SAN MAC Address 1 (function 0)
0x4 ... 0x6	0xFFFF	Alternate SAN MAC Address 2 (function 1)
0x7	0xFFFF	Alternate WWNN prefix
0x8	0xFFFF	Alternate WWPN prefix

6.3.4.4.2 Capabilities - Offset 0x0

The capabilities block indicates which words are supported. It is primarily for future expansion, if necessary.

Bits	Name	Default	Description
15:2	Reserved	0x0	Reserved
1	Alternate WWN Base	1b	Alternate WWN base address (0x7 and 0x8) are allocated in the alternate SAN MAC address block and can be read or written to.
0	Alternate SAN MAC Address	1b	Alternate SAN MAC Address words (0x1…0x6) are available and can be written to.

6.3.4.5 Software Reserved Word 17 — Word Address 0x28

Word 0x28 points to the Active SAN MAC Address block used for FCoE (SPMA and FPMA) and DCB.

Bits	Name	Default	Description
15:0	SAN MAC Address Pointer	0xFFFF	Pointer to the Active SAN MAC Address block. 0xFFFF indicates an invalid pointer.

6.3.4.5.1 Active SAN MAC Address Block

Word 0x28 points to the Active SAN MAC Address block used for FCoE (SPMA and FPMA) and DCB. Offsets to the MAC addresses are defined in the following table:

Word Offset	Description
0x0 ... 0x2	SAN MAC Address 1 (function 0)
0x3 ... 0x5	SAN MAC Address 2 (function 1)

6.3.4.6 Software Reserved Word 18 — DS_Version — Word Address 0x29

Word 0x29 is for the device starter version.

Bits	Name	Default	Description
15:0	DS_Version	0xFFFF	Device starter version.

6.3.4.7 Software Reserved Word 19 — OEM_Version / ID — Word Address 0x2A

Optional field that enables an OEM to write a version and OEM identifier in the NVM image. Used only for OEM NVM images.

Bits	Name	Default	Description
15:0	OEM Version and OEM ID	0xFFFF	OEM Version and OEM ID. Format defined by OEM.

6.3.4.8 Software Reserved Word 21 — Word Address 0x2C

This word is for platform/NIC/LOM specific capabilities.

Bits	Name	Default	Description
15:4	Reserved	0x0	Reserved
3:2	Wake On LAN Support	11b	This bit indicates to software if the X540 supports Wake on LAN. 00b = Reserved (not supported). 01b = WoL supported on both ports.[1] 10b = WoL supported on port A only. 11b = WoL not supported on either port.
1:0	Reserved	1b	Reserved

1. Reserved bit for the X540 single port configuration.

6.3.4.9 Software Reserved Words 22/23 — Image Unique IDs — Word Addresses 0x2D/0x2E

These words contain a unique 32-bit ID for each image generated by Intel to enable tracking of images and comparison to the original image if testing a customer EEPROM image.

6.3.5 VPD Module Pointer — Word Address 0x2F

The Vital Product Data (VPD) module is located using the Word pointer *VPD Pointer* field in word 0x2F. The block length is embedded in the VPD module. The VPD section size is usually 64 words, and is initialized to 0x0 or 0xFFFF. Customers write their own data in this module. During run time this module is accessible through the VPD capability in the PCI configuration space.

6.3.6 PXE Configuration Words — Word Address 0x30-0x36

Words 0x30 through 0x36 are reserved for configuration and version values used by pre-boot software (PXE/iSCSI boot/ FCoE boot/ UEFI codes).

Word Address	Description	Reserved
0x30	PXE Setup Options PCI Function 0 - Section 6.3.6.1	
0x31	PXE Configuration Customization Options PCI Function 0 - Section 6.3.6.2	
0x32	PXE Version - Section 6.3.6.3	
0x33	Flash Capabilities - Section 6.3.6.4	
0x34	PXE Setup Options PCI Function 1 - Section 6.3.6.5	
0x35	PXE Configuration Customization Options PCI Function 1 - Section 6.3.6.6	
0x36	iSCSI Option ROM Version - Section 6.3.6.7	

6.3.6.1 PXE Setup Options PCI Function 0 — Word Address 0x30

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.

Bits	Name	Default	Description
15:13			Reserved. Must be 0x0.
12:10	FSD		Bits 12-10 control forcing speed and duplex during driver operation. Valid values are: 000b = Auto-negotiate. 001b = Reserved. 010b = 100 Mb/s half duplex. 011b = Not valid (treated as 000b). 100b = Not valid (treated as 000b). 101b = Reserved. 110b = 100 Mb/s full duplex. 111b = 1000 Mb/s full duplex. **Note:** Only applicable for copper-based adapters. Not applicable to 10 GbE. Default value is 000b.
9	LWS		Legacy OS Wakeup Support. (For X540-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 don't 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.
8	DSM		Display Setup Message. If the bit is set to 1b, the "Press Control-S" message is displayed after the title message. Default value is 1b.
7:6	PT		Prompt Time. These bits control how long the "Press Control-S" setup prompt message is displayed, if enabled by DIM. 00b = 2 seconds (default). 01b = 3 seconds. 10b = 5 seconds. 11b = 0 seconds. Note: The Ctrl-S message is not displayed if 0 seconds prompt time is selected.
5	Disable iSCSI Setup Menu	0x0	This controls the iSCSI init message (Ctrl+D menu prompt) when iSCSI is disabled. When iSCSI option ROM is disabled and this bit is set to 1b, the init message is not displayed for the port. When iSCSI option ROM is disabled and this bit is set to 0b, the init message is displayed for the port. When iSCSI option ROM is enabled this bit does not matter, as the init message is displayed for the port.

Bits	Name	Default	Description
4:3	DBS		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

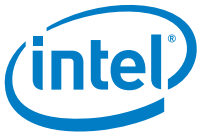
Bits 2:0 are defined as follows:

Bit(s)	Value	Port Status	CLP (Combo) Executes	iSCSI Boot Option ROM CTRL-D Menu	FCoE Boot Option ROM CTRL-D Menu
2:0	0	PXE	PXE	Displays port as PXE. Allows changing to Boot Disabled, iSCSI Primary or Secondary.	Displays port as PXE. Allows changing to Boot Disabled, FCoE enabled.
	1	Boot Disabled	NONE	Displays port as Disabled. Allows changing to iSCSI Primary/Secondary.	Displays port as Disabled. Allows changing to FCoE enabled.
	2	iSCSI Primary	iSCSI	Displays port as iSCSI Primary. Allows changing to Boot Disabled, iSCSI Secondary.	Displays port as iSCSI. Allows changing to Boot Disabled, FCoE enabled.
	3	iSCSI Secondary	iSCSI	Displays port as iSCSI Secondary. Allows changing to Boot Disabled, iSCSI Primary.	Displays port as iSCSI Allows changing to Boot Disabled, FCoE enabled.
	4	FCoE	FCOE	Displays port as FCoE. Allows changing port to Boot Disabled, iSCSI Primary or Secondary.	Displays port as FCoE Allows changing to Boot Disabled.
	7:5	Reserved	Same as disabled.	Same as disabled.	Same as disabled.

6.3.6.2 PXE Configuration Customization Options PCI Function 0 - Word Address 0x31

Word 0x31 of the NVM contains settings that can be programmed by an OEM or network administrator to customize the operation of the software. These settings cannot be changed from within the Control-S setup menu. The lower byte contains settings that would typically be configured by a network administrator using an external utility; these settings generally control which setup menu options are changeable. The upper byte is generally settings that would be used by an OEM to control the operation of the agent in a LOM environment, although there is nothing in the agent to prevent their use on a NIC implementation. The default value for this word is 0x4000.

Bits	Name	Default	Description
15:14	SIG		Signature. Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software.
13	RFU		Reserved. Must be 0b.
12	RFU		Reserved. Must be 0b.
11	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 in case of hardware error conditions such as an invalid NVM checksum or failing to establish link. Default value is 0b.
10:8	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 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 support in the BIOS and assumes the BIOS is not compliant. The user can change the BIOS boot order in the Setup Menu. 010b = Force BBS mode. The agent assumes the BIOS is BBS-compliant, even though it might not be detected as such by the agent's detection code. The user CANNOT change the BIOS boot order in the Setup Menu. 011b = Force PnP Int18 mode. The agent assumes the BIOS allows boot order setup for PnP Expansion ROMs and hooks interrupt 0x18 (to inform the BIOS that the agent is a bootable device) in addition to registering as a BBS IPL device. The user CANNOT change the BIOS boot order in the Setup Menu. 100b = Force PnP Int19 mode. The agent assumes the BIOS allows boot order setup for PnP Expansion ROMs and hook interrupt 0x19 (to inform the BIOS that the agent is a bootable device) in addition to registering as a BBS IPL device. The user CANNOT change the BIOS boot order in the Setup Menu. 101b = Reserved for future use. If specified, is treated as a value of 000b. 110b = Reserved for future use. If specified, is treated as a value of 000b. 111b = Reserved for future use. If specified, is treated as a value of 000b.
7	RFU		Reserved. Must be 0b.
6	RFU		Reserved. Must be 0b.
5	DFU		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.
4	DLWS		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.
3	DBS		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.
2	DPS		Disable Protocol Select. If set to 1b, the user is not allowed to change the boot protocol. Default value is 0b.
1	DTM		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.



Bits	Name	Default	Description
0	DSM		Disable Setup Menu. If this bit is set to 1b, the user is not allowed to invoke the setup menu by pressing Control-S. In this case, the NVM can only be changed via an external program. Default value is 0b.

6.3.6.3 PXE Version — Word Address 0x32

Word 0x32 of the NVM is used to store the version of the PXE that is stored in the Flash image. When the PXE loads, it can check this value to determine if any first-time configuration needs to be performed. PXE then updates this word with its version. Some diagnostic tools also read this word to report the version of the PXE in the Flash.

Bits	Name	Default	Description
15:12	MAJ		PXE Major Version. Default value is 0x0.
11:8	MIN		PXE Minor Version. Default value is 0x0.
7:0	BLD		PXE Build Number. Default value is 0x0.

6.3.6.4 Flash Capabilities — Word Address 0x33

Word 0x33 of the NVM is used to enumerate the boot technologies that have been programmed into the Flash. This is updated by Flash configuration tools and is not updated by option ROMs.

Bits	Name	Default	Description
15:14	SIG		Signature. Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software.
13:6	RFU		Reserved. Must be 0x0.
5	FCOE		FCoE boot code is present if set to 1b.
4	ISCSI		iSCSI boot code is present if set to 1b.
3	EFI		EFI UNDI driver is present if set to 1b.
2	RPL		RPL module is present if set to 1b. Reserved bit for devices.
1	UNDI		PXE UNDI driver is present if set to 1b.
0	BC		PXE Base Code is present if set to 1b.

6.3.6.5 PXE Setup Options PCI Function 1 — Word Address 0x34

This word is the same as word 0x30, but for PCIe function 1 of the device.

6.3.6.6 PXE Configuration Customization Options PCI Function 1 — Word Address 0x35

This word is the same as word 0x31, but for PCIe function 1 of the device.

6.3.6.7 iSCSI Option ROM Version — Word Address 0x36

Word 0x36 of the NVM is used to store the version of iSCSI Option ROM updated as the same format as PXE Version at Word 0x32. The value must be above 0x2000 and the value below (word 0x1FFF = 16 KB NVM size) is reserved. iSCSIUtl, FLAUtil, DMiX update iSCSI Option ROM version if the value is above 0x2000, 0x0000, or 0xFFFF. The value (0x0040 - 0x1FFF) should be kept and not be overwritten.

6.3.7 Alternate Ethernet MAC Address Pointer — Word Address 0x37

This word is used as a pointer to an EEPROM block that contains the space for two MAC addresses. The first three words of the EEPROM block are used to store the MAC address for PCIe Function 0. The second three words of the EEPROM block is used to store the MAC address for PCIe Function 1. Initial and default values in the EEPROM block should be set to 0xFFFF (for both addresses) indicating that no alternate MAC address is present. See Section 4.6.13 for more details.

Note: Word 0x37 must be set to 0xFFFF if alternate MAC addresses are not used. Also, alternate MAC addresses are ignored by hardware and require specific software support for activation.

Word Offset	Description	Reserved
0x0 ... 0x2	Alternate Ethernet MAC Address 1 (function 0)	
0x3 ... 0x5	Alternate Ethernet MAC Address 2 (function 1)	

6.3.8 Checksum Word Calculation (Word 0x3F)

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 4 KB 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 software. Hardware does not calculate nor check its content but rather checks the *Signature* field in NVM Control Word 1.

The checksum word calculation is as follows:

```
#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;
    }
```

6.4 Hardware Sections

Note: This module contains address control words and hardware pointers indicated as HW in the table of Section 6.1. The process of loading this module (or any of it sub-modules) into the device is referred as MAC auto-load process.

6.4.1 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, excepted after LAN_PWR_GOOD or Reset Pin events.

Table 6-6 NVM Section Auto-Read

Description	System Events That Cause Reload							
	Power Up or LAN_PWR_GOOD (Reset Pin)	PCIe Reset or PCIe In-band Reset	D3 to D0 Transition or FLR (per port)	SW Reset (per port)	Link Reset (per port)	FW Reset	Force TCO	PHY Image Reload Comm and via MDIO
NVM Section								
PCIe Analog Configuration	X[1]							
PCIe General Configuration		X						

Table 6-6 NVM Section Auto-Read

Description	System Events That Cause Reload							
	Power Up or LAN_PWR_GOOD (Reset Pin)	PCIe Reset or PCIe In-band Reset	D3 to D0 Transition or FLR (per port)	SW Reset (per port)	Link Reset (per port)	FW Reset	Force TCO	PHY Image Reload Comm and via MDIO
PCIe Function 0/1 Config Space (section for each function)		X	X					
LAN Core and CSRs (for each LAN port)		X	X	X	X		X	
MAC Module (for each LAN port)	X[2]	X[3]	X[3]	X[3]	X		X	
PHY Module (for each LAN port)	X							X
Manageability Firmware Module	X					X[4]		

1. This is the unique module that requires power-up to be reloaded.
2. The module is auto-read here only if manageability or wake up are enabled.
3. The module is auto-read only if the manageability unit is not enabled.
4. Parts of firmware module are not stored in internal shadow RAM and will be retrieved from NVM device.

6.4.2 NVM Init Module

The Init section (NVM Control Word 1, 2, and 3) are read after a LAN_PWR_GOOD reset and PCIe Reset.

6.4.2.1 NVM Control Word 1 — Address 0x00

Bits	Name	Default	Description	Reserved
15:11	Reserved	0	Reserved	
10:8	NVM Size	101b	These bits indicate the Flash size. Mapped to FLA.FL_SIZE (see field definition in the FLA register section).	
7:6	Signature	01b	The *Signature* field indicates to the X540 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.	
5	MNG Enable	1b	Manageability Enable. When set, indicates that the manageability block is enabled. When cleared, the manageability block is disabled (clock gated). Mapped to GRC.MNG_EN	
4	Reserved	0b	Must be cleared to 0b.	
3	Disable Burst	0b	Disable burst write accesses to the Flash. 0b = Burst write access to the Flash is enabled up to 256 bytes. 1b = Write access to the Flash is limited to 1 byte at a time. Mapped to EEC.Disable Burst. **Restricted:** should not be exposed in external documentation.	
3:0	Reserved	0x0	Reserved	

6.4.2.2 NVM Control Word 2 — Address 0x01

Bits	Name	Default	Description	Reserved
15:4	Reserved	0xFFE	Reserved	
3	Deadlock Timeout Enable	1b	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.	
2	Reserved	0b	Reserved	
1	Core clocks gate disable	0b	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.	

Bits	Name	Default	Description	Reserved
0	PCI-E PLL Gate disable	0b	When set, PCIe PLL locks only after LAN_PWR_GOOD. During assertion of PE_RST_N, PLL does not reset. Note that in this case, PLL might not be in a lock state. Once REFCLK is stable, the PLL re-locks. When this bit is cleared (default), PLL is reset upon assertion of PE_RST_N and starts re-lock only after PE_RST_N de-assertion.	

6.4.2.3 NVM Control Word 3 — Address 0x38

Bits	Name	Default	Description	Reserved
15:9	Reserved	0x0	Reserved	
8	D10GMP Port 1[1]	0b	Disable 10 GbE in LPLU for LAN Port 1. When set, LAN port 1 never advertises 10 GbE speed capability when in LPLU state (D3/Dr).	
7	D1GMP Port 1[1]	0b	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.	
6	D10GMP Port 0	0b	Disable 10 GbE in LPLU for LAN Port 0. When set, LAN port 0 never advertises 10 GbE speed capability when in LPLU state (D3/Dr).	
5	D1GMP Port 0	0b	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.	
4	Reserved	0b	Reserved	
3	Enable LPLU	1b	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. See Section 5.2.3.2.4.	
2	Keep_PHY_Link_Up_En	1b	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. See Section 4.3. When cleared, the MMNGC.MNG_VETO bit is meaningless.	
1	APM Enable Port 1[1]	0b	Initial value of advanced power management wake up enable in the General Receive Control register (GRC.APME). Mapped to GRC.APME of port 1. If the LAN PCI disable bit in the NVM is set for *Port 1*, then the APM bit must be cleared.[2]	

Bits	Name	Default	Description	Reserved
0	APM Enable Port 0	0b	Initial value of advanced power management wake up enable in the General Receive Control register (GRC.APME). Mapped to GRC.APME of port 0. If the LAN PCI disable bit in the NVM is set for *Port 0*, then the APM bit must be cleared.[2]	

1. When using the X540 single port configuration, this bit is reserved and should be set to 0b.
2. In regular mode (port swap disabled) Wol of port 0 is mapped to GRC.APE of NC-SI channel 0, port 0.
   In regular mode (port swap disabled) Wol of port 1 is mapped to GRC.APE of NC-SI channel 1, port 1.
   In port swap mode Wol of port 0 is mapped to GRC.APE of NC-SI channel 1, port 0.
   In port swap mode Wol of port 1 is mapped to GRC.APE of NC-SI channel 0, port 1.

6.4.3 PCIe Analog Configuration Module

These sections are loaded only after LAN_PWR_GOOD only. These sections contain the analog default configurations for the X540's PCIe analog parts. Word 0x3 is the pointer for this section (the NVM address, in words).

The structure of this section is listed in the following table.

Word offset	Description	
0x0	Section Length	
0x1	PCIe* Analog Address 1	
0x2	PCIe* Analog Data 1	
0x3	PCIe* Analog Address 2	
0x4	PCIe* Analog Data 2	
….	……	

6.4.3.1 Section Length — Offset 0x00

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.

6.4.3.2 PCIe Analog Address — Offset 0x01, 0x03, 0x05...

Each odd offset word in the PCIe analog section is the register address. The PCIe analog registers are 2 words wide with a 15-bit address width. There are two addressable register spaces for this section: Link/PHY register space and PCI SerDes register space. Bit 14 selects between the two spaces. When set to 1, PCIe SerDes space is selected, otherwise Link/PHY space is selected.

6.4.3.2.1 Link/PHY Address Word (Bit 14 = 0b)

Bits	Name	Default	Description	Reserved
15	Reserved	0b	Reserved	
14	SerDes/Link PHY	0b	0b=Select Link/PHY register space	
13:12	Reserved	00b	Reserved	
11:2	Reg Address	0x0	Link/PHY register	
1	Word Select	0b	Selects upper/lower word of the DWord register in Link/PHY register space. 1b = Upper. 0b = Lower.	
0	Reserved	0b	Reserved	

6.4.3.2.2 PCIe SerDes Address Word (Bit 14 = 1b)

Bits	Name	Default	Description	Reserved
15	Reserved	0b	Reserved	
14	SerDes/Link PHY	0b	1b= Select PCIe SerDes register space.	
13	Quad 1 Select	0b	Select Quad 1 (Lane 4 to 7) for data write.	
12	Quad 0 Select	0b	Select Quad 0 (Lane 0 to 3) for data write.	
11:0	Reg Address	0x0	PCIe SerDes register.	

6.4.3.3 PCIe Analog Data — Offset 0x02, 0x04, 0x06...

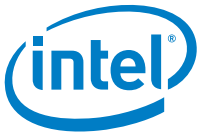
Each even offset word in the PCIe analog section is the register data. After reading a pair of address word and data word, the register specified in the address word is loaded with the data from the data word.

In case the write access is for Link/PHY space, 16 bits are written to the specified register.

In case the write access is for PCIe SerDes space, the lower data byte is directed to Quad 0 (Lanes 0 to 3) and the upper byte is directed to Quad 1 (Lanes 4 to 7).

6.4.4 PCIe General Configuration Module

This section is loaded after a PCIe Reset. It contains general configuration for the PCIe interface (not function specific) and is pointed to by word 0x06 in the NVM (full-byte address; must be word aligned).



Offset	Description
0x00	Section Length Section 6.4.4.1.
0x01	PCIe Init Configuration 1 Section 6.4.4.2
0x02	PCIe Init Configuration 2 Section 6.4.4.3
0x03	PCIe Init Configuration 3 Section 6.4.4.4
0x04	PCIe Control 1 Section 6.4.4.5
0x05	PCIe Control 2 Section 6.4.4.6
0x06	PCIe LAN Power Consumption Section 6.4.4.7
0x07	PCIe Control 3 Section 6.4.4.8
0x08	PCIe Sub-System ID Section 6.4.4.9
0x09	PCIe Sub-System Vendor ID Section 6.4.4.10
0x0A	PCIe Dummy Device ID Section 6.4.4.11
0x0B	PCIe Device Revision ID Section 6.4.4.12
0x0C	IOV Control Word 1 Section 6.4.4.13
0x0D	IOV Control Word 2 Section 6.4.4.14
0x0E	Reserved
0x0F	Reserved
0x10	Reserved
0x11	Serial Number Ethernet MAC Address Section 6.4.4.15
0x12	Serial Number Ethernet MAC Address Section 6.4.4.16
0x13	Serial Number Ethernet MAC Address Section 6.4.4.17
0x14	PCIe L1 Exit latencies Section 6.4.4.18
0x15	Spare Section 6.4.4.19

6.4.4.1 Section Length — Offset 0x00

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.

Bits	Name	Default	Description	Reserved
15:0	Section Length		Section Length in words.	

6.4.4.2 PCIe Init Configuration 1 — Offset 0x01

Bits	Name	Default	Description	Reserved
15	Reserved	0b	Reserved.	
14:12	L0s acceptable latency	011b	Loaded to the *Endpoint L0s Acceptable Latency* field in the Device Capabilities in the PCIe Configuration registers at power up.	
11:9	L0s G2 Sep exit latency	111b	L0s exit latency G2S. Loaded to *L0s Exit Latency* field in the Link Capabilities register in the PCIe configuration registers in PCIe Gen2 v1.0 (5GT/s) system at separate clock setting.	
8:6	L0s G2 Com exit latency	100b	L0s exit latency G2C. Loaded to *L0s Exit Latency* field in the Link Capabilities register in the PCIe configuration registers in PCIe Gen2 v1.0 (5GT/s) system at common clock setting.	
5:3	L0s G1 Sep exit latency	111b	L0s exit latency G1S. Loaded to *L0s Exit Latency* field in the Link Capabilities register in the PCIe configuration registers in PCIe Gen 1 v2.0 (2.5GT/s) system at Separate clock setting.	
2:0	L0s G1 Com exit latency	011b	L0s exit latency G1C. Loaded to *L0s Exit Latency* field in the Link Capabilities register in the PCIe configuration registers in PCIe Gen 1 v2.0 (2.5GT/s) system at common clock setting.	

6.4.4.3 PCIe Init Configuration 2 – Offset 0x02

Bits	Name	Default	Description	Reserved
15	IO_by_cfg	1b	Enable CSR access via PCIe Configuration space. When set, enables CSR access via the configuration registers located at configuration address space 0x98 and 0x9C.	
14	AER capability version	0b	0b = AER capability version is 2, as required for PCIe version 2.1. 1b - AER capability version is 1.	
13	Serial number disable	1b	When cleared, the PCIe serial number capability is exposed in the configuration space. Refer to Section 9.4.2. When set, the PCIe serial number capability is not exposed in the configuration space. The capability can also be disabled by setting the PCIe Serial Number Ethernet MAC Address bytes 3:0 to zero.	
12	FLR delay disable	0b	Disable FLR value in the FLR delay field in this word.	

Bits	Name	Default	Description	Reserved
11:8	FLR delay	0x1	FLR response time in cycles defines the delay from FLR assertion to its affect.	
7:6	PCI-E capability version	10b	PCIe Capability Version This field must be set to 10b to use extended configuration capability (used for a timeout mechanism). This field is mapped to GCR.PCIe_Capability_Version.	
5	ECRC generation enable	1b	Loaded into the ECRC Generation Capable bit of the PCIe Configuration registers. At 1b the device is capable of generating ECRC.	
4	ECRC check enable	1b	Loaded into the ECRC Check Capable bit of the PCIe Configuration registers. At 1b the device is capable of checking ECRC.	
3	FLR capability enable	1b	FLR capability Enable bit is loaded to PCIe Configuration registers (Device Capabilities).	
2	Cache line size	0b	Cache Line Size. 0b = 64 bytes. 1b = 128 bytes.	
1:0	Max payload size	10b	Maximum payload size support for TLPs. Loaded to PCIe Configuration registers (Device Capabilities).	

6.4.4.4 PCIe Init Configuration 3 – Offset 0x03

Bits	Name	Default	Description	Reserved
15:4	Reserved	0x0	Reserved	
3	GIO Down Reset Disable	0b	Disables a core and reset when the PCIe link goes down.	
2:1	Act_Stat_PM_Sup	11b	Active State Link PM Support is loaded to PCIe Configuration registers (Link Capabilities).	
0	Slot_Clock_Cfg	1b	Slot Clock Configuration. When set, the X540 uses the PCIe reference clock supplied on the connector (for add-in solutions). This bit is loaded to the PCIe Configuration registers (Link Status).	

6.4.4.5 PCIe Control 1 – Offset 0x04

Bits	Name	Default	Description	Reserved
15:6	Reserved	0x0	Reserved	
5	Wake Pin Enable	0b	Enables the use of the wake pin for a PME event in all power states.	

Bits	Name	Default	Description	Reserved
4:0	Reserved	11111b	Reserved	

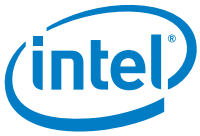
6.4.4.6 PCIe Control 2 – Offset 0x05

Bits	Name	Default	Description	Reserved
15	Completion Timeout Resend	0b	When set, enables a response to a request once the completion timeout expired. This bit is mapped to GCR.Completion_Timeout_Resend. 0b = Do not resend request on completion timeout. 1b = Resend request on completion timeout.	
14:4	Reserved	0x1	Reserved	
3	LAN Function Select[1]	0b	When the *LAN Function Select* field = 0b, LAN 0 is routed to PCI function 0 and LAN 1 is routed to PCI function 1. If the *LAN Function Select* field = 1b, LAN 0 is routed to PCI function 1 and LAN 1 is routed to PCI function 0. This bit is mapped to FACTPS[30].	
2	Dummy Function Enable[1]	1b	Controls the behavior of function 0 when disabled. See Section 4.4. 0b = Legacy mode. 1b = Dummy function mode.	
1	LAN PCI Disable[2]	0b	LAN PCI Disable. When set to 1b, 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 up to the PHY included.	
0	LAN Disable Select[2]	0b	LAN Disable Select 0b = LAN 0 is disabled. 1b = LAN 1 is disabled.	

1. When using the X540 single port configuration, bits3:2 are reserved and should be set to 0b.
2. When using the X540 single port configuration, bits 1:0 should be set to 1b.

6.4.4.7 PCIe LAN Power Consumption – Offset 0x06

Bits	Name	Default	Description	Reserved
15:8	LAN 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). Power is defined in 100 mW units. The power includes also the external logic required for the LAN function.	



Bits	Name	Default	Description	Reserved
7:5	Function 0 Common Power		The value in this field is reflected in the PCI Power Management Data register of function 0 when the Data_Select field is set to 8 (common function). The MSBs in the Data register that reflects the power values are padded with zeros. When one port is used this field should be set to 0.	
4:0	LAN 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). Power is defined in 100 mW units. The power includes also the external logic required for the LAN function. The MSBs in the Data register that reflects the power values are padded with zeros.	

6.4.4.8 PCIe Control 3 – Offset 0x07

Bits	Name	Default	Description	Reserved
15	Reserved	0b	Reserved	
14	PREFBAR	1b	Prefetchable bit indication in the memory BARs (loaded to the BARCTRL register): 0b = BARs are marked as non prefetchable. 1b = BARs are marked as prefetchable. This bit should be set only in systems that do not generate prefetchable cycles.	
13	CSR_Size	0b	The CSR_Size and *NVM Control Word 1 NVM Size* fields define the usable Flash size and CSR mapping window size as shown in BARCTRL register description.	
12	IO_Sup	1b	I/O Support (effects I/O BAR request). When set to 1b, I/O is supported. When cleared the *I/O Access Enable* bit in the Command Reg in the Mandatory PCI Configuration is RO at 0b.	
11	Reserved	1b	Reserved	
10:5	Reserved	0x30	Reserved	
4	DEV_OFF_EN	0b	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.	
3	Reserved	0b	Reserved	

Bits	Name	Default	Description	Reserved
2	SDP_FUNC_OFF_EN	0b	PCIe Function Off via SDP Pins Enable. 0b = Legacy mode (default), SDPn_1 pins does not control PCIe functions off. 1b = SDP1pins of both ports 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.	
1	Load Subsystem IDs	1b	When set to 1b, indicates that the function is to load its PCIe sub-system ID and sub-system vendor ID from the NVM (offset 0x8 and 0x9 in this section).	
0	Load Device ID	1b	When set to 1b, indicates that the function is to load its PCI device ID from the NVM (offset 0x0A in this section for Dummy Device ID and offset 2 in PCIe configuration space 0/1 section for active functions).	

6.4.4.9 PCIe Sub-System ID – Offset 0x08

If the load sub-system IDs in offset 0x7 of this section is set, this word is read in to initialize the sub-system ID. The default value is 0x0.

Bits	Name	Default	Description	Reserved
15:0	Sub System ID	0x0		

6.4.4.10 PCIe Sub-System Vendor ID – Offset 0x09

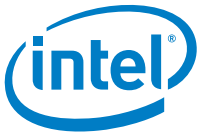
If the load sub-system IDs in offset 0x7 of this section is set, this word is read in to initialize the sub-system vendor ID. The default value is 0x8086.

Bits	Name	Default	Description	Reserved
15:0	Sub System Vendor	0x8086		

6.4.4.11 PCIe Dummy Device ID – Offset 0x0A

If the Load Device ID in offset 0x7 of this section is set, this word is read in to initialize the device ID of the dummy device in this function (if enabled). The default value is 0x10A6.

Bits	Name	Default	Description	Reserved
15:0	Sub Device_ID	0x10A6		



6.4.4.12 PCIe Device Revision ID – Offset 0x0B

Bits	Name	Default	Description	Reserved
15:8	Reserved	0x0	Set to 0x0.	
7:0	DEVREVID	0x0	Device Rev ID The actual device revision ID is the NVM value XORed with the hardware value (0x00 for the X540 A-0).	

6.4.4.13 IOV Control Word 1 — Offset 0x0C

This word controls the behavior of IOV functionality.

Bits	Name	Default	Description	Reserved
15:11	Reserved	0x0	Reserved	
10:5	Max VFs	0x3F	Defines the value of MaxVFs exposed in the IOV structure. Valid values are 0-63. The value exposed, is the value of this field + 1. **Note:** The queue pair of one VF should be assigned to the PF, therefore the maximum usable number of VFs is 63.	
4:3	MSI-X table	0x2	Defines the size of the MSI-X table (number of requested MSI-X vectors) — valid values are 0-2.	
2	64 bit advertisement	1b	0b = VF BARs advertise 32 bit size. 1b = VF BARs advertise 64 bit size.	
1	Prefetchable	0b	0b = IOV memory BARS (0 and 3) are declared as non prefetchable. 1b = IOV memory BARS (0 and 3) are declared as prefetchable.	
0	IOV enabled	1b	0b = IOV and ARI capability structures are not exposed as part of the capabilities link list. 1b = IOV and ARI capability structures are exposed as part of the capabilities link list.	

6.4.4.14 IOV Control Word 2 — Offset 0x0D

Bits	Name	Default	Description	Reserved
15:0	VirtDev ID	0x1515	Virtual function Device ID.	No

6.4.4.15 Serial Number Ethernet MAC Address — Offset 0x11

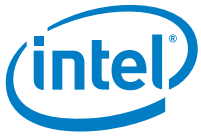
Bits	Name	Default	Description	Reserved
15:8	Serial Number Ethernet MAC Address 0, Byte 1		Part of the Ethernet MAC address used to generate the PCIe serial number. See Section 9.4.2.	
7:0	Serial Number Ethernet MAC Address 0, Byte 0		Part of the Ethernet MAC address used to generate the PCIe serial number. See Section 9.4.2.	

6.4.4.16 Serial Number Ethernet MAC Address — Offset 0x12

Bits	Name	Default	Description	Reserved
15:8	Serial Number Ethernet MAC Address 0, Byte 3		Part of the Ethernet MAC address used to generate the PCIe serial number. See Section 9.4.2.	
7:0	Serial Number Ethernet MAC Address 0, Byte 2		Part of the Ethernet MAC address used to generate the PCIe serial number. See Section 9.4.2.	

6.4.4.17 Serial Number Ethernet MAC Address — Offset 0x13

Bits	Name	Default	Description	Reserved
15:8	Serial Number Ethernet MAC Address 0, Byte 5		Part of the Ethernet MAC address used to generate the PCIe serial number. See Section 9.4.2	
7:0	Serial Number Ethernet MAC Address 0, Byte 4		Part of the Ethernet MAC address used to generate the PCIe serial number. See Section 9.4.2.	



6.4.4.18 PCIe L1 Exit Latencies — Offset 0x14

Bits	Name	Default	Description	Reserved
15	Reserved	1b	Reserved.	
14:12	L1_Act_Acc_Latency	110b	Loaded to the *Endpoint L1 Acceptable Latency* field in the "Device Capabilities" in the "PCIe configuration registers" at power up.	
11:9	L1 G2 Sep exit latency	101b	L1 exit latency G2S. Loaded to Link Capabilities (L1 Exit Latency) at PCIe Gen2 v1.0 (5GT/s) system in separate clock setting.	
8:6	L1 G2 Com exit latency	011b	L1 exit latency G2C. Loaded to Link Capabilities (L1 Exit Latency) at PCIe Gen2 v1.0 (5GT/s) system in common clock setting.	
5:3	L1 G1 Sep exit latency	100b	L1 exit latency G1S. Loaded to Link Capabilities (L1 Exit Latency) at PCIe Gen 1 v2.0 (2.5GT/s) system in separate clock setting.	
2:0	L1 G1 Com exit latency	010b	L1 exit latency G1C. Loaded to Link Capabilities (L1 Exit Latency) at PCIe Gen 1 v2.0 (2.5GT/s) system in common clock setting.	

6.4.4.19 Reserved — Offset 0x15

Bits	Name	Default	Description	Reserved
15:0	Reserved	0x0003	Reserved	

6.4.5 PCIe Configuration Space 0/1 Modules

Word 0x7 points to the PCIe configuration space defaults of function 0 while word 0x8 points to function 1 defaults. Both sections are loaded after PCIe Reset and D3 to D0 transition of the specific function. The structures of both functions are identical as listed in the following table.

Offset	Description
0x00	Section Length Section 6.4.5.1.
0x1	Control Word Section 6.4.5.2
0x2	Device ID Section 6.4.5.3
0x3	Reserved
0x4	Reserved
0x5	Reserved Section 6.4.5.4

6.4.5.1 Section Length — Offset 0x00

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.

Bits	Name	Default	Description	Reserved
15:0	Section Length	0x0	Section length in words.	

6.4.5.2 Control Word — Offset 0x01

Bits	Name	Default	Description	Reserved
15:14	Reserved	00b	Reserved	
13:12	Interrupt Pin	0b for Lan0 1b for Lan1	Controls the value advertised in the Interrupt Pin field of the PCI configuration header for this device/function. Values of 00b, 01b, 10b and 11b correspond to INTA#, INTB#, INTC# and INTD# respectively. When one port is used this field must be set to 00b (using INTA#) to comply with PCI spec requirements.	
11	Storage Class	0b	When set, the class code of this port is set to 0x010000 (SCSI). When cleared, the class code of this port is set to 0x020000 (LAN).	
10	MSI Mask	1b	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.	
9	Reserved	1b	Reserved	
8	LAN Boot Disable	1b	A value of 1b disables the expansion ROM BAR in the PCI configuration space.	
7	Reserved	0b	Reserved	
6:0	MSI_X_N	0x3F	This field specifies the number of entries in the MSI-X tables for this function. MSI_X_N is equal to the number of entries minus one. For example, a return value of 0x7 means 8 vectors are available. This field is loaded into the PCIe MSI-X capabilities structure. The MSI_X_N must not exceed 0x3F (64 MSI-X vectors).	

6.4.5.3 Device ID — Offset 0x02 Device ID

Bits	Name	Default	Description	Reserved
15:0	Device_ID	0x1512	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.	

6.4.5.4 Spare 0/1 — Offset 0x05

Bits	Name	Default	Description	Reserved
15:0	Reserved	0	Reserved	

6.4.6 LAN Core 0/1 Modules

Word 0x9 points to the core configuration defaults of LAN port 0 while word 0xA points to LAN port 1 defaults. The section of each function is loaded at the de-assertion of its core master reset: PCIe Reset, D3 to D0 transition, software reset and link reset. The structures of both functions are identical as listed in the following table.

Offset	High Byte[15:8]	Low Byte[7:0]	Section
0x0	Section Length - Section 6.4.6.1.		
0x1	Ethernet MAC Address Byte 2	Ethernet MAC Address Byte 1	Section 6.4.6.2.1
0x2	Ethernet MAC Address Byte 4	Ethernet MAC Address Byte 3	Section 6.4.6.2.2
0x3	Ethernet MAC Address Byte 6	Ethernet MAC Address Byte 5	Section 6.4.6.2.3
0x4	LED 1 configuration	LED 0 Configuration	Section 6.4.6.3.1
0x5	LED 3 Configuration	LED 2 Configuration	Section 6.4.6.3.2
0x6	SDP Control		Section 6.4.6.4
0x7	Filter Control		Section 6.4.6.5

6.4.6.1 Section Length — Offset 0x00

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.

Bits	Name	Default	Description	Reserved
15:0	Section Length	0x0	Section length in words.	

6.4.6.2 Ethernet MAC Address Registers

The Ethernet Individual Address (IA) is a 6-byte field that must be unique for each NIC 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

6.4.6.2.1 Ethernet MAC Address Register1 – Offset 0x01

Bits	Name	Default	Description	Reserved
15:8	Eth_Addr_Byte2	0x0	Ethernet MAC Address Byte2	
7:0	Eth_Addr_Byte1	0x0	Ethernet MAC Address Byte1	

6.4.6.2.2 Ethernet MAC Address Register2 – Offset 0x02

Bits	Name	Default	Description	Reserved
15:8	Eth_Addr_Byte4	0x0	Ethernet MAC Address Byte4	
7:0	Eth_Addr_Byte3	0x0	Ethernet MAC Address Byte3	

6.4.6.2.3 Ethernet MAC Address Register3 – Offset 0x03

Bits	Name	Default	Description	Reserved
15:8	Eth_Addr_Byte6	0x0	Ethernet MAC Address Byte6	
7:0	Eth_Addr_Byte5	0x0	Ethernet MAC Address Byte5	

6.4.6.3 LED Configuration

The LEDCTL register defaults are loaded from two words as listed in the following tables.

6.4.6.3.1 LED Control Lower Word – Offset 0x04

Bits	Name	Default	Description	Reserved
15:8	LEDCTL1	0x0	LED 1 Control	
7:0	LEDCTL0	0x0	LED 0 Control	

6.4.6.3.2 LED Control Upper Word – Offset 0x05

Bits	Name	Default	Description	Reserved
15:8	LEDCTL3	0x0	LED 3 Control	
7:0	LEDCTL2	0x0	LED 2 Control	

Note: The content of the NVM words is similar to the register content.

6.4.6.4 SDP Control — Offset 0x06

Bits	Name	Default	Description	Reserved
15	SDP1_NATIVE	0b	Defines SDP1 operating mode is mapped to ESDP.SDP1_NATIVE loaded at power up: 0b = Operates as generic software controlled I/O. 1b = Native mode operation (hardware function).	
14:12	Reserved	0x0	Set to 0x0.	
11	SDPDIR[3]	0b	SDP3 Pin. Initial direction is mapped to ESDP.SDP3_IODIR loaded at power up.	
10	SDPDIR[2]	0b	SDP2 Pin. Initial Direction is mapped to ESDP.SDP2_IODIR loaded at power up.	
9	SDPDIR[1]	0b	SDP1 Pin. Initial Direction is mapped to ESDP.SDP1_IODIR loaded at power up.	
8	SDPDIR[0]	0b	SDP0 Pin. Initial Direction is mapped to ESDP.SDP0_IODIR loaded at power up.	
7:4	Reserved	0x0	Reserved	

Bits	Name	Default	Description	Reserved
3	SDPVAL[3]	0b	SDP3 Pin. Initial Output Value is mapped to ESDP.SDP3_DATA loaded at power up.	
2	SDPVAL[2]	0b	SDP2 Pin. Initial Output Value is mapped to ESDP.SDP2_DATA loaded at power up.	
1	SDPVAL[1]	0b	SDP1 Pin. Initial Output Value is mapped to ESDP.SDP1_DATA loaded at power up.	
0	SDPVAL[0]	0b	SDP0 Pin. Initial Output Value is mapped to ESDP.SDP0_DATA loaded at power up.	

6.4.6.5 Filter Control – Offset 0x07

Bits	Name	Default	Description	Reserved
15:0	Reserved	0x0001	Reserved	

6.4.7 MAC 0 / 1 Modules

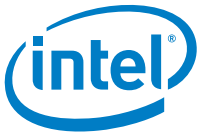
Word 0xB points to the LAN MAC configuration defaults of function 0 while word 0xC points to function 1 defaults. Both sections are loaded at the de-assertion of their core master reset. The structures of both sections are identical.

6.4.8 CSR 0 / 1 Auto Configuration Modules

Word 0xD points to the CSR auto configuration of function 0 while word 0xE points to function 1. Both sections are loaded at the de-assertion of their core master reset.

The structures of both sections are identical; the structure is listed in the following table.

Offset	High Byte[15:8]	Low Byte[7:0]	Section
0x0	Section Length = 3*n		
0x1	CSR Address		Section 6.4.8.2
0x2	Data LSB		Section 6.4.8.3
0x3	Data MSB		Section 6.4.8.4



Offset	High Byte[15:8]	Low Byte[7:0]	Section
	…		
3*n — 2	CSR Address		Section 6.4.8.2
3*n — 1	Data LSB		Section 6.4.8.3
3*n	Data MSB		Section 6.4.8.4

Note: The X540 blocks any write to the analog configuration registers through these sections.

6.4.8.1 Section Length — Offset 0x0

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.

Bits	Name	Default	Description	Reserved
15:0	Section_length	0x0	Section length in words.	

6.4.8.2 CSR Address — Offset 0x1, 0x4, 0x7...

Bits	Name	Default	Description	Reserved
15:0	CSR_ADDR	0x0	CSR Address bits [16:1], assuming bit [0] is always cleared. This allows coverage of full CSR address range [16:0].	

6.4.8.3 CSR Data LSB — Offset 0x2, 0x5, 0x8...

Bits	Name	Default	Description	Reserved
15:0	CSR_Data_LSB	0x0	CSR Data LSB.	

6.4.8.4 CSR Data MSB — Offset 0x3, 0x6, 0x9...

Bits	Name	Default	Description	Reserved
15:0	CSR_Data_MSB	0x0	CSR Data MSB	

6.5 Firmware Module

The Firmware Module is formed by a header and sub-modules. Content of the header is listed in Table 6-7, it is a list of "logical" pointers to all the firmware sub-modules. A firmware "logical" pointer is basically an offset from a base address. It is converted to a "physical" NVM word address as follows:

— A firmware sub-module "logical" pointer greater or equal to 2K corresponds to a firmware sub-module that is "physically" mapped in the Firmware Extension Module area.

  Its base address expressed in word units is: [(Firmware Module Pointer) x 2K], as the Firmware Module Pointer read from NVM word address 0x0F is expressed in 2K word units.

— A firmware sub-module "logical" pointer smaller to 2K corresponds to a firmware sub-module that is "physically" mapped in the Legacy EEPROM Modules area. Its base address is 0 when using EEPROM-Mode access.

The Firmware Module is organized as a continuous "logical" address range, which is "physically" mapped across two separate physical NVM regions, the Legacy EEPROM Modules area and the Firmware Extension Module area.

Note: As a side effect, any firmware sub-module mapped in the Firmware Extension Module region is mapped beyond the first 2K words from the beginning of this region. The first 2K words of the Firmware Extension Module region contains only the Firmware Module Header, leaving the remaining words unusable.

The Firmware Module Header listed in Table 6-7 is always physically mapped at the beginning of the Firmware Extension Module region.

An example showing how to calculate the firmware module header is as follows:

Word 0xF is set to 8002. Pointer is 2 (bits 14:0) in 4 KB sectors (indicated by bit 15), which equates to 8 KB.

8192/2 = 4096 words. In hex equates to 0x1000, which is where the firmware module header starts.

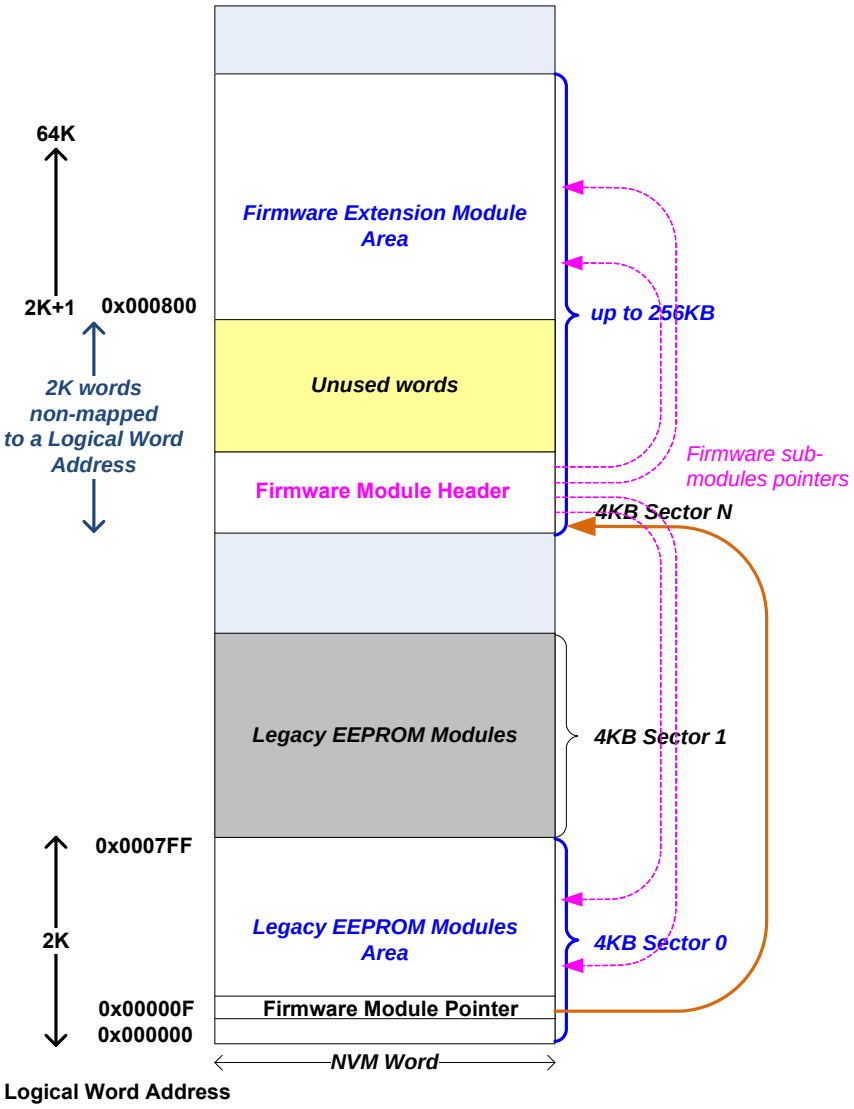


Figure 6-2 Firmware Module Organization

Table 6-7 Firmware Module Header

Global MNG Word Offset	Description
0x0	Test Configuration Pointer - Section 6.5.1
0x1	Loader Patch Pointer -Section 6.5.1
0x2	No Manageability Patch Pointer (Patch structure identical to the Loader Patch) - Section 6.5.6

Table 6-7 Firmware Module Header

Global MNG Word Offset	Description
0x3	Common Firmware Parameters - Section 6.5.2
0x4	Pass Through Patch Configuration Pointer (Patch structure identical to the Loader Patch) - Section 6.5.3
0x5	Pass Through LAN 0 Configuration Pointer - Section 6.5.3
0x6	SideBand Configuration Pointer - Section 6.5.4
0x7	Flexible TCO Filter Configuration Pointer - Section 6.5.4.11
0x8	Pass Through LAN 1 Configuration Pointer - Section 6.5.3
0x9	NC-SI Microcode Download Pointer - Section 6.5.6
0xA	NC-SI Configuration Pointer - Section 6.5.7
0xB	Traffic Types Parameters - Section 6.5.8
0xC	Reserved

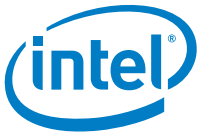
6.5.1 Test Configuration Module (Global MNG Offset 0x00)

6.5.1.1 Section Header — Offset 0x0

Bits	Name	Default	Description	Reserved
15:8	Block CRC			
7:0	Block Length		Block length in words.	

6.5.2 Common Firmware Parameters – (Global MNG Offset 0x3)

Bits	Name	Default	Description	Reserved
15	Enable Firmware Reset		0b = Firmware reset via HICR is disabled. 1b = Firmware reset via HICR is enabled.	



Bits	Name	Default	Description	Reserved
14:13	Redirection Sideband Interface		00b = SMBus. 01b = NC-SI. 10b = MCTP. 11b = Reserved.	
12	Restore MAC Address		0b = Do not restore MAC address at power on. 1b = Restore MAC address at power on.	
11	Reserved	0b	Reserved	
10:8	Manageability Mode		0x0 = None. 0x1 = Reserved. 0x2 = Pass Through (PT) mode. 0x3:0x7 = Reserved.	
7:6	Reserved	11b	Reserved.	
5	LAN1 Force TCO Reset Disable[1]	1b	0b = Enable Force TCO reset on LAN1. 1b = Disable Force TCO reset on LAN1.	
4	LAN0 Force TCO Reset Disable	1b	0b = Enable Force TCO reset on LAN0. 1b = Disable Force TCO reset on LAN0.	
3	Enable All PHYs in D3 N	1b	NC-SI mode only: 0b - Only ports activated for MC activity will stay active in D3, according to enable channel command to the specific port. 1b - All PHYs will stay active in D3 if PT is used.	
2	OS2BMC Capable		0b = Disable. 1b = Enable.	
1:0	Reserved	11b	Reserved	

1. This bit is ignored when using the X540 single port configuration.

6.5.3 Pass Through LAN 0/1 Configuration Modules (Global MNG Offsets 0x05 and 0x08)

The following sections describe pointers and structures dedicated to pass-through mode for LAN 0 and LAN 1. LAN 0 structure is pointed by the Firmware Module pointer at offset 0x5. LAN 1 structure is pointed by the Firmware Module pointer at offset 0x8.

Note: When using the the X540 single port configuration, the LAN 1 configuration is ignored.

6.5.3.1 Section Header — Offset 0x0

Bits	Name	Default	Description	Reserved
15:8	Block CRC8			
7:0	Block Length		Block length in words.	

6.5.3.2 LAN 0/1 IPv4 Address 0 (LSB) MIPAF0 — Offset 0x01

Bits	Name	Default	Description	Reserved
15:8			LAN 0/1 IPv4 Address 0, Byte 1	
7:0			LAN 0/1 IPv4 Address 0, Byte 0	

6.5.3.3 LAN 0/1 IPv4 Address 0 (MSB) (MIPAF0) — Offset 0x02

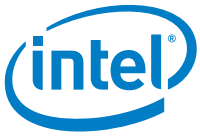
Bits	Name	Default	Description	Reserved
15:8			LAN 0/1 IPv4 Address 0, Byte 3	
7:0			LAN 0/1 IPv4 Address 0, Byte 2	

6.5.3.4 LAN 0/1 IPv4 Address 1 MIPAF1 — Offset 0x03-0x04

Same structure as LAN0 IPv4 Address 0.

6.5.3.5 LAN 0/1 IPv4 Address 2 MIPAF2 — Offset 0x05-0x06

Same structure as LAN0 IPv4 Address 0.



6.5.3.6 LAN 0/1 IPv4 Address 3 MIPAF3 — Offset 0x07-0x08

Same structure as LAN0 IPv4 Address 0.

6.5.3.7 LAN 0/1 Ethernet MAC Address 0 (LSB) MMAL0 — Offset 0x09

This word is loaded by firmware to the 16 LS bits of the MMAL[0] register.

Bits	Name	Default	Description	Reserved
15:8			LAN 0/1 Ethernet MAC Address 0, Byte 1	
7:0			LAN 0/1 Ethernet MAC Address 0, Byte 0	

6.5.3.8 LAN 0/1 Ethernet MAC Address 0 (Mid) MMAL0 — Offset 0x0A

This word is loaded by firmware to the 16 MS bits of the MMAL[0] register.

Bits	Name	Default	Description	Reserved
15:8			LAN 0/1 Ethernet MAC Address 0, Byte 3	
7:0			LAN 0/1 Ethernet MAC Address 0, Byte 2	

6.5.3.9 LAN 0/1 Ethernet MAC Address 0 (MSB) MMAH0 — Offset 0x0B

This word is loaded by firmware to the MMAH[0] register.

Bits	Name	Default	Description	Reserved
15:8			LAN 0/1 Ethernet MAC Address 0, Byte 5	
7:0			LAN 0/1 Ethernet MAC Address 0, Byte 4	

6.5.3.10 LAN 0/1 Ethernet MAC Address 1 MMAL/H1 — Offset 0x0C-0x0E

Same structure as LAN0 Ethernet MAC Address 0. Loaded to MMAL[1], MMAH[1].

6.5.3.11 LAN 0/1 Ethernet MAC Address 2 MMAL/H2 — Offset 0x0F-0x11

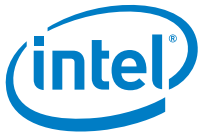
Same structure as LAN0 Ethernet MAC Address 0. Loaded to MMAL[2], MMAH[2].

6.5.3.12 LAN 0/1 Ethernet MAC Address 3 MMAL/H3 — Offset 0x12-0x14

Same structure as LAN0 Ethernet MAC Address 0. Loaded to MMAL[3], MMAH[3].

6.5.3.13 LAN 0/1 UDP Flexible Filter Ports 0 — 15 (MFUTP Registers) — Offset 0x15 - 0x24

Offset	Bits	Description	Reserved
0x15	15:0	LAN UDP Flexible Filter Value Port0	
0x16	15:0	LAN UDP Flexible Filter Value Port1	
0x17	15:0	LAN UDP Flexible Filter Value Port2	
0x18	15:0	LAN UDP Flexible Filter Value Port3	
0x19	15:0	LAN UDP Flexible Filter Value Port4	
0x1A	15:0	LAN UDP Flexible Filter Value Port5	
0x1B	15:0	LAN UDP Flexible Filter Value Port6	
0x1C	15:0	LAN UDP Flexible Filter Value Port7	
0x1D	15:0	LAN UDP Flexible Filter Value Port8	
0x1E	15:0	LAN UDP Flexible Filter Value Port9	
0x1F	15:0	LAN UDP Flexible Filter Value Port10	
0x20	15:0	LAN UDP Flexible Filter Value Port11	
0x21	15:0	LAN UDP Flexible Filter Value Port12	



Offset	Bits	Description	Reserved
0x22	15:0	LAN UDP Flexible Filter Value Port13	
0x23	15:0	LAN UDP Flexible Filter Value Port14	
0x24	15:0	LAN UDP Flexible Filter Value Port15	

6.5.3.14 LAN 0/1 VLAN Filter 0 — 7 (MAVTV Registers) — Offset 0x25 — 0x2C

Offset	Bits	Description	Reserved
0x25	15:12	Reserved	
0x25	11:0	LAN 0/1 VLAN Filter 0 Value	
0x26	15:12	Reserved	
0x26	11:0	LAN 0/1 VLAN Filter 1 Value	
0x27	15:12	Reserved	
0x27	11:0	LAN 0/1 VLAN Filter 2 Value	
0x28	15:12	Reserved	
0x28	11:0	LAN 0/1 VLAN Filter 3 Value	
0x29	15:12	Reserved	
0x29	11:0	LAN 0/1 VLAN Filter 4 Value	
0x2A	15:12	Reserved	
0x2A	11:0	LAN 0/1 VLAN Filter 5 Value	
0x2B	15:12	Reserved	
0x2B	11:0	LAN 0/1 VLAN Filter 6 Value	
0x2C	15:12	Reserved	
0x2C	11:0	LAN 0/1 VLAN Filter 7 Value	

6.5.3.15 LAN 0/1 Manageability Filters Valid (MFVAL LSB) — Offset 0x2D

Bits	Name	Default	Description	Reserved
15:8	VLAN		Indicates if the VLAN filter registers (MAVTV) contain valid VLAN tags. Bit 8 corresponds to filter 0, etc.	
7:4	Reserved			
3:0	MAC		Indicates if the MAC unicast filter registers (MMAH, MMAL) contain valid Ethernet MAC Addresses. Bit 0 corresponds to filter 0, etc.	

6.5.3.16 LAN 0/1 Manageability Filters Valid (MFVAL MSB) — Offset 0x2E

Bits	Name	Default	Description	Reserved
15:12	Reserved			
11:8	IPv6		Indicates if the IPv6 address filter registers (MIPAF) contain valid IPv6 addresses. Bit 8 corresponds to address 0, etc. Bit 11 (filter 3) applies only when IPv4 address filters are not enabled (MANC.EN_IPv4_FILTER=0).	
7:4	Reserved		Reserved	
3:0	IPv4		Indicates if the IPv4 address filters (MIPAF) contain a valid IPv4 address. These bits apply only when IPv4 address filters are enabled (MANC.EN_IPv4_FILTER=1).	

6.5.3.17 LAN 0/1 MANC value LSB (LMANC LSB) — Offset 0x2F

Bits	Name	Default	Description	Reserved
15:0	Reserved	0x0	Reserved	

6.5.3.18 LAN 0/1 MANC Value MSB (LMANC MSB) — Offset 0x30

Bits	Name	Default	Description	Reserved
15:9	Reserved		Reserved.	
8	Enable IPv4 Address Filters		This bit is loaded to the EN_IPv4_FILTER bit in the MANC register.	
7	Enable Xsum Filtering to MNG		This bit is loaded to the EN_XSUM_FILTER bit in the MANC register.	
6	VLAN MNG Filtering		This bit is loaded to the *Bypass VLAN* bit in the MANC register.	
5	Enable MNG Packets to Host Memory		This bit is loaded to the EN_MNG2HOST bit in the MANC register.	
4:0	Reserved		Reserved.	

6.5.3.19 LAN 0/1 Receive Enable 1 (LRXEN1) — Offset 0x31

Bits	Name	Default	Description	Reserved
15:8	Receive Enable byte 12		MC SMBus slave address.	
7	Enable BMC Dedicated MAC			
6	Reserved		Reserved Must be set to 1b.	
5:4	Notification method		00b = SMBus alert. 01b = Asynchronous notify. 10b = Direct receive. 11b = Reserved.	
3	Enable ARP Response			
2	Enable Status Reporting			
1	Enable Receive All			
0	Enable Receive TCO			

6.5.3.20 LAN 0/1 Receive Enable 2 (LRXEN2) — Offset 0x32

Bits	Name	Default	Description	Reserved
15:8	Receive Enable byte 14	0x0	Alert Value	
7:0	Receive Enable byte 13	0x0	Interface Data	

6.5.3.21 LAN 0/1 MANC2H Value (LMANC2H LSB) — Offset 0x33

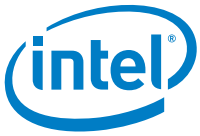
Bits	Name	Default	Description	Reserved
15:8	Reserved			
7:0	Host Enable		When set, indicates that packets routed by the manageability filters to the manageability block are also sent to the host. Bit 0 corresponds to decision rule 0, etc.	

6.5.3.22 LAN 0/1 MANC2H Value — LMANC2H MSB- Offset 0x34

Bits	Name	Default	Description	Reserved
15:0	Reserved	0x0	Reserved	

6.5.3.23 Manageability Decision Filters — MDEF0 (1) — Offset 0x35

Bits	Name	Default	Description	Reserved
15:0	MDEF0_L		Loaded to 16 LS bits of MDEF[0] register.	



6.5.3.24 Manageability Decision Filters — MDEF0 (2) — Offset 0x36

Bits	Name	Default	Description	Reserved
15:0	MDEF0_M		Loaded to 16 MS bits of MDEF[0] register.	

6.5.3.25 Manageability Decision Filters — MDEF0 (3) — Offset 0x37

Bits	Name	Default	Description	Reserved
15:0	MDEFEXT0_L		Loaded to 16 LS bits of MDEF_EXT[0] register.	

6.5.3.26 Manageability Decision Filters — MDEF0 (4) — Offset 0x38

Bits	Name	Default	Description	Reserved
15:0	MDEF0EXT_M		Loaded to 16 MS bits of MDEF_EXT[0] register.	

6.5.3.27 Manageability Decision Filters — MDEF1-6 (1-4) — Offset 0x39-0x50

Same as words 0x035...0x38 for MDEF[1] and MDEF_EXT[1]...MDEF[6] and MDEF_EXT[6]

6.5.3.28 Manageability Ethertype Filter 0.1 — METF0 (1) - Offset 0x51

Bits	Name	Default	Description	Reserved
15:0	METF0_L		Loaded to 16 LS bits of METF[0] register.	

6.5.3.29 Manageability Ethertype Filter 0.2 — METF0 (2) - Offset 0x52

Bits	Name	Default	Description	Reserved
15:0	METF0_M		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).	

6.5.3.30 Manageability Ethertype Filter 1...3 (1 and 2) — METF1...3 - Offset 0x53...0x58

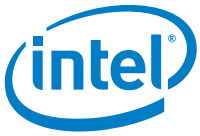
Same as words 0x51 and 0x52 for METF[1]...METF[3] registers.

6.5.3.31 ARP Response IPv4 Address 0 (LSB) — Offset 0x59

Bits	Name	Default	Description	Reserved
15:0			ARP Response IPv4 Address 0, Byte 1 (firmware use).	
7:0			ARP Response IPv4 Address 0, Byte 0 (firmware use).	

6.5.3.32 ARP Response IPv4 Address 0 (MSB) — Offset 0x5A

Bits	Name	Default	Description	Reserved
15:8			ARP Response IPv4 Address 0, Byte 3 (firmware use).	
7:0			ARP Response IPv4 Address 0, Byte 2 (firmware use).	



6.5.3.33 LAN 0/1 IPv6 Address 0 (n=0...7) (MIPAF.IPV6ADDR0) — Offset 0x5B...0x62

Bits	Name	Default	Description	Reserved
15:0			Loaded to MIPAF registers IPV6ADDR0: DWORD offset 'n'/2 to the lower 16 bits for even 'n' and upper 16 bits for odd 'n'. For 'n' = 0...7.	

6.5.3.34 LAN 0/1 IPv6 Address 1 (MIPAF.IPV6ADDR1) — Offset 0x63-0x6A

Same structure as LAN 0/1 IPv6 Address 0.

6.5.3.35 LAN 0/1 IPv6 Address 2 (MIPAF) — Offset 0x6B:0x72

Same structure as LAN 0/1 IPv6 Address 0.

6.5.4 Sideband Configuration Module (Global MNG Offset 0x06)

This module is pointed to by global offset 0x06 of the manageability control table.

6.5.4.1 Section Header — Offset 0x0

Bits	Name	Default	Description	Reserved
15:8	Block CRC8	0x0		
7:0	Block Length	0x0	Section length in words.	

6.5.4.2 SMBus Maximum Fragment Size — Offset 0x01

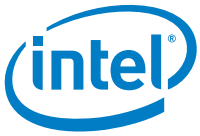
Bits	Name	Default	Description	Reserved
15:0	Max Fragment Size	0x0	SMBus Maximum Fragment Size (bytes). Supported range is between 32 and 240 bytes. **Note**: In MCTP mode, this value should be set to 0x45 (64 bytes payload + 5 bytes of MCTP header).	

6.5.4.3 SMBus Notification Timeout and Flags — Offset 0x02

Bits	Name	Default	Description	Reserved
15:8	SMBus Notification Timeout (ms)			
7:6	SMBus Connection Speed		00b = Standard SMBus connection. 01b = Reserved. 10b = Reserved. 11b = Reserved.	
5	SMBus Block Read Command		0b = Block read command is 0xC0. 1b = Block read command is 0xD0.	
4	SMBus Dual Port Mode	1b	0b = Single port configuration. 1b = SMBus dual port mode.	
3	Enable fairness arbitration		0b = Disable fairness arbitration. 1b = Enable fairness arbitration.	
2	Disable SMBus ARP Functionality			
1	SMBus ARP PEC			
0	Reserved		Reserved.	

6.5.4.4 SMBus Slave Addresses — Offset 0x03

Bits	Name	Default	Description	Reserved
15:9	SMBus 1 Slave Address		Dual address mode only.	
8	Reserved		Reserved.	
7:1	SMBus 0 Slave Address			



Bits	Name	Default	Description	Reserved
0	Reserved		Reserved.	

6.5.4.5 Reserved — Offset 0x04

Bits	Name	Default	Description	Reserved
15:0	Reserved		Reserved.	

6.5.4.6 Reserved — Offset 0x05

Bits	Name	Default	Description	Reserved
15:0	Reserved		Reserved.	

6.5.4.7 NC-SI Configuration 1 - Offset 0x06

Bits	Name	Default	Description	Reserved
15:11	Reserved		Reserved.	
10	Reserved	0b	Must be set to 0b.	
9	NC-SI HW Arbitration Enable	0b	0b = Not supported. ~~Must be set to 0b.~~ 1b = Supported.	
8	NC-SI HW-based Packet Copy Enable	1b	0b = Disable. 1b = Enable.	
7:5	Package ID	0b	Meaningful only when bit 15 of NC-SI Configuration 2 word (offset 0x07) is cleared.	
4:0	Reserved	0x0	Reserved, must be set to 0x0.	

6.5.4.8 NC-SI Configuration 2 - Offset 0x07

Bits	Name	Default	Description	Reserved
15	NC-SI Package ID from SDP	0b	Read NC-SI Package ID from 0b = NVM, NC-SI Configuration 1 word bits 7:5 (offset 0x06). 1b = SDP, SDPn_0 pins of LAN ports n=1,0 (where SDP0_0 is the LS bit of NC-SI Package ID). **Note:** See section 3.5 for more details.	
14:4	Reserved		Reserved.	
3:0	Reserved		Reserved	

6.5.4.9 NC-SI Hardware Arbitration Configuration - Offset 0x08

Bits	Name	Default	Description	Reserved
15:0	Token Timeout	0xC800	NC-SI HW-Arbitration TOKEN Timeout (in 16 ns cycles). In order to get the value if NC-SI REF_CLK cycles, this field should be multiplied by 4/5. Setting value to 0 disables the timeout mechanism.	

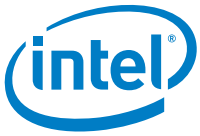
6.5.4.10 Reserved Words - Offset 0x09 — 0x0D

Reserved for future use.

6.5.4.11 MCTP UUID - Time Low LSB (0ffset 0x09)

The value stored in the MCTP UUID register should indicate the creation date of the image or an earlier arbitrary date.

Bits	Name	Default	Description	Reserved
15:0	time low LSB		Byte 0 & 1 of UUID as defined in DSP0236	



6.5.4.12 MCTP UUID - Time Low MSB (0ffset 0x0A)

Bits	Name	Default	Description	Reserved
15:0	time low MSB		Byte 2 & 3 of UUID as defined in DSP0236	

6.5.4.13 MCTP UUID - Time MID (0ffset 0x0B)

Bits	Name	Default	Description	Reserved
15:0	time mid		Byte 4 & 5 of UUID as defined in DSP0236	

6.5.4.14 MCTP UUID - Time High and Version (0ffset 0x0C)

Bits	Name	Default	Description	Reserved
15:0	time high and version		Byte 7 & 8 of UUID as defined in DSP0236	

6.5.4.15 MCTP UUID - Clock Seq (0ffset 0x0D)

Bits	Name	Default	Description	Reserved
15:0	Clock seq and reserved		Byte 9 & 10 of UUID as defined in DSP0236	

6.5.4.16 Alternative IANA - 0ffset 0x0E

Bits	Name	Default	Description	Reserved
15:0	Alternative IANA number	0x0	If not zero and not 0x157, the X540 will accept NC-SI OEM commands with this IANA number.	

6.5.4.17 NC-SI over MCTP configuration - 0x0F

Bits	Name	Default	Description	Reserved
15:8	NC-SI packet type	0x2	Defines the MCTP packet type used to identify NC-SI packets	
7	Simplified MCTP	0x0	If set, a payload type byte is expected in NC-SI over MCTP packets after the packet type - otherwise, it is assumed all the packets received and sent are NC-SI control packets (commands, responses and AENs). In this mode, only SOM & EOM bits are used for the reassembly process. Relevant only in SMBus mode	
6:0	Reserved	0x0	Reserved	

6.5.5 Flexible TCO Filter Configuration Module (Global MNG Offset 0x07)

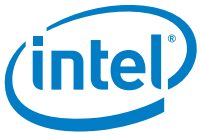
This module is pointed to by global offset 0x07 of the manageability control section.

6.5.5.1 Section Header — Offset 0x0

Bits	Name	Default	Description	Reserved
15:8	Block CRC8	0x0		
7:0	Block Length	0x0	Section length in words.	

6.5.5.2 Flexible Filter Length and Control — Offset 0x01

Bits	Name	Default	Description	Reserved
15:8	Flexible Filter Length (bytes)			
7:5	Reserved		Reserved	
4	Last Filter			
3:2	Filter Index (0-3)			
1	Apply Filter to LAN 1			
0	Apply Filter to LAN 0			



6.5.5.3 Flexible Filter Enable Mask — Offset 0x02 – 0x09

Bits	Name	Default	Description	Reserved
15:0	Flexible Filter Enable Mask			

6.5.5.4 Flexible Filter Data — Offset 0x0A – Block Length

Bits	Name	Default	Description	Reserved
15:0	Flexible Filter Data			

Note: This section loads all of the flexible filters, The control + mask + filter data are repeatable as the number of filters. Section length in offset 0 is for all filters.

6.5.6 NC-SI Microcode Download Module (Global MNG Offset 0x09)

This module is pointed to by global offset 0x09 of the manageability control table.

6.5.6.1 Patch Data Size — Offset 0x0

6.5.6.2 Rx and Tx Code Size — Offset 0x1

Bits	Name	Default	Description	Reserved
15:8	Rx Code Length	0x0	Rx Code Length in Dwords.	
7:0	Tx Code Length	0x0	Tx Code Length in Dwords.	

6.5.6.3 Download Data — Offset 0x2 – Data Size

Bits	Name	Default	Description	Reserved
15:8	Download Data	0x0	Download Data	

6.5.7 NC-SI Configuration Module (Global MNG Offset 0x0A)

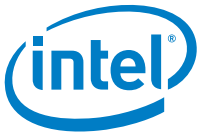
This module is pointed to by global offset 0x0A of the manageability control table.

6.5.7.1 Section Header — Offset 0x0

Bits	Name	Default	Description	Reserved
15:8	Block CRC8	0x0		
7:0	Block Length	0x0	Section length in words.	

6.5.7.2 Rx Mode Control1 (RR_CTRL[15:0]) - Offset 0x1

Bits	Name	Default	Description	Reserved
15:8	Reserved		Set to 0x0.	
7:4	Reserved		Reserved	
3	NC-SI Speed		When set, the NC-SI MAC speed is 100 Mb/s. When reset, NC-SI MAC speed is 10 Mb/s.	
2	Receive Without Leading Zeros		If set, packets without leading zeros (J/K/ symbols) between TXEN assertion and TXD the first preamble byte can be received.	
1	Clear Rx Error		Should be set when the Rx path is stuck because of an overflow condition.	
0	NC-SI Loopback Enable		When set, enables NC-SI TX to RX loop. All data that is transmitted from NC-SI is returned to it. No data is actually transmitted from NC-SI.	



6.5.7.3 Rx Mode Control2 (RR_CTRL[31:16]) - Offset 0x2

Bits	Name	Default	Description	Reserved
15:0	Reserved	0x0		

6.5.7.4 Tx Mode Control1 (RT_CTRL[15:0]) - Offset 0x3

Bits	Name	Default	Description	Reserved
15:3	Reserved		Set to 0x0.	
2	Transmit With Leading Zeros		When set, sends leading zeros (J/K/ symbols) from CRS_DV assertion to the start of preamble (PHY Mode). When de-asserted, does not send leading zeros (MAC mode).	
1	Clear Tx Error		Should be set when Tx path is stuck because of an underflow condition Cleared by hardware when released.	
0	Enable Tx Pads		When set, the NC-SI TX pads are driving; otherwise, they are isolated.	

6.5.7.5 Tx Mode Control2 (RT_CTRL[31:16]) - Offset 0x4

Bits	Name	Default	Description	Reserved
15:0	Reserved	0x0	Set to 0x0.	

6.5.7.6 MAC Tx Control Reg1 (TxCntrlReg1 (15:0]) - Offset 0x5

Bits	Name	Default	Description	Reserved
15:7	Reserved	0x0	Set to 0x0.	
6	NC-SI_enable		Enable the MAC internal NC-SI mode of operation (disables external NC-SI gasket).	
5	Two_part_deferral		When set, performs the optional two part deferral.	
4	Append_fcs		When set, computes and appends the FCS on Tx frames.	

Bits	Name	Default	Description	Reserved
3	Pad_enable		Pad the TX frames, which are less than the minimum frame size.	
2:1	Reserved		Reserved	
0	Tx_ch_en		Tx Channel Enable This bit can be used to enable the Tx path of the MAC. This bit is for debug only and the recommended way to enable the Tx path is via the RT_UCTL_CTRL.TX_enable bit.	

6.5.7.7 MAC Tx Control Reg2 (TxCntrlReg1 (31:16]) - Offset 0x6

Bits	Name	Default	Description	Reserved
15:0	Reserved		Reserved Should be set to 0b.	

6.5.8 Traffic Type Parameters – (Global MNG Offset 0xB)

Bits	Name	Default	Description	Reserved
15:6	Reserved		Reserved	
5:4	Port 1 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 parameter NVM word (Section 6.5.2) is set.	
3:2	Reserved		Reserved.	
1:0	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 parameter NVM word (Section 6.5.2) is set.	

6.6 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 module (no pointers to sub-sections) and must fit into 512 KB.

It is not required for LOM systems where it is rather stored on the BIOS Flash device.

The module is pointed by the PCIe expansion/option ROM pointer at NVM word address 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 X540 would continue to use the old pointer each time it maps internally accesses to the expansion ROM. Refer to Section 3.4.4.1.

6.7 PHY Module

Refer to Section 3.6.3.3 for the way this module is used by the X540.

The module is pointed by the PHY module pointer at NVM word address 0x000004 (or word address 0x000804), expressed in 4 KB sector units. Before performing auto-load, the PHY is reading this pointer directly from the NVM.

The PHY module is composed of three sections as shown in Figure 6-3:

1. Header: Provides configuration information for the SPI interface and pointers to the Instruction RAM and Data RAM locations. (See Table 6-8 for detailed description of the header content).
2. Instruction RAM: This section of the image is loaded into the instructional SRAM (ISRAM).
3. Data RAM: This section of the image is loaded into the data SRAM (DSRAM).

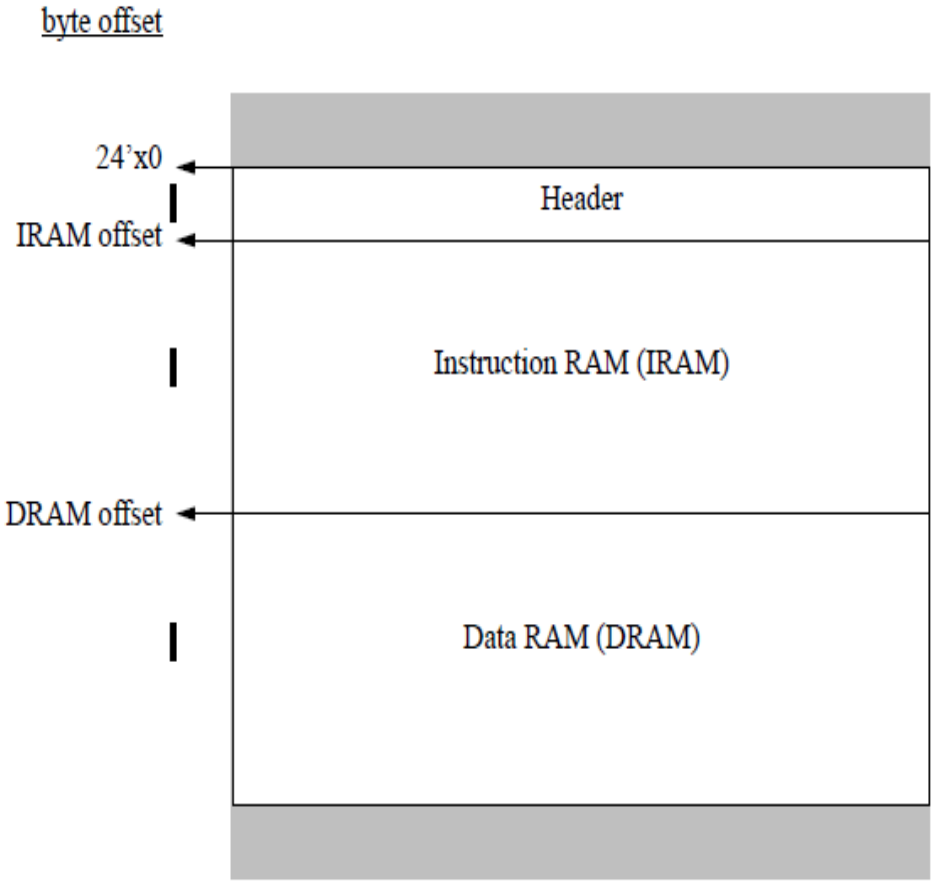


Figure 6-3 PHY Module Map

Table 6-8 PHY Module Header

Byte Offset	Description
0x00	SPI Clock Divider[7:0]
0x01	Daisy Chain Clock Divider[7:0]
0x02	Wait Timer[7:0]
0x03	Wait Timer[15:8]
0x04	MCP IRAM Head Pointer[7:0]
0x05	MCP IRAM Head Pointer[15:8]
0x06	MCP IRAM Head Pointer[23:16]
0x07	MCP IRAM Byte Length[7:0]
0x08	MCP IRAM Byte Length[15:8]
0x09	MCP IRAM Byte Length[23:16]
0x0A	MCP DRAM Head Pointer[7:0]
0x0B	MCP DRAM Head Pointer[15:8]
0x0C	MCP DRAM Head Pointer[23:16]
0x0D	MCP DRAM Byte Length[7:0]
0x0E	MCP DRAM Byte Length[15:8]
0x0F	MCP DRAM Byte Length[23:16]
0x10	{Timeout Configuration[3:0], Reserved[2:0], Enabled CRC}
0x11	Reserved
0x12	Reserved
0x13	Reserved
0x14	Reserved
0x15	Reserved
0x16	Expected CRC16[7:0]
0x17	Expected CRC16[15:8]

The PHY module is organized as follow, where offsets are expressed in bytes from the beginning of the module:

- 0x00000000 - 0x000001BF: 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.

- 0x000001C0 - 0x000002BF: 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.

- 0x000002C0 - 0x000002FF: Version String (62 bytes), Firmware Major/Minor revision numbers (2 bytes).

- 0x00000300 - 0x0003FFFF: IRAM/DRAM load tables and contents.

  The boot code uses the load table and data in this region to initialize IRAM and DRAM.

6.7.1 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 10.0 are affected.

This page intentionally left blank.

7.0 Inline Functions

7.1 Receive Functionality

Packet reception consists of:

- Recognizing the presence of a packet on the wire
- Performing address filtering
- DMA queue assignment
- Storing the packet in the receive data FIFO
- Transferring the data to assigned receive queues in host memory
- Updating the state of a receive descriptor.

A received packet goes through 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), an IOVM, a VMM, or the like. The three stages are:

1. First stage — 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 Section 7.1.1.
2. Second stage — 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 7.1.2.2.
3. Third stage — A receive packet that successfully passed the Rx filters is associated with one of many receive descriptor queues as described in this section.

In addition to the filtering rules, a packet must also meet the following criteria:

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-packet bit is set (FCTRL.SBP), then 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-packet bit.
2. Min Packet Size (Runt packets) — Rx packets, smaller than 21 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. Furthermore, Rx packets are posted to system memory only if their length does not exceed a certain length, defined per queue in the RXDCTL[n].RLPML field enabled by the RXDCTL[n].RLPML_EN. This filter enables software to use smaller buffers than the size defined by the SRRCTL[n].BSIZEPACKET.

Note: CRC errors before the Start Frame Delimiter (SFD) are ignored. All packets must have a valid SFD in order to be recognized by the device (even bad packets).

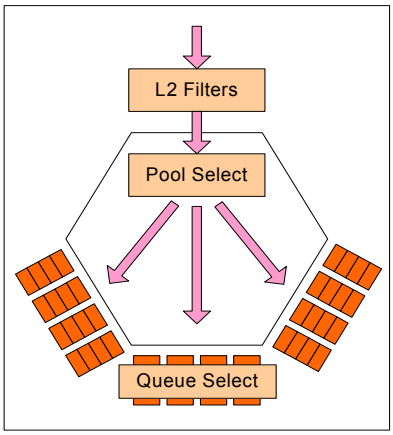


Figure 7-1 Stages in Packet Filtering

7.1.1 Packet Filtering

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 three stages:

1. Packets are filtered by L2 filters (Ethernet MAC address, unicast/multicast/broadcast). See Section 7.1.1.1.
2. Packets are filtered by VLAN if a VLAN tag is present. See Section 7.1.1.2.
3. Packets are filtered by the manageability filters (port, IP, flex, other). See Section 11.3.

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 7.1.1.1.
- The packet does not pass VLAN filtering, as described in Section 7.1.1.2.
- The packet passes manageability filtering and the manageability filters determine that the packet should not pass to the host as well (see MANC2H register).

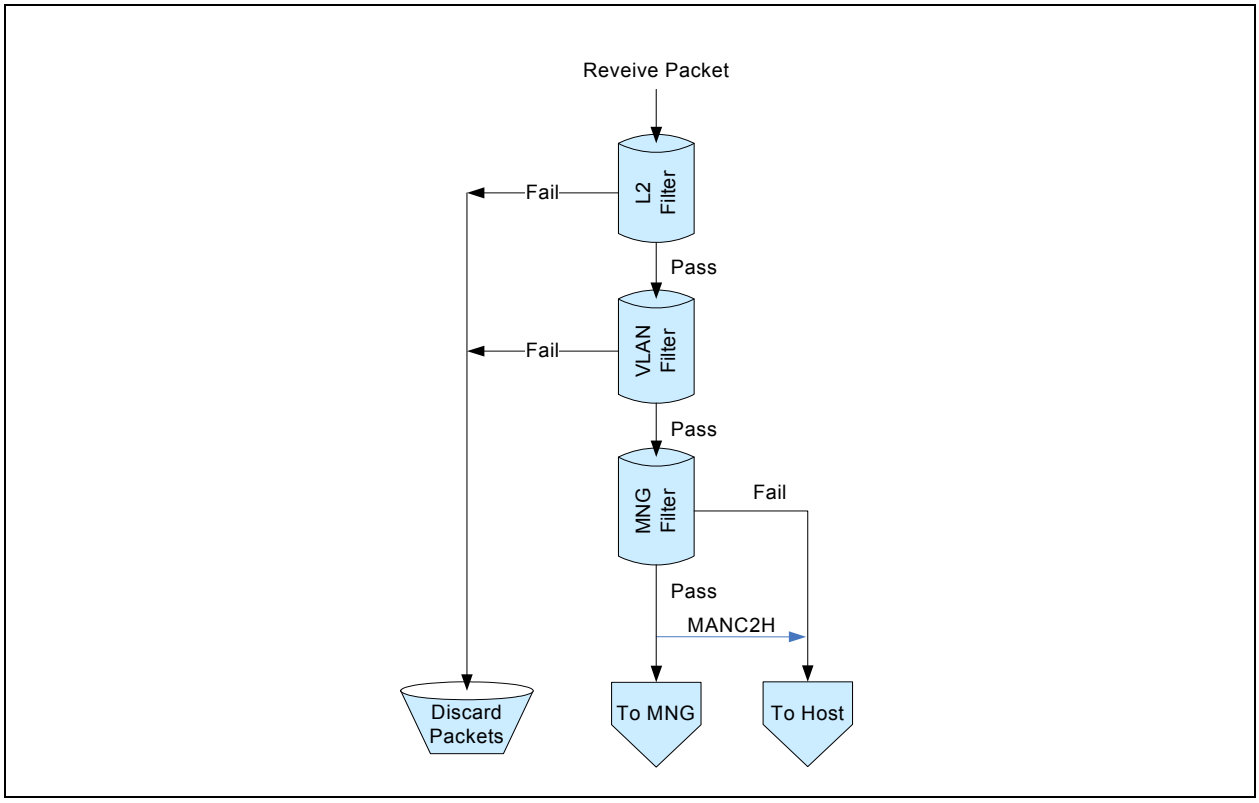


Figure 7-2 Rx Filtering Flow Chart

7.1.1.1 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 or manageability).
- Multicast packet filtering — Promiscuous multicast filtering is enabled by either the host or manageability (FCTRL.MPE=1b or MANC.MCST_PASS_L2 =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).
- Broadcast packet filtering to manageability — Always enabled depending on the MDEF filters.

7.1.1.1.1 Unicast Filter

The Ethernet MAC address is checked against the 128 host unicast addresses, 4 KB hash-based unicast address filters and four management unicast addresses (if enabled). The host unicast addresses are controlled by the host interface (the manageability controller

must not change them). The other four addresses are dedicated to management functions and are only accessed by the manageability. The destination address of an incoming packet must exactly match one of the pre-configured host address filters or the manageability address filters. These addresses can be unicast or multicast. Those filters are configured through Receive Address Low (RAL), Receive Address High (RAH), Manageability Ethernet MAC Address Low (MMAL) and Manageability Ethernet MAC Address High (MMAH) registers. 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 can be set/cleared only through the host interface (not by the manageability controller) and it is usually used when the LAN device is used as a sniffer.

7.1.1.1.2 Multicast Filter (Partial)

The 12-bit portion of the incoming packet multicast address must exactly match the multicast filter address in order to pass multicast filter. These bits (out of 48 bits of the destination address) can be selected by the *MO* field in the MCSTCTRL register. The entries can be configured only by the host interface and cannot be controlled by the manageability controller.

Promiscuous Multicast — Receive all multicasts. Promiscuous multicast mode can be set/cleared only through the host interface (not by the manageability controller) and it is usually used when the LAN device is used as a sniffer.

7.1.1.2 VLAN Filtering

The X540 provides exact VLAN filtering for host traffic and manageability traffic, as follows:

- Host VLAN filters are programmed by the VFTA[n] registers.
- Manageability VLAN filters are activated by the MDEF filters. One of eight VLAN tags are programmed by the MAVTV[7:0] registers while enabled by the MFVAL register.
- A VLAN match might relate to the *CFI* bit in the VLAN header. It is enabled for host filtering only by the VLNCTRL.CFIEN while the expected value is defined by the VLNCTRL.CFI.

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 X540 ignore the first (external) VLAN in this mode.

A receive packet that passes L2 layer 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 the packet is broadcast and MANC.RCV_TCO_EN bit is set, then it passes to the next filtering stage.
3. Else, if the packet passes a valid manageability VLAN filter and at least one VLAN_AND bit is set in the MDEF[n] registers, then it passes to the next filtering stage.

4. Else, if host VLAN filters are not enabled (VLNCTRL.VFE = 0b), the packet is forwarded to the next filtering stage.
5. Else, if the packet matches an enabled host VLAN filter and CFI checking (if enabled), the packet is forwarded to the next filtering stage.
6. Else, if manageability VLAN filtering is not required (MANC.Bypass_VLAN is set), the packet is forwarded to the next filtering stage as a potential candidate only for manageability.
7. Otherwise, the packet is dropped.

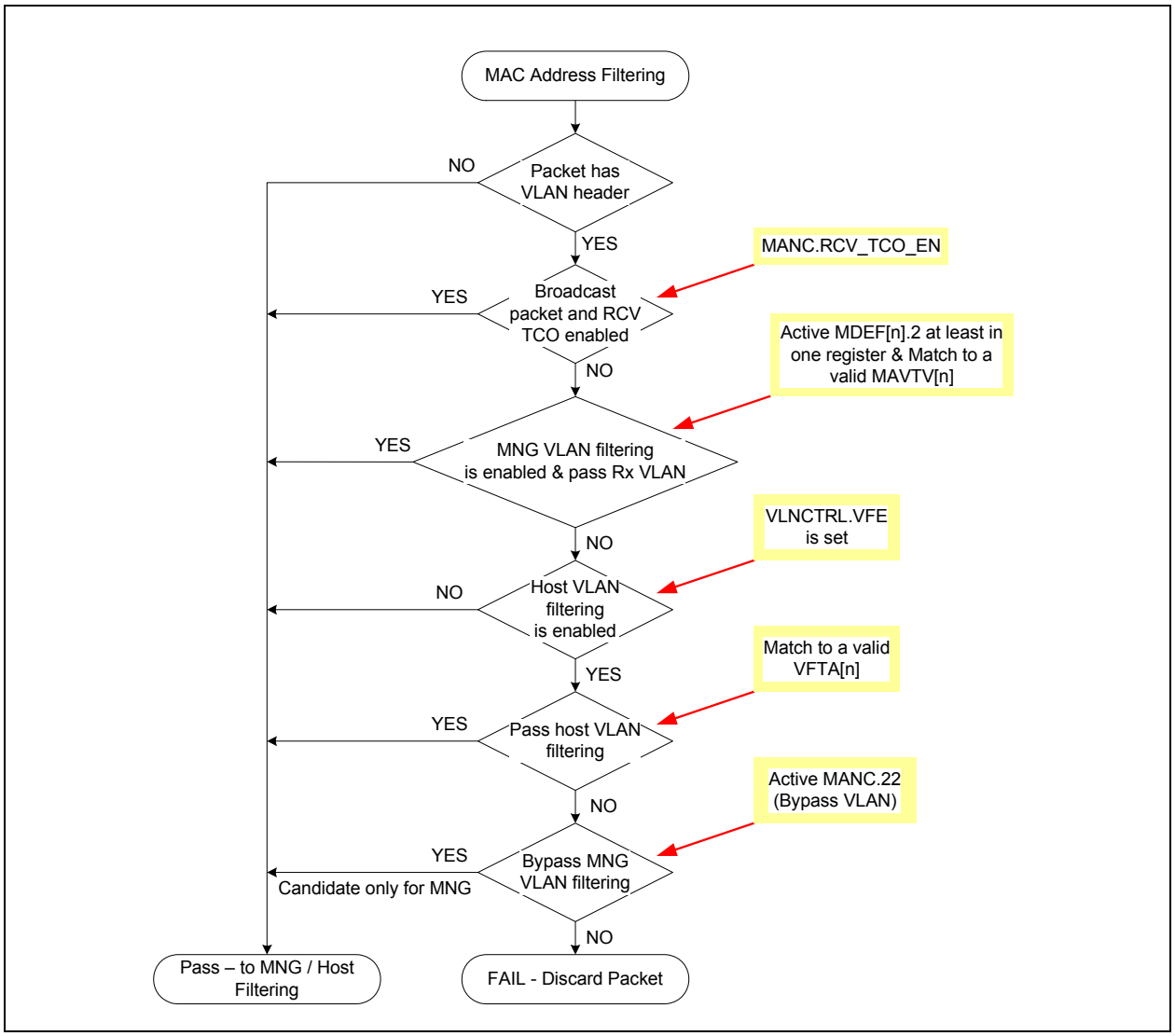


Figure 7-3 VLAN Filtering

7.1.1.3 Manageability / Host Filtering

Packets that pass the MAC address filters and VLAN address filters described in the previous sections are subjected to MNG / Host filtering shown in Figure 7-4. The manageability filters are described in Section 11.3. Packets that are not accepted for manageability automatically become candidates for the host queue filters described in Section 7.1.2. Packets that pass the manageability filters may still be posted to the host as well if they match the BMC-to-host filters defined by the MANC2H register.

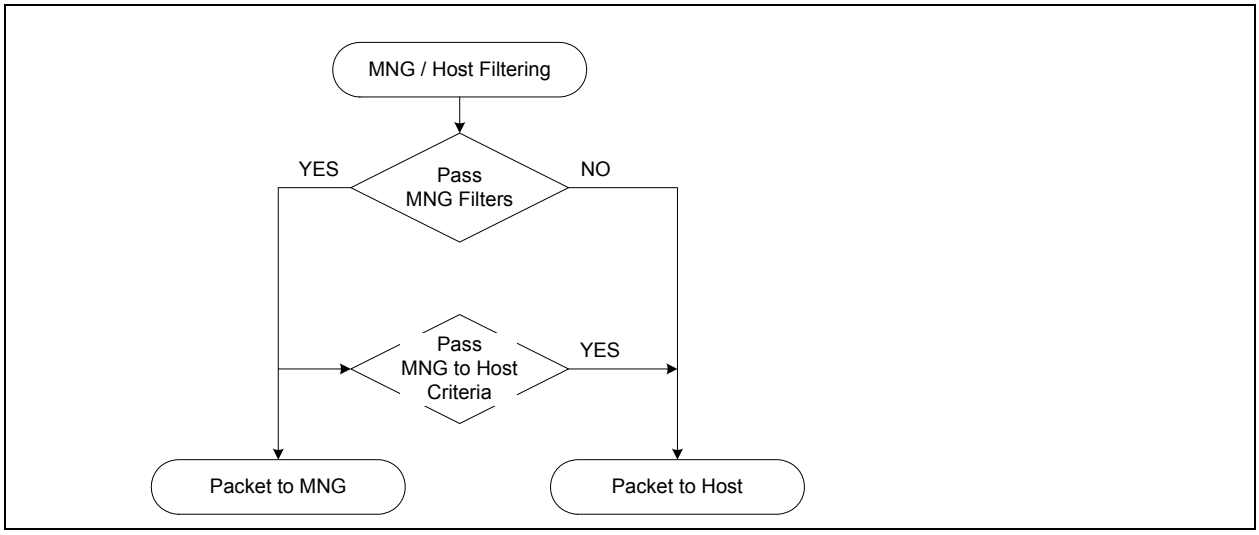


Figure 7-4 Manageability / Host Filtering

7.1.2 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 7.1.2.2 for more on routing for virtualization.
- DCB — 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 7.7 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 of a set of Rx queues based on their RSS index and other considerations such as DCB and virtualization. See Section 7.1.2.8 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 7.1.2.3 for details. The X540 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 7.1.2.4 for details. See also Section 7.13.3.3 for Large FC receive and direct data placement.
- L3/L4 5-tuple Filters — These filters identify specific L3/L4 flows or sets of L3/L4 flows. Each filter consists of a 5-tuple (protocol, source and destination IP addresses, source and destination TCP/UDP port) and routes packets into one of the Rx queues. The X540 incorporates 128 such filters. See Section 7.1.2.5 for details.
- Flow Director Filters — These filters are an expansion of the L3/L4 5-tuple filters that provide up to additional 32 K filters. See Section 7.1.2.7 for details.
- TCP SYN Filters — The X540 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 7.1.2.6 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)
- If SYNQF.SYNQFP is zero, then
  - Queue by L3/L4 5-tuple filters (if match)
  - Queue by SYN filter (if match)
- If SYNQF.SYNQFP is one, then
  - Queue by SYN filter (if match)
  - Queue by L3/L4 5-tuple filters (if match)
- Queue by flow director filters
- Define a pool (in case of virtualization)
- Queue by DCB and/or RSS as described in Section 7.1.2.1 and Section 7.1.2.2.

7.1.2.1 Queuing in a Non-virtualized Environment

Table 7-1 lists the queuing schemes. Table 7-2 illustrates the queue indexing. Selecting a scheme is done via the *Multiple Receive Queues Enable* field in the MRQ 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	16 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 16 RSS	Supported

1. Special filters include: L2 filters; FCoE redirection; SYN filter and L3/L4 5-tuple filters. 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

Queue Index bits	6	5	4	3	2	1	0
RSS	0	0	0	RSS			
DCB(4) + RSS	TC		0	RSS			
DCB(8) + RSS	TC			RSS			

A received packet is assigned to a queue according to the ordering shown in Figure 7-6).

- DCB and RSS filters and FCoE redirection — Packets that do not meet any of the previous filtering conditions described in Section 7.1.2 are assigned to one of 128 queues as listed in Table 7-1. The following modes are supported:
  - — No DCB, No RSS and No FCoE redirection — Queue 0 is used for all packets.
  - — RSS only — A set of 16 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 one of 64 queues (4 TCs x 16 RSS) through the DCB traffic class of the packet and the RSS index. The TC index is used as the MS bit of the Rx queue index, and the LSBits are defined by the RSS index.
  - — FCoE redirection — Up to eight queues can be allocated for FCoE traffic by the FCoE redirection table defined by FCRETA[n] registers.

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 (0/1/2/4 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 an example of allocation of Rx queues by the various queue filters previously described for the following case:

- DCB and RSS enabled to 4 TCs x 16 RSS queues
- RSS is used at various width per TC
- SYN filter allocated
- Ethertype filters are used
- 5-tuple filters are used

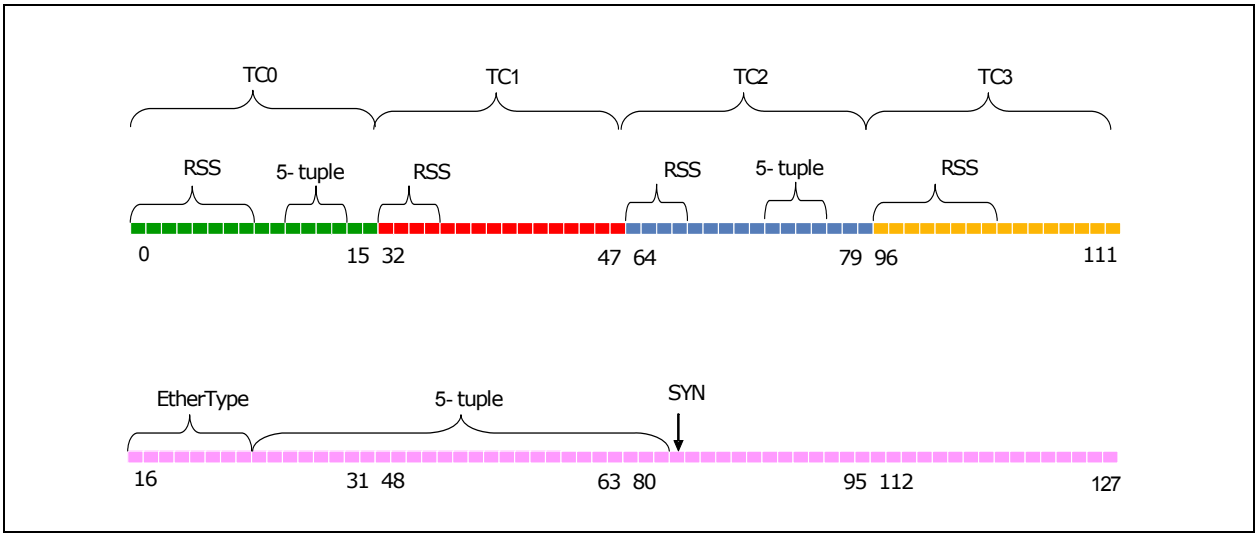


Figure 7-5 Example of Rx Queue Allocation (Non-virtualized)

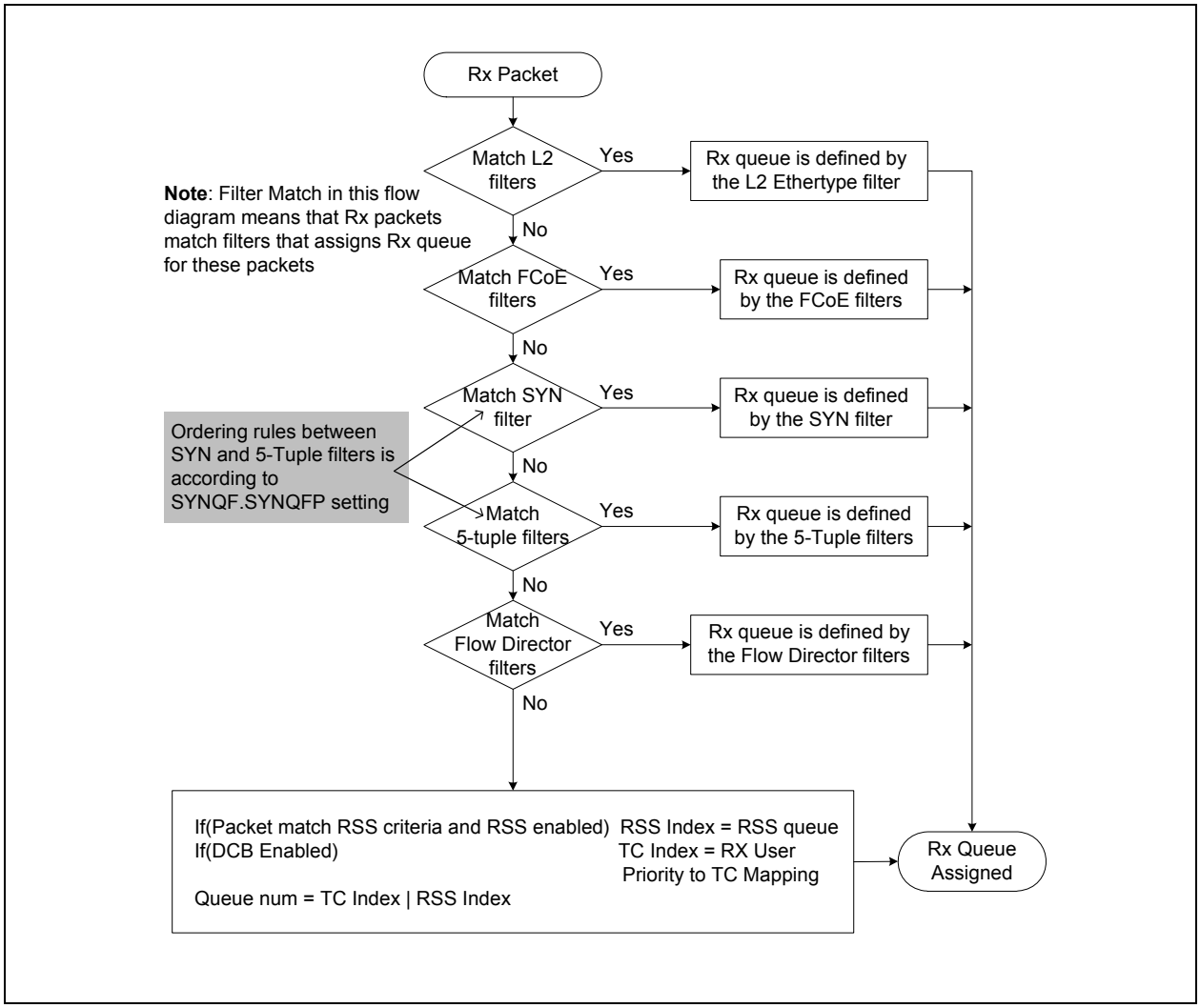


Figure 7-6 Rx Queuing Flow (Non-virtualized)

7.1.2.2 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, an IOVM, or to VMs. In IOV mode, each pool is associated with a VF.

Incoming packets are associated with pools based on their L2 characteristics as described in Section 7.10.3. This section describes the following stage, where an Rx queue is assigned to each replication of the Rx packet as determined by its pools association.

Table 7-3 lists the queuing schemes supported with virtualization. Table 7-4 lists the queue indexing.

Table 7-3 Rx Queuing Schemes Supported with Virtualization

DCB	RSS	DCB / RSS Queues	Special Filters (2)
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 (1)	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	Not supported	

Notes:

(1) RSS might not be useful for IOV mode since the X540 supports a single RSS table for the entire device.

(2) Special filters include: L2 filters; FCoE redirection; SYN filter and L3/L4 5-tuple filters. 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

Queue Index bits	6	5	4	3	2	1	0
VT(64) + RSS	VF Index						RSS
VT(32) + RSS	VF Index					RSS	
VT(16) + RSS	Not Supported						
VT(32) + DCB(4)	VF Index					TC	
VT(16) + DCB(8)	VF Index				TC		

Selecting a scheme is done in the following manner:

- Non-IOV mode
  - Selected 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 9.4.4.5). 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.
  - Note that RSS is not supported in IOV mode since there is only a single RSS hash function in the hardware.

A received packet is assigned to an absolute queue index according to the ordering shown in Figure 7-7). 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 allocated per 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'.
  - RSS only — All 128 queues are allocated to pools. TwoConfigurations are supported: 32 pools with 4 RSS queues each and 64 pools with 2 queues each. Note that 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.
  - 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.

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 (0/1/2/4- 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.

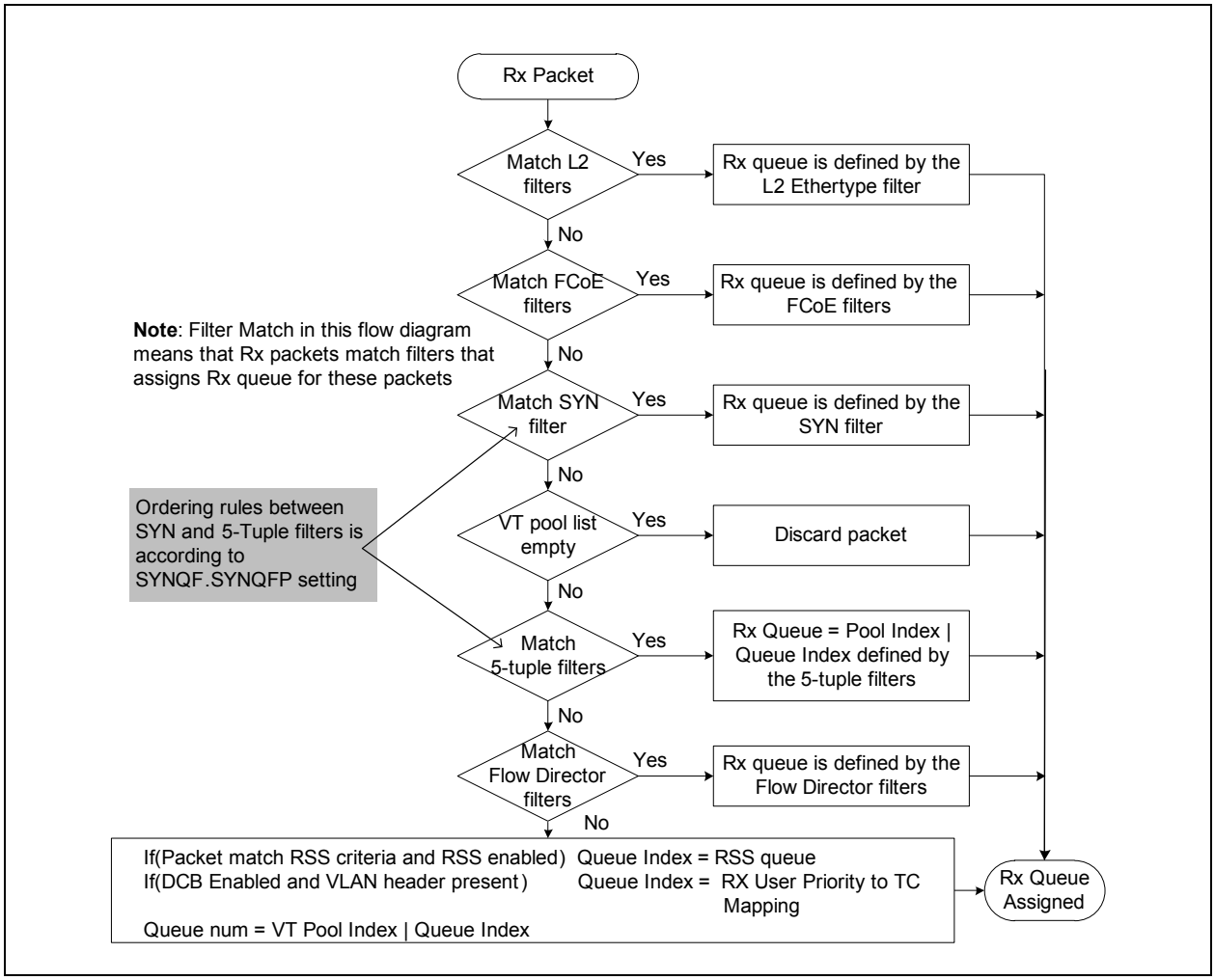


Figure 7-7 Rx Queuing Flow (Virtualization Case)

7.1.2.3 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 X540 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 *Ether-Type* 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 7.1.6.2 for a description of the *Packet Type* field.

The following flow is used by the Ethertype filters:

1. If the *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 might 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 MANC2H register setting.
4. If the *FCoE* field is set, the packet is identified as an FCoE packet.
5. If the *1588 Time Stamp* field is set, the packet is identified as an IEEE 1588 packet.
6. If the *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 *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 7.10.3. See the sections that follow for more details on the use of the *Pool* field.
8. The *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 *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 then 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.
- The *Low Latency Interrupt* bit enables LL interrupt assertion by the Rx packet that matches this filter.

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 a candidate for mirrorring using the standard mirroring rules.
  - The Ethertype filter does not take part in the decision on the destination of the mirrored packet.

7.1.2.4 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 7-8 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 three LS bits of the EX_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.

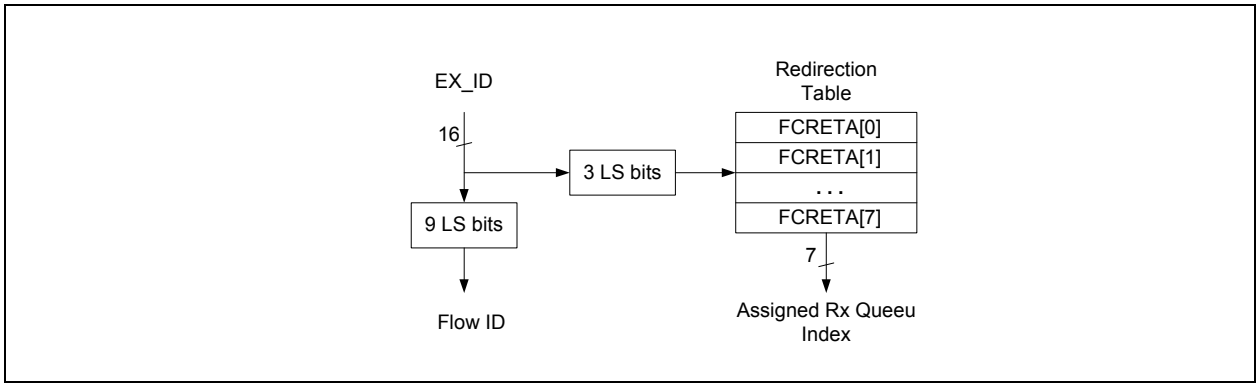


Figure 7-8 FCoE Redirection Table

7.1.2.5 L3/L4 5-tuple Filters

These filters identify specific L3/L4 flows or sets of L3/L4 flows and routes them to dedicated queues. Each filter consists of a 5-tuple (protocol, source and destination IP addresses, source and destination TCP/UDP/SCTP port) and routes packets into one of the Rx queues.

The X540 incorporates 128 such filters, used also to initiate low latency interrupts. The specific filtering rules are:

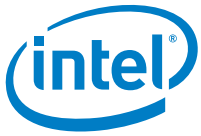
- Filtering rules for IPv6 packets:
  - If a filter defines at least one of the IP source and destination addresses, then an IPv6 packet always misses such a filter.
  - If a filter masks both the IP source and destination addresses, then an IPv6 packet is compared against the remaining fields of the filter.
- Packets with tunneling (any combination of IPv4 and IPv6) miss the 5-tuple filters.
- Fragmented packets miss the 5-tuple filters.

In a virtualized environment, any 5-tuple filters is associated with a unique pool:

- The packet must first match the L2 filters described in Section 7.10.3.3 and Section 7.10.3.4. The outcome of the L2 filters is a set of pool values associated with the packet. The *Pool* field of the 5-tuple filter is then compared against the set of pools to which the packet is steered. A filter match is considered only to the indicated pool in the filter.
- The queue index of the filters must be defined within the pool associated with the packet.

If a packet matches more than one 5-tuple filter, then:

- For queuing decision — The priority field identifies the winning filter and therefore the destination queue.
- For queuing decision — If the packet matches multiple filters with the same priority, the filters with the lower index takes affect.



- For Low Latency Interrupt (LLI) — An LLI is issued if one or more of the matching filters are set for LLI.

The 5-tuple filters are configured via the FTQF, SDPQF, L34TIMIR, DAQF, and SAQF registers, as follows (described by filter):

- Protocol — Identifies the IP protocol, part of the 5-tuple. Enabled by a bit in the mask field. Supported protocol fields are TCP, UDP, SCTP or other (neither TCP nor UDP nor SCTP).
- Source address — Identifies the IP source address, part of the 5-tuple. Enabled by a bit in the mask field. Only IPv4 addresses are supported.
- Destination address — Identifies the IP destination address, part of the 5-tuple. Enabled by a bit in the mask field. Only IPv4 addresses are supported.
- Source port — Identifies the TCP/UDP/SCTP source port, part of the 5-tuple. Enabled by a bit in the mask field.
- Destination port — Identifies the TCP/UDP/SCTP destination port, part of the 5-tuple queue filters. Enabled by a bit in the mask field.
- Queue Enable — Enables the packets routing to queues based on the Rx Queue index of the filter.
- Rx Queue — Determines the Rx queue for packets that match this filter.
- Pool — Applies only in the virtualized case (while *Pool Mask* bit = 0b). This field must match one of the pools enabled for this packet in the L2 filters.
  - In non-virtualized case the *Pool Mask* bit must be set to 1b.
  - In the virtualized case, the pool must be defined (*Pool Mask* = 0b and *Pool* = valid index). The *Rx Queue* field defines the absolute queue index. In case of mirroring or replication, only the copy of the packet destined to the matched pool in the filter is routed according to the *Rx Queue* field.
- Mask — A 5-bit field that masks each of the fields in the 5-tuple (L4 protocol, IP addresses, TCP/UDP ports). The filter is a logical AND of the non-masked 5-tuple fields. If all 5-tuple fields are masked, the filter is not used for queue routing.
- Priority — A 3-bit field that defines one of seven priority levels (001b-111b), with 111b as the highest priority. Software must insure that a packet never matches two or more filters with the same priority value.

7.1.2.6 SYN Packet Filters

The X540 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.

7.1.2.7 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 X540 shares the Rx packet buffer for the storage of these filters. Basic rules for the flow director filters are:

- IP packets are candidates for the flow director filters (meaning non-IP packets miss all filters)
- Packets with tunneling (any combination of IPv4 and IPv6) miss all filters
- Fragmented packets miss all filters
- 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:

- VLAN header
- Source IP and destination IP addresses
- Source port and destination port numbers (for UDP and TCP packets)
- IPv4 / IPv6 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)

Note: IPv6 extended headers are parsed by the X540, enabling TCP layer header recognition. As such, the IPv6 extended header fields are not taken into account for the queue classification by RSS filter. This rule does not apply for security headers and fragmentation headers. Packets with fragmentation headers 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 X540 support 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 X540 support up to 8 K -2 perfect match filters.
- Signature filters — The hardware checks a match between a hash-based signature of the masked fields of the received packet. The X540 supports up to 32 K -2 signature filters.

The *Perfect Match* fields and *Signature* field are denoted as *Flow ID* fields.

The X540 supports masking / range for the previously described fields. These masks are defined globally for all filters in the FDIR…M register.

- 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. Note that in perfect match filters the destination IPv6 address can only be compared as a whole (with no range support) to the IP6AT.
- The following fields can be either enabled or disabled completely for the match functionality: VLAN ID tag; VLAN Priority + CFI bit; 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 7.1.2.7.11 and Section 7.1.2.7.12.

7.1.2.7.1 Flow Director Filters Actions

Flow director filters might have one of the following actions programmed per filter in the FDIRCTRL 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 nor the 5-tuple filters for queue assignment.
  - Packets that match a filter are directed to the Rx queue defined in the filter context as programmed by the FDIRCTRL.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 setting, 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 X540 drops these packets if software does not enable the specific Rx queue.
- Trigger low latency interrupt is enabled by the *INT* bit.
  - Matched packets to a flow director filter generates LLI if the packet does not match the L2 filters for queue assignment nor the SYN filter for queue assignment nor the 5-tuple filters for queue assignment.

7.1.2.7.2 Flow Director Filters Status Reporting

Shared status indications for all packets:

- The X540 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, the *Flow Director Filter ID* field can be used by software for future programming of a matched filter.

— 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 7-9):

- 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 already at 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.Rx-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. On the other hand, software might choose to resolve the collision by programming the collided flows in the 5-tuples filters. Only one flow (out of the collided ones) might remain in the flow director filters. In order 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.

7.1.2.7.3 Flow Director Filters Block Diagram

The following figure 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 7.1.2.7.5).

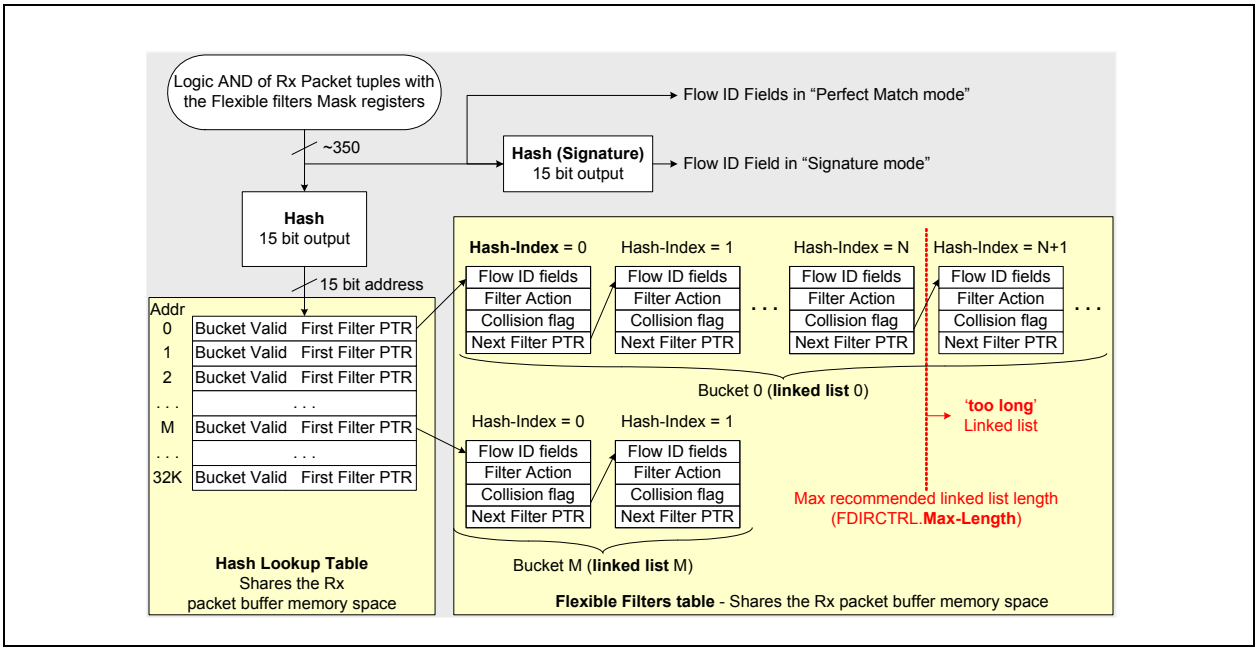


Figure 7-9 Flow Director Filters Block Diagram

7.1.2.7.4 Rx Packet Buffer Allocation

Flow director filters can consume zero space (when disabled) up to ~256 KB of memory. As shown in Figure 7-9, flow director filters share the same memory with the Rx packet buffer. Setting the *PBALLOC* field in the FDIRCTRL register, the software might enable and allocate memory for the flow director filters. The memory allocated to reception is the remaining part of the Rx packet buffer.

Table 7-5 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
00b Flow Director is disabled	384 KB	0	0	N/A	0	N/A
01b	320 KB	64 KB	8 K - 2 filters	13 bit	2 K - 2 filters	11 bit
10b	256 KB	128 KB	16 K - 2 filters	14 bit	4 K - 2 filters	12 bit
11b	128 KB	256 KB	32 K - 2 filters	15 bit	8 K - 2 filters	13 bit

Note: It is the user responsibility to ensure that sufficient buffer space is left for reception. The required buffer space for reception is a function of the number of traffic classes, flow control threshold values and remaining buffer space in between the thresholds. If flow director is enabled (such as

PBALLOC > 0), software should set the RXPBSIZE[n] registers according to the total remaining part of the Rx packet buffer for reception.

When allocating 256 KB to the Flow Director table, it is mostly probable that the 128 KB packet buffer left is too small to permit operating the port in DCB enabled mode.

For example, if PBALLOC equals one and there is only one buffer in the system, software should set RXPBSIZE[0] to 0x140 (320 K) and RXPBSIZE[1...7] to zero. Another example is if PBALLOC equals two and DCB is enabled with four traffic classes then software might set RXPBSIZE[0...3] to 0x40 (64 K) and RXPBSIZE[4...7] to zero. Refer to Section 3.6.5.3.2 through Section 3.6.5.3.5 for recommended setting of the Rx packet buffer sizes and flow control thresholds.

7.1.2.7.5 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 X540 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 7.1.2.7.15.
- The address in the hash lookup table points to the selected linked list of the flow director filters.
- The X540 checks the *Bucket Valid* flag. If it is inactive, then the packet does not match any filter. Otherwise, *Bucket Valid* flag is active, proceed for the next steps.
- The X540 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 and LLI) according to queue assignment priority. Meaning, the actions defined in this filter takes place only of the packet did not match any L2 filters or SYN filter or 5-tuple filter that assign an Rx queue to the packet.
  - Rx queue assignment according to the filter context takes place if *Queue-EN* is set. In VT mode, the Rx queue in the filter context defines a relative queue within the pool.
  - LLI is generated if the *INT* bit is set in the filter context.
  - Post the packet to host including the flow director filter match indications as described in Section 7.1.2.7.2.
- 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 7.1.2.7.2.

7.1.2.7.6 Add Filter Flow

The software programs the filters parameters in the registers described in Section 7.1.2.7.12 and Section 7.1.2.7.13 while keeping the FDIRCMD.Filter-Update bit inactive. As a result, the X540 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 that handle a presence of an existing linked list or creating a new linked list.

- Case 1: Add a filter to existing linked list:

  The X540 checks the linked list until it reaches the last filter in the list or until a matched filter is found. Handle the filter programming in one of the following cases:

  — 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 X540 keeps the old setting of the filter while setting the *Collision* flag in the filter context and increments the COLL counter in the FDIRFREE register (see Section 7.1.2.7.2 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 X540 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 X540 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, then 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.

Note: The X540 also reports the number of collided filters in FDIRFREE.COLL. Software might monitor this field periodically as an indication for the filters efficiency.

7.1.2.7.7 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 7.1.2.7.6.

Setting the *Filter-Update* bit in the FDIRCMD register has the following action:

- 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.

7.1.2.7.8 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 X540 follows these steps:

- Check if such a filter exists in the flow director filters table.
- If there is no flow, then increment the FREMOVE counter in the FDIRFSTAT register and skip the next steps.
- If the requested filter is the only filter in the linked list, then invalidate its entry in the hash lookup table by clearing the *Valid* bit.
- Else, if the requested filter is the last filter in the linked list, then invalidate the entry by setting the *Last* flag in the previous filter in the linked list.
- 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:

- Increment the remove statistic counter in the FDIRUSTAT register.
- Increment the FREE counter in the FDIRFREE register.

7.1.2.7.9 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 both *FREE* and *COLL* fields 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).

7.1.2.7.10 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 7.1.2.7.4). All other fields in the register should be valid as well (according to required setting) while the FDIRCTRL register is expected to be programmed by a single cycle. Any further programming of the FDIRCTRL register with non-zero value PBALLOC initializes the flow director table once again.
- 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 (it is 5.5 ms at 100 M/ps; however, this speed is not expected to be activated unless the X540 is in a sleep state).

7.1.2.7.11 Query Filter Flow

Software might query specific filter settings and bucket length using the Query command.

- Program the filter Hash and Signature/Software-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 X540 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.

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

Query Outcome	FDIRHASH -> Bucket Valid	FDIRCMD -> Filter Valid	FDIRLEN -> Bucket Length	FDIRCMD -> Filter ID Fields	FDIRCMD -> Filter Action
Found Perfect Match Filter	1	1	Filter index within the linked list	Filter's parameters	Filter's parameters

7.1.2.7.12 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 7-6 lists the recommended setting.

Table 7-6 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 as perfect match filters.
Valid	Should be set to 1b. It is shared for signature as perfect match filters.
Flow ID — FDIRHASH	
Signature	16-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 the FDIRCMD register section for all fields descriptions.	

7.1.2.7.13 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 7-7 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 7-7 Perfect Match Filter Parameters

Filter Bucket Parameters and Software Index — FDIRHASH	
Hash	See Section 7.1.2.7.12.
Valid	See Section 7.1.2.7.12.
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.

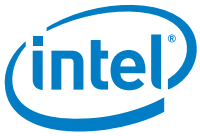


Table 7-7 Perfect Match Filter Parameters

FDIRCMD — Programming Command and Filter Action Set in the FDIRCMD register section for All Fields Descriptions	
Flow ID — Perfect Match Flow ID Parameters are Listed in the Following Registers and Fields	
FDIRSIPv6[0…2].IP6SA	Three MS DWord of the source IPv6. Meaningful for IPv6 flows depending on the FDIRIP6M.SIPM setting.
FDIRVLAN.VLAN	VLAN fields are meaningful depending on the FDIRM.VLANID and FDIRM.VLANP setting.
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 and UDP packets depending on the FDIRTCPM.SportM and FDIRUDPM.SportM setting.
FDIRPORT.Destination	L4 destination port. Meaningful for TCP and UDP packets depending on the FDIRTCPM.SportM and FDIRUDPM.SportM setting.
FDIRIPDA.IP4DA	IPv4 destination address. Meaningful depending on the FDIRDIP4M.IP-EN setting.
FDIRIPSA.IP4SA	IPv4 source address or LS DWord of the source IPv6 address. Meaningful for IPv4 flows depending on the FDIRSIP4M.IP-EN setting and for IPv6 flows depending on the FDIRIP6M.SIPM setting.

7.1.2.7.14 Multiple CPU Cores Considerations

Perfect match filters programming and any query cycles requires access to multiple registers. In order 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.

7.1.2.7.15 Flow Director Hash Function

The X540 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 follow. 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 7-8 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 the Table 7-8 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 7-8 Input Bit Stream for Hash Calculation

Bytes	Field
Bytes 0…1	VLAN tag
Bytes 2…17	Source IP (16 bytes for IPv6; 12 bytes of zero's | source IP for IPv4)
Bytes 18…33	Destination IP (16 bytes for IPv6; 12 bytes of zero's | source IP for IPv4)
34...37	L4 source port number | L4 destination port number Meaningful for TCP and UDP packets and zero bytes for SCTP packets
38...39	Flexible bytes
40	00b | pool number (as defined by FDIRCMD.Pool)
41	00000b | IPv6/IPv4 type | L4 type (as defined by FDIRCMD.IPV6 and FDIRCMD.L4TYPE, respectively)

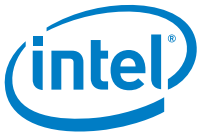
7.1.2.8 RSS

RSS 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.

As described in Section 7.1, the X540 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 X540 global assignment uses these bits (or only some of the LSBs) as part of the queue number.

Figure 7-10 shows 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 X540 supports a single hash function, as defined by Microsoft* (MSFT) RSS. The X540 therefore does not indicate to the device driver which hash function is used. The 32-bit result is fed into the packet receive descriptor.
3. The seven LSBs of the hash result are used as an index into a 128-entry redirection table. Each entry provides a 4-bit RSS output index.



When RSS is enabled, the X540 provides software with the following information as:

1. Required by MSFT RSS
2. Provided for device driver assist:
   - A Dword result of the MSFT RSS hash function, to be used by the stack for flow classification, is written into the receive packet descriptor (required by MSFT RSS).
   - A 4-bit RSS *Type* field conveys the hash function used for the specific packet (required by MSFT RSS).

Enabling rules:

- RSS is enabled in the MRQC register.
- RSS enabling cannot be done dynamically while it 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.

Disabling rules:

- Disabling RSS on the fly is not allowed, and the X540 must be reset after RSS is disabled.
- When RSS is disabled, packets are assigned an RSS output index = zero.

When multiple request queues are enabled in RSS mode, un-decodable packets are assigned an RSS output index = zero. The 32-bit tag (normally a result of the hash function) equals zero.

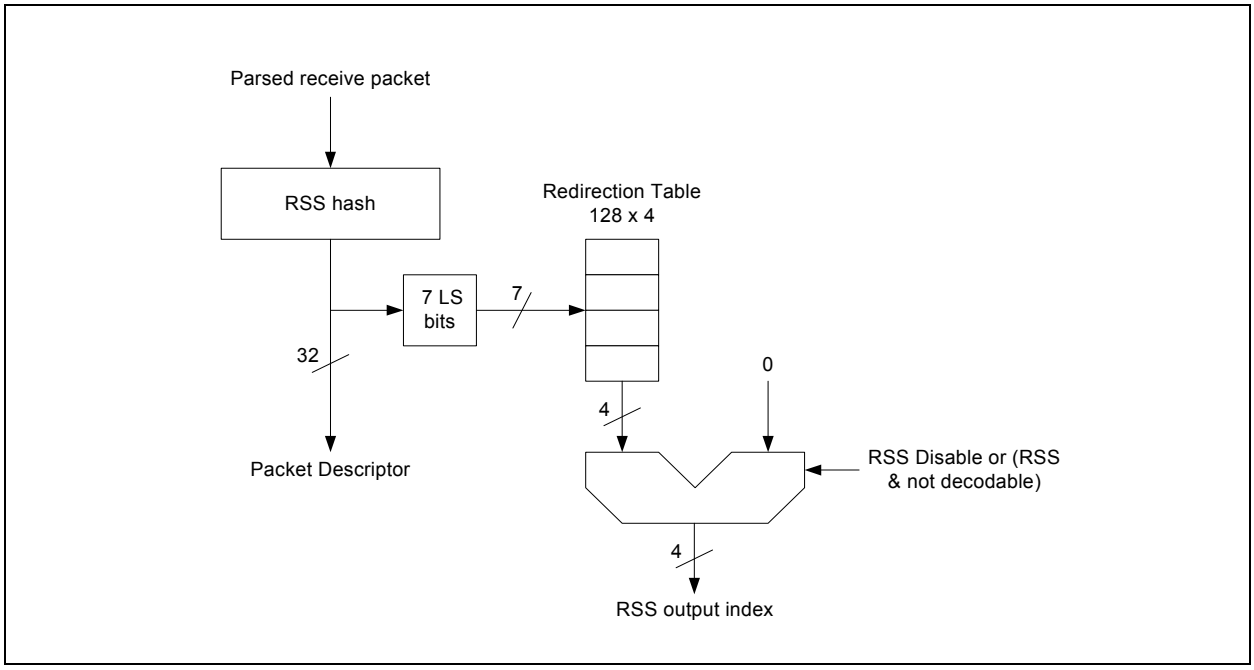


Figure 7-10 RSS Block Diagram

7.1.2.8.1 RSS Hash Function

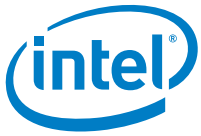
This section provides a verification suite used to validate that the hash function is computed according to MSFT nomenclature.

The X540's hash function follows the MSFT definition. A single hash function is defined with several variations for the following cases:

- TcpIPv4 — The X540 parses the packet to identify an IPv4 packet containing a TCP segment per the following criteria. If the packet is not an IPv4 packet containing a TCP segment, RSS is not done for the packet.
- IPv4 — The X540 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 X540 parses the packet to identify an IPv6 packet containing a TCP segment per the following criteria. If the packet is not an IPv6 packet containing a TCP segment, RSS is not done for the packet.
- IPv6 — The X540 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.The following additional cases are not part of the MSFT RSS specification:

- UdpIPV4 — The X540 parses the packet to identify a packet with UDP over IPv4.
- UdpIPV6 — The X540 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 X540, enabling TCP layer header recognition. As such, the IPv6 extended header fields are not taken into account for the queue classification by RSS filter. This rule does not apply for security headers and fragmentation headers. Packets with fragmentation headers 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) register 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).

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;
```

7.1.2.8.1.1 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);
```

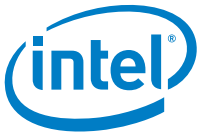
Hash for IPv4 without TCP

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 TCP

```
Input[32] = @8-23, @24-39
Result = ComputeHash(Input, 32)
```

7.1.2.8.2 Redirection Table

The redirection table is a 128-entry structure, indexed by the seven LSBs of the hash function output.

System software might 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.

7.1.2.8.3 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 7-9 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 7-10 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

7.1.3 MAC Layer Offloads

7.1.3.1 CRC Strip

The X540 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 policy for CRC strip is as follows:

- When RSC is enabled on any queue, the global CRC strip bit should be set (HLREG0.RXCRCSTRP = 1b).
- When either MACsec or IPsec are enabled, the global CRC strip bit should be set (HLREG0.RXCRCSTRP= 1b), since the payload of the packet changes and the CRC value is stale due to it.

7.1.4 Receive Data Storage in System Memory

The X540 posts receive packets into data buffers in system memory.

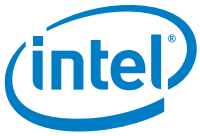
The following controls are provided for the data buffers:

- The SRRCTL[n].BSIZEPACKET field defines the data buffer size. See section Section 7.1.2 for packet filtering by size.
- The SRRCTL.BSIZEHEADER field defines the size of the buffers allocated to headers (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).

7.1.5 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 X540 uses the Legacy Rx descriptor as listed in Table 7-11. 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, MACsec, IPSec, FCoE or RSC. Packets that match these cases might be dropped from queues that use legacy receive descriptors.

Refer to Table 7-11 and the field descriptions that follow.

Table 7-11 Legacy Receive Descriptor (RDESC) Layout

	63	48 47	40 39	2 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. In order 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 7.1.13.

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 7-13.

Table 7-12 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, then 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 please see Section 7.4.

IPCS (IPv4 Checksum), L4CS (L4 Checksum), UDPCS (UDP Checksum)

these bits are described in the following table. In I/O mode: 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.

See Section 7.1.11 for a description of supported packet types for receive checksum offloading. IPv6 packets do not have the *IPCS* bit set, but might have the *L4CS* bit and *UDPCS* bit set if the X540 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 7-13 and the following text describes the possible errors reported by the hardware.

Table 7-13 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.

IPv6/UDP packets that carry a null UDP checksum field are reported with L4CS=1 and TPCE=1.

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. 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) then the VLAN header is stripped from the packet data storage. In this case the 16 bits of the VLAN tag, priority tag and CFI from the received packet are posted to the *VLAN Tag* field in the receive descriptor. Otherwise, the *VLAN Tag* field contains 0x0000.

Table 7-14 VLAN Tag Field Layout (for 802.1q Packet)

15 13	12	11 0
PRI	CFI	VLAN

Priority and CFI are part of 803.1Q specifications. The VLAN field is provided in network order.

7.1.6 Advanced Receive Descriptors

7.1.6.1 Advanced Receive Descriptors — Read Format

Table 7-15 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 7-15 Descriptor Read Format

	63 1	0
0	Packet Buffer Address [63:1]	A0/NSE
8	Header Buffer Address [63:1]	DD

Packet Buffer Address (64)

This is the physical address of the packet buffer. The lowest bit is either A0 (LSB of the address) or No Snoop Enable (NSE), depending on the RXCTL.RXdataWriteNSEn bit of the relevant queue. NSE does not affect the data written to the header buffer address. When software sets the *NSE* bit, the X540 places the received packet associated with this descriptor in memory at the packet buffer address with the *No-Snoop* bit set in the PCIe attribute fields. Using no-snoop expedites the DMA from the X540 to the host memory on the expense of coherency lost between host memory and CPU cache (refer to the following notes).

Note: When *No-Snoop Enable* is used, relaxed ordering should also be enabled with CTRL_EXT.RO_DIS.

No-snoop can be enabled for packet buffers that are not in the processor's cache. The software driver should be able to conclude that the buffers were not touched by the processor since they were delivered to the driver, therefore, snooped cycles would have yielded a cache miss.

If software needs to access the packet buffer it should invalidate the processor cache forcing the processor to fetch the host memory rather than its internal cache.

Using no-snoop is most efficient when IOAT moves the data from the packet buffer into application buffers, since it accesses the host memory rather than CPU cache.

Header Buffer Address (64)

The physical address of the header buffer with the lowest bit being Descriptor Done (DD). When a packet spans in multiple descriptors, the header buffer address is used only on the first descriptor. 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.

When a packet spans in more than one descriptor, the header buffer address is not used for the second, third, etc. descriptors; only the packet buffer address is used in this case.

Note: The X540 does not support null descriptors meaning packet or header addresses are zero.

7.1.6.2 Advanced Receive Descriptors — Write-Back Format

When the X540 writes back the descriptors, it uses the format listed in Table 7-16. The SRRCTL[n]. DESCTYPE should be set to a value other than 000b when using the advanced descriptor format.

Table 7-16 Descriptor Write-Back Format

	63 48	47 32	31	30 21	20 17	16 4	3 0
0	RSS Hash / Fragment Checksum / PCoE_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 X540 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

RSS Type	Description
0x7	HASH_UDP_IPv4
0x8	HASH_UDP_IPv6
0x9 – 0xE	Reserved
0xF	Packet reports flow director filters status

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.

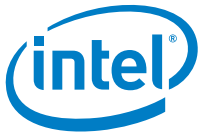
Bit Index	Bit 11 = 0	Bit 11 = 1 (L2 packet)
0	IPV4 — IPv4 header present	Ethertype — ETQF register index that matches the packet. Special types are defined for 802.1X, 1588 and FCoE.
1	IPV4E — IPv4 with extensions	
2	IPV6 — IPv6 header present	
3	IPV6E- IPv6 with extensions	Reserved for extension of the *EtherType* field.
4	TCP — TCP header present	Reserved for extension of the *EtherType* field.
5	UDP — UDP header present	Reserved
6	SCTP — SCTP header	Reserved
7	NFS — NFS header	Reserved
8	IPSec ESP – IPSec encapsulation	Reserved
9	IPSec AH – IPSec encapsulation	Reserved
10	MACsec – MACsec encapsulation	MACsec – MACsec encapsulation
11	0b = non L2 packet	1b = L2 packet
12	Reserved	Reserved

Note: UDP, TCP and IPv6 indications are not set in an IPv4 fragmented packet.

In IOV mode, packets might be received from other local VMs. the X540 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.

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 7.1.10 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 only on the last descriptor (EOP) of a packet. See additional details on SPH, PKT_LEN and HDR_LEN as a function of split modes in Table 7-21.

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.

Fragment Checksum

For non-FCoE packets, this field might hold the UDP fragment checksum (described in Section 7.1.13) if both the RXCSUM.PCSD bit is cleared and RXCSUM.IPPCSE bit is set. This field is meaningful only for UDP packets when the *UDPV* bit in the Extended Status word is set.

RSS Hash / Flow Director Filters ID

For non-FCoE packets, this field might hold the RSS hash value or flow director filters ID if the RXCSUM.PCSD bit is set. Further more, if the FDIRCTRL.Report-Status bit is set, then the flow director filters ID is reported; otherwise, the RSS hash is reported.

Table 7-17 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 7.1.2.8. 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 7.1.2.7). 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 32K 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 7-18 lists the extended status word in the last descriptor of a packet (*EOP* bit is set). Table 7-19 lists the extended status word in any descriptor but the last one of a packet (*EOP* bit is cleared).

Table 7-18 Receive Status (RDESC.STATUS) Layout of *Last* Descriptor

19	18	17	16	15	14	13	12	11	10
BMC	LB	SECP	TS	Rsv				LLINT	UDPV

VEXT	Reserved	PIF	IPCS	L4I	UDPCS	VP	FLM	EOP	DD
			FCEOFs	FCSTAT					
9	8	7	6	5	4	3	2	1	0

Table 7-19 Receive Status (RDESC.STATUS) Layout of *Non-Last* Descriptor

19 4	3 : 2	1	0
Next Descriptor Pointer — NEXTP	Rsv	EOP = 0b	DD

Rsv (8), Rsv (15:12) — 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.

EOP (1) and DD (0) — *End of Packet* and *Done* bits are listed in the following table:

DD	EOP	Description
0	0	Software setting of the descriptor when it hands it to hardware.
0	1	Reserved (invalid option)
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 is 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.

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 7.1.13. 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.

LLINT (11) — The *Low Latency Interrupt* bit indicates that the packet caused an immediate interrupt via the low latency interrupt mechanism.

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 7.9.

SECP (17) — Security processing bit indicates that the hardware identified the security encapsulation and processed it as configured.

MACsec processing — This bit is set each time MACsec processing is enabled regardless if a matched SA was found.

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 7.1.

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 7-20 and the following text describe the possible errors reported by hardware.

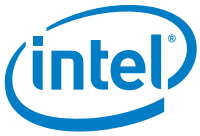


Table 7-20 Receive Errors (RDESC.ERRORS) Layout

11	10	9	8:7	6:4	3	2:0
IPE	L4E	RXE	Rsv	Rsv	HBO	Rsv
FCEOFe			SECERR			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, 10b or 11b.
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 PSRCTL.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 PSRCTL.BSIZEHEADER bytes of the packet.

Rsv (5:4) — Reserved at zero.

SECERR (8:7): Security error indication for MACsec or IPsec. This field is meaningful only if the *SECP* bit in the extended status is set.

SECERR	MACsec Error Reporting	IPsec Error Reporting
00	No errors found or no security processing	No errors found while an active SA found or no security processing.
01	No SA match	Invalid IPsec Protocol: IPsec protocol field (*ESP* or *AH*) in the received IP header does not match expected one stored in the SA table.
10	Replay error	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	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).

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.	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 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).

VLAN Tag (16-bit offset 48, 2nd line)

This field is described in the legacy descriptor format in Section 7.1.5.

7.1.7 Receive Descriptor Fetching

7.1.7.1 Fetch On Demand

The X540 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.

7.1.8 Receive Descriptor Write-Back

The X540 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 7.1.7), then 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 7.11.5.1 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 X540.

7.1.9 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.

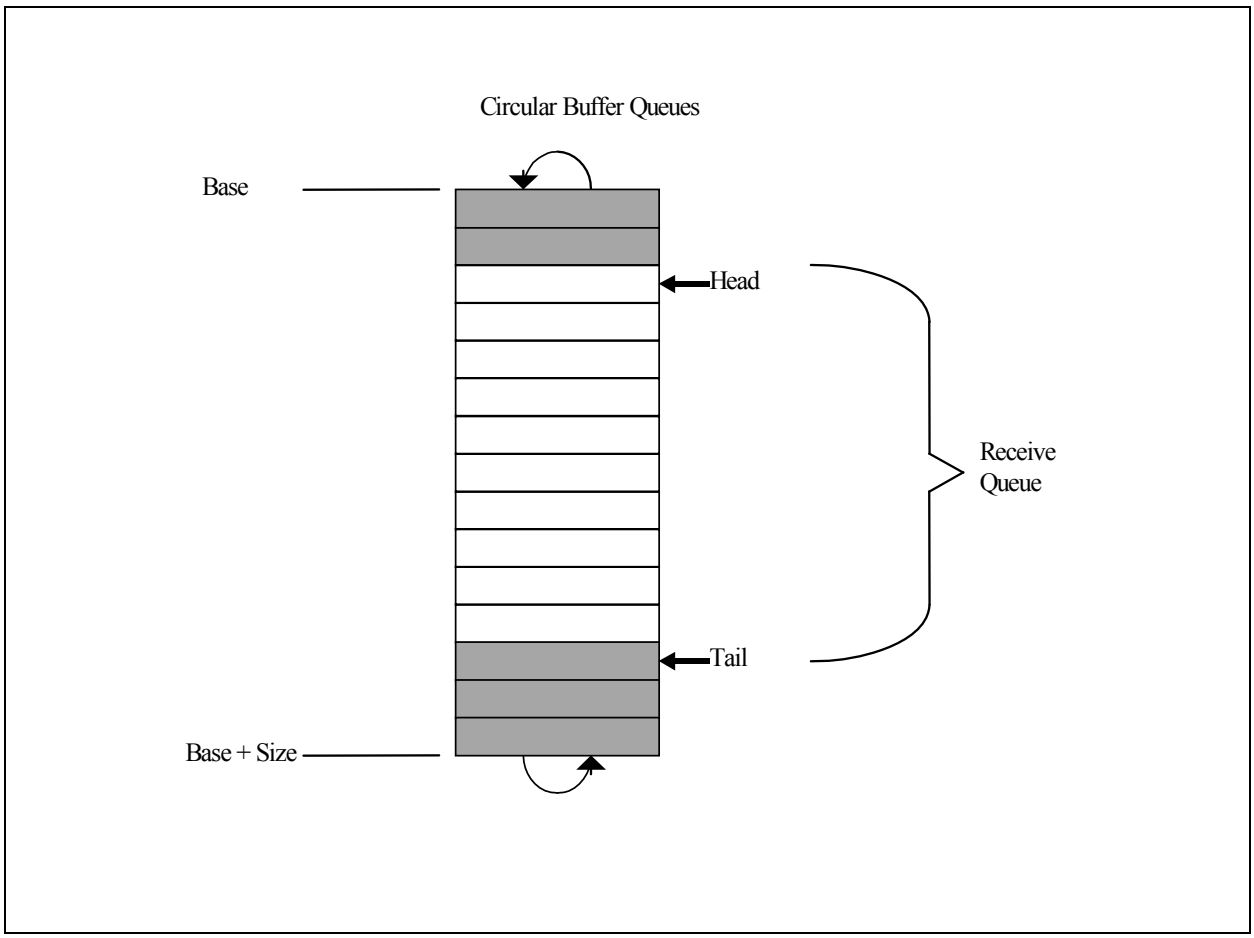


Figure 7-11 Receive Descriptor Ring Structure

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 X540 adjusts its internal tail pointer(s) accordingly. As packets arrive, they are stored in memory and the internal head pointer(s) is incremented by the X540.

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 X540 stops storing packets in system memory until software advances the tail pointer(s), making more receive buffers available.

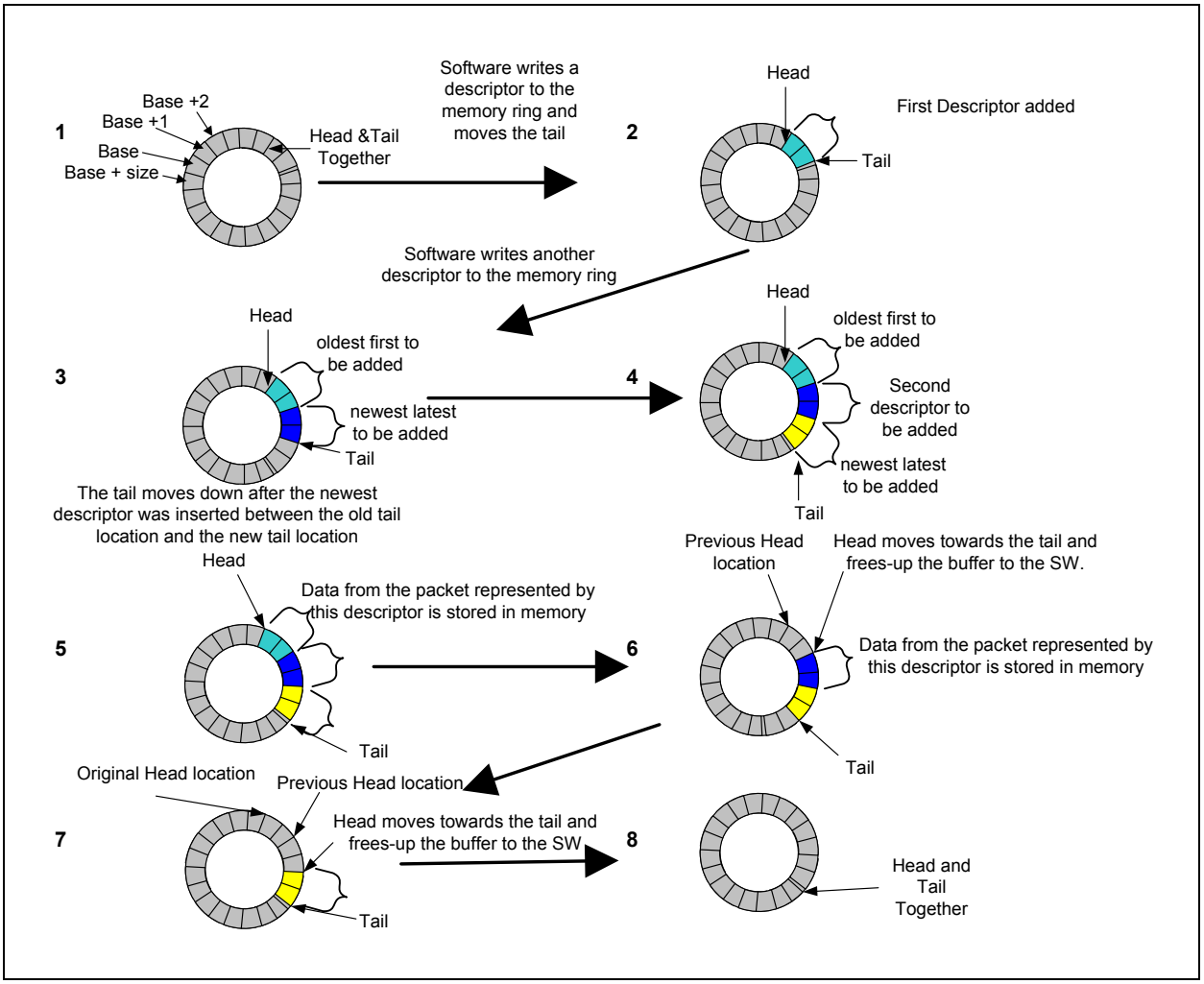


Figure 7-12 Descriptors and Memory Rings

The X540 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 incremented 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 16-byte boundary and is stored in two consecutive 32-bit registers. Hardware ignores the lower 4 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 PIO 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.

  For testability purpose only: If the tail pointer is larger then the ring length, then the X540 reads the descriptor ring in an endless loop until the queue is disabled. Prior to setting such a tail pointer value, it is required to initialize all the descriptors of the ring and set the RDWBOFST register. During reception, hardware does not write back the Rx descriptors on the Rx ring since they are needed for the endless reception. Instead, hardware writes back the Rx descriptors at the Rx descriptor offset plus RDWBOFST. RDWBOFST is defined in descriptor units as the head and tail registers while that there is a single value defined for all Rx descriptor rings.

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.

7.1.9.1 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. In order to avoid this

situation, the X540 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.

7.1.10 Header Splitting

7.1.10.1 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 in order to be processed by the host CPU. It is recommended to perform this transaction with DCA enabled (see Section 7.5).

The packet (header + payload) is stored in memory through a (optionally) non-snoop transaction. Later, an IOAT transaction moves the payload from the driver space to the application memory.

The X540'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 X540 uses packet splitting when the SRRCTL[n].DESCTYPE is greater than one.

7.1.10.2 Description

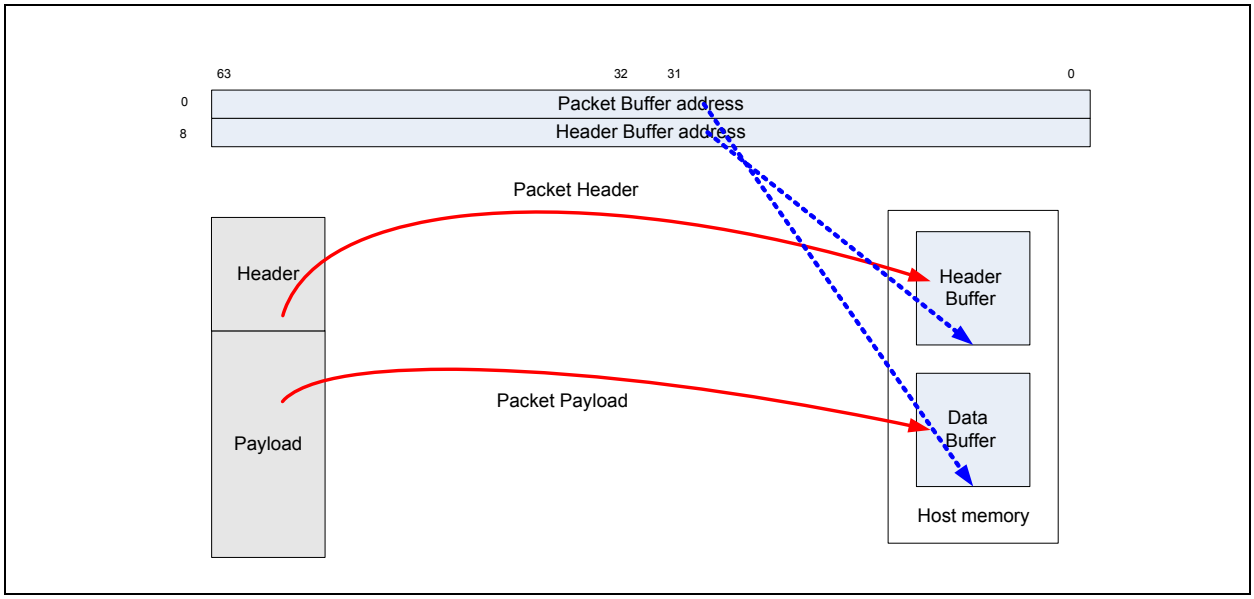


Figure 7-13 Header Splitting Diagram

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.

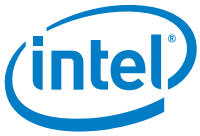
The sizes of these buffers are statically defined in the SRRCTL[n] registers:

- 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 X540 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 the PSRTYPE register section 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 fragmented extension header.



- Header split must not be used in a queue used for a FCoE large receive.
- An IP packet with more than a single IP header (such as any combination of IPv4 and IPv6 tunneling) is not split.
- 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 X540 always include IPsec header in a split done at IP boundary.
  - IPsec packets handled in software must never do header split.

Table 7-21 lists the behavior of the X540 in the different modes.

Table 7-21 Behavior in Header Split Modes

DESCTYPE	Condition	SPH	HBO	PKT_LEN	HDR_LEN	Header and Payload DMA
Split	1. Header can't be decoded	0	0	Min(packet length, buffer size)	0x0	Header + Payload -> Packet Buffer
	2. Header <= BSIZEHEADER	1	0	Min (payload length, buffer size)[3]	Header size	Header -> Header Buffer Payload -> Packet Buffer
	3. Header > BSIZEHEADER	1	1	Min (packet length, buffer size)	Header size[5]	Header + Payload -> Packet Buffer
Split – always use header buffer	1. Header can't be decoded and packet length <= BSIZEHEADER	0	0	0x0	Packet length	Header + Payload -> Header Buffer
	2. Header can’t be decoded and packet length > BSIZEHEADER	0	0	Min (packet length – BSIZEHEADER, data buffer size)	BSIZEHEADER	Header + Payload -> Header + Packet Buffers[4]
	3. Header <= BSIZEHEADER	1	0	Min (payload length, data buffer size)	Header Size	Header -> Header Buffer Payload -> Packet Buffer
	4. Header > BSIZEHEADER	1	1	Min (packet length – BSIZEHEADER, data buffer size)	Header Size[5]	Header + Payload -> Header + Packet Buffer

Notes:

1. Partial means up to BSIZEHEADER.
2. HBO is set to 1b if the header size is bigger than BSIZEHEADER and zero otherwise.
3. In a header only packet (such as TCP ACK packet), the PKT_LEN is zero.
4. If the packet spans more than one descriptor, only the header buffer of the first descriptor is used.
5. HDR_LEN doesn't reflect the actual data size stored in the header buffer. It reflects the header size determined by the parser.

7.1.11 Receive Checksum Offloading

The X540 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 X540. The X540 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 X540 calculates the TCP or UDP checksum and indicates a pass/fail condition to software via the *TCP/UDP Checksum Error* bit (RDESC.TCPE). These error bits are valid when the respective status bits indicate the checksum was calculated for the packet (RDESC.IPCS and RDESC.L4CS, respectively).

Similarly, if RFCTL.Ipv6_DIS and RFCTL.IP6Xsum_DIS are cleared to zero, the X540 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).

Supported frame types:

- Ethernet II
- Ethernet SNAP

Table 7-22 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

Table 7-22 Supported Receive Checksum Capabilities (Continued)

Packet Type	Hardware IP Checksum Calculation	Hardware TCP/UDP Checksum Calculation
Packet is an IPv4 fragment.	Yes	UDP checksum assist
Packet is greater than 1522 bytes.	Yes	Yes
Packet has 802.3ac tag.	Yes	Yes

The previous table lists general details about what packets are processed. In more detail, the packets are passed through a series of filters to determine if a receive checksum is calculated.

Ethernet MAC Address Filter

This filter checks the MAC destination address to be sure it is valid (that is IA match, broadcast, multicast, etc.). The receive configuration settings determine which Ethernet MAC addresses are accepted. See the various receive control configuration registers such as FCTRL, MCSTCTRL (RTCL.UPE, MCSTCTRL.MPE, FCTRL.BAM), MTA, RAL, and RAH for details.

SNAP/VLAN Filter

This filter checks the next headers looking for an IP header. It is capable of decoding Ethernet II, Ethernet SNAP, and IEEE 802.3ac headers. It skips past any of these intermediate headers and looks for the IP header. The receive configuration settings determine which next headers are accepted. See the various receive control configuration registers such as VLNCTRL.VFE, VLNCTRL.VET, and VFTA for more details.

IPv4 Filter

This filter checks for valid IPv4 headers. The version field is checked for a correct value (4).

IPv4 headers are accepted if they are any size greater than or equal to five (Dwords). If the IPv4 header is properly decoded, the IP checksum is checked for validity.

IPv6 Filter

This filter checks for valid IPv6 headers, which are a fixed size and have no checksum. The IPv6 extension headers accepted are: Hop-by-hop, destination options, and routing. The maximum extension header size supported is 256 bytes. The maximum total header size supported is 1 KB.

IPv6 Extension Headers

IPv4 and TCP provide header lengths, which enable hardware to easily navigate through these headers on packet reception for calculating checksum and CRC, etc. For receiving IPv6 packets, however, there is no IP header length to help hardware find the packet's ULP (such as TCP or UDP) header. One or more IPv6 extension headers might exist in a packet between the basic IPv6 header and the ULP header. Hardware must skip over these extension headers to calculate the TCP or UDP checksum for received packets.

The IPv6 header length without extensions is 40 bytes. The IPv6 field *Next Header Type* indicates what type of header follows the IPv6 header at offset 40. It might be an upper layer protocol header such as TCP or UDP (next header type of 6 or 17, respectively), or it might indicate that an extension header follows. The final extension header indicates with it's *Next Header Type* field the type of ULP header for the packet.

IPv6 extension headers have a specified order. However, destinations must be able to process these headers in any order. Also, IPv6 (or IPv4) might be tunneled using IPv6, and thus another IPv6 (or IPv4) header and potentially its extension headers might be found after the extension headers.

The IPv4 next header type is at byte offset 9. In IPv6, the first next header type is at byte offset 6.

All IPv6 extension headers have the next header type in their first eight bits. Most have the length in the second eight bits (Offset Byte[1]) as follows:

Table 7-23 Typical IPv6 Extended Header Format (Traditional Representation)

1 0 1 2 3 4 5 6 7 8 9 0	2 0 1 2 3 4 5 6 7 8 9 0	3 0 1 2 3 4 5 6 7 8 9 0
Next Header Type	Length	

Table 7-24 lists the encoding of the *Next Header Type* field and information on determining each header type's length. Other IPv6 extension headers - not indicated in Table 7-24 - are not recognized by the X540. Any processing of packet content that follows such extension headers is not supported.

Table 7-24 Header Type Encoding and Lengths

Header	Next Header type	Header Length (units are bytes unless otherwise specified)
IPv6	6	Always 40 bytes
IPv4	4	Offset bits[7:4] Unit = 4 bytes
TCP	6	Offset Byte[12].Bits[7:4] Unit = 4 bytes
UDP	17	Always 8 bytes
Hop-by-Hop Options	0 - Note 1	8+Offset Byte[1]
Destination Options	60	8+Offset Byte[1]
Routing	43	8+Offset Byte[1]
Fragment	44	Always 8 bytes
Authentication	51	Note 3
Encapsulating Security Payload	50	Note 3
No Next Header	59	Note 2

Notes:

1. Hop-by-hop options header is only found in the first next header type of an IPv6 header.
2. When no next header type is found, the rest of the packet should not be processed.
3. Encapsulated security payload packet handled by software — The X540 cannot offload packets with this header type.

UDP/TCP Filter

This filter checks for a valid UDP or TCP header. The prototype next header values are 0x11 and 0x06, respectively.

7.1.12 SCTP Receive Offload

If a receive packet is identified as SCTP, the X540 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 7-25 SCTP Header

0 1 2 3 4 5 6 7	1 8 9 0 1 2 3 4 5	2 6 7 8 9 0 1 2 3	3 4 5 6 7 8 9 0 1
Source Port		Destination Port	
Verification Tag			
CRC Checksum (CRC32c)			
Chunks 1..n			

7.1.13 Receive UDP Fragmentation Checksum

The X540 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

Incoming Packet Type	Fragment Checksum	UDPV	UDPCS / L4CS
IPv6 packet	0	0	Depends on transport UDP: 1 / 1 TCP: 0 / 1
Non fragmented IPv4 packet			
Fragmented IPv4 with protocol = UDP, first fragment (UDP protocol present)	The unadjusted 1's complement checksum of the IP payload	1 if the UDP header checksum is valid (not 0)	1 / 0
Fragmented IPv4, when not first fragment	The unadjusted 1's complement checksum of the IP payload	0	1 / 0

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. Refer to Section 7.1.2.8.3 for supported packet formats.

7.1.14 Receive Statistics

General Notes:

- All Statistics registers are cleared on read. In addition, they stick at 0xFF...F when the maximum value is reached.
- For the receive statistics it should be noted that a packet is indicated as received if it passes the device filters and is placed into the packet buffer memory. A packet does not have to be DMA'd to host memory in order to be counted as received.
- Due to divergent paths between interrupt-generation and logging of relevant statistics counts, it might be possible to generate an interrupt to the system for a noteworthy event prior to the associated statistics count actually being incremented. This is extremely unlikely due to expected delays associated with the system interrupt-collection and ISR delay, but might be an explanation for interrupt statistics values that do not quite make sense. Hardware guarantees that any event noteworthy of inclusion in a statistics count is reflected in the appropriate count within 1 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.

Statistics Hierarchy:

The following diagram describes the relations between the packet flow and the different statistic counters.

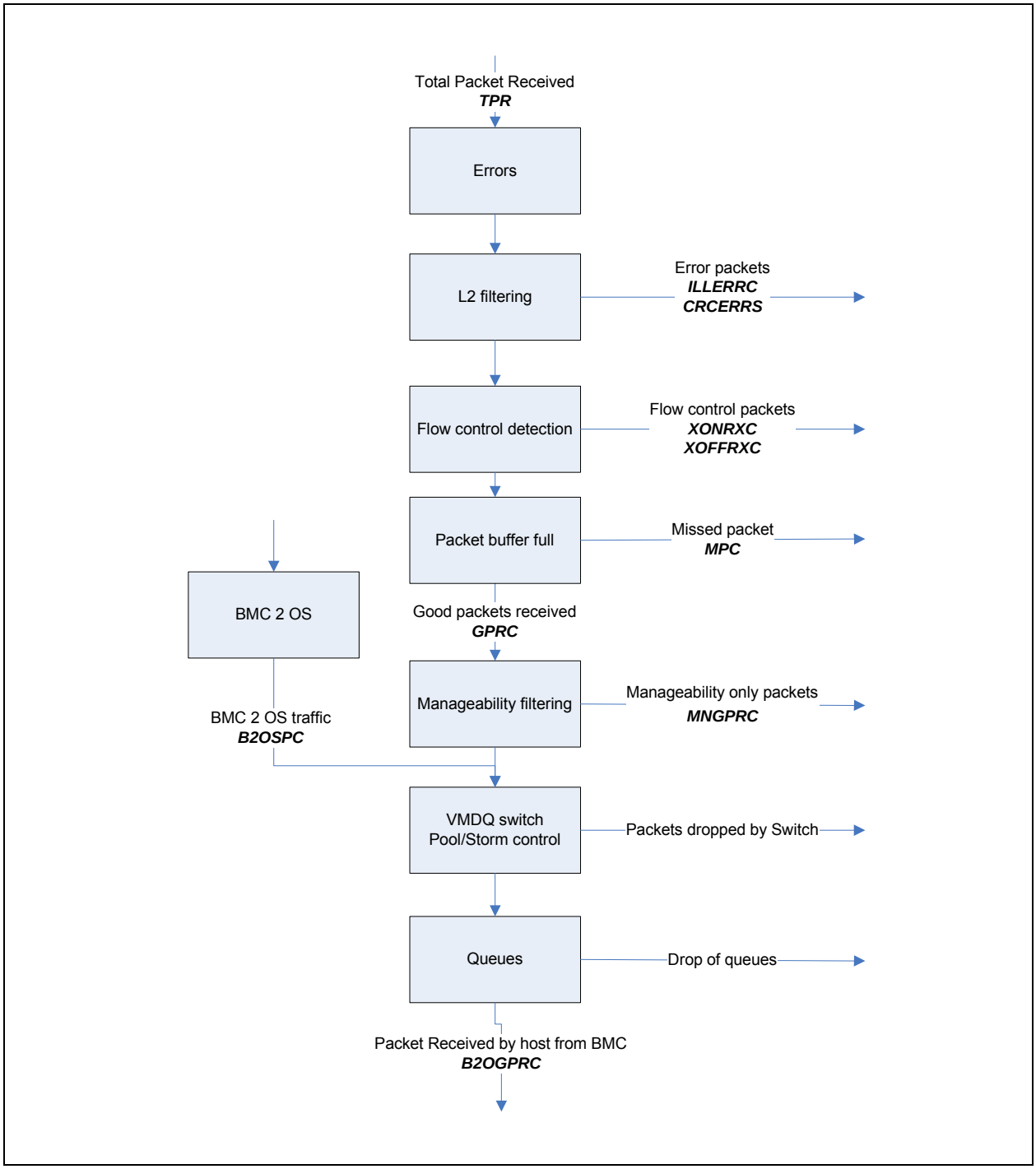


Figure 7-14 Receive Flow Statistics

Note: This page intentionally left blank.

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 Section 7.2.3.5.1 for WTHRESH details). For best performance it is recommended to minimize the number of buffers as possible.

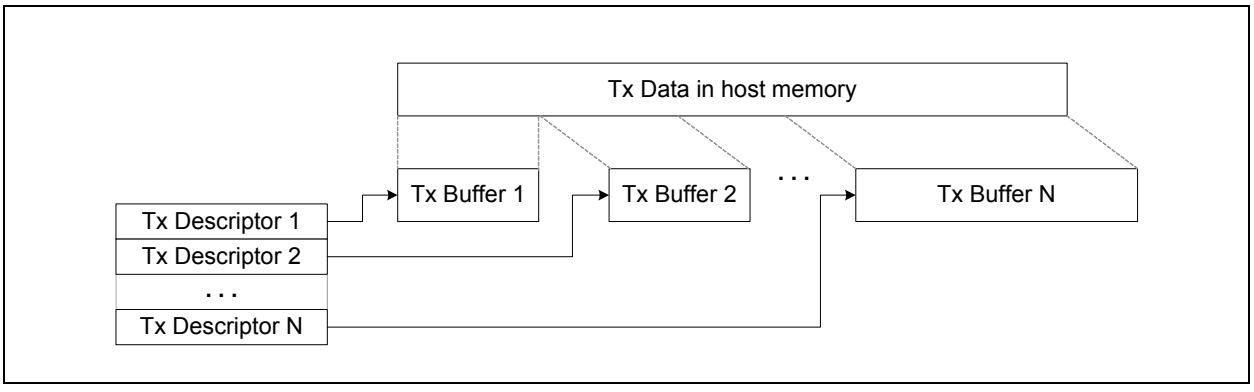


Figure 7-15 Tx Packet in Host Memory

7.2.1.2 Transmit Path in the X540

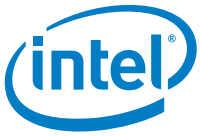
The transmit path in the X540 consists of the following stages:

- Descriptor plane
  - The X540 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.
  - A fetch mechanism loads Tx descriptors from the descriptor rings in system memory to the respective descriptor queues in the X540. 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 7.10.
  - 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 7.10.

7.2.1.2.1 Tx Queues Assignment

The X540 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 in order to reach the desired data rate. Therefore, multiple queues might be provided for the same functionality.

Table 7-26 lists the queuing schemes. Scheme selection is done via the MTQC register.

Table 7-26 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.

Device Setting			Device Functionality	
DCB_ena	VT_Ena	NUM_TC_OR_Q	Tx Queues	TC & VT
0	0	00	0 — 63	-
<> 0	<> 0	00	Reserved.	
0	0	<> 00	Reserved.	
1	0	01	Reserved.	
1	0	10	0 — 127	TC0 — TC3
1	0	11	0 — 127	TC0 — TC7
0	1	01	0 — 127	64 VMs
0	1	10	0 — 127	32 VMs
0	1	11	Reserved.	

Device Setting			Device Functionality	
DCB_ena	VT_Ena	NUM_TC_OR_Q	Tx Queues	TC & VT
1	1	01	Reserved.	
1	1	10	0 — 127	TC0 — TC3 & 32 VMs
1	1	11	0 — 127	TC0 — TC7 & 16 VMs

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 7-27 and Table 7-28.

Table 7-27 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 (63..0)						TSS
32 VFs + TSS *or* 4 TCs	VF (31 ..0)					TSS / TC	
16 VFs + 8 TCs	VF (15 ..0)				TC		

Table 7-28 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 X540 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.

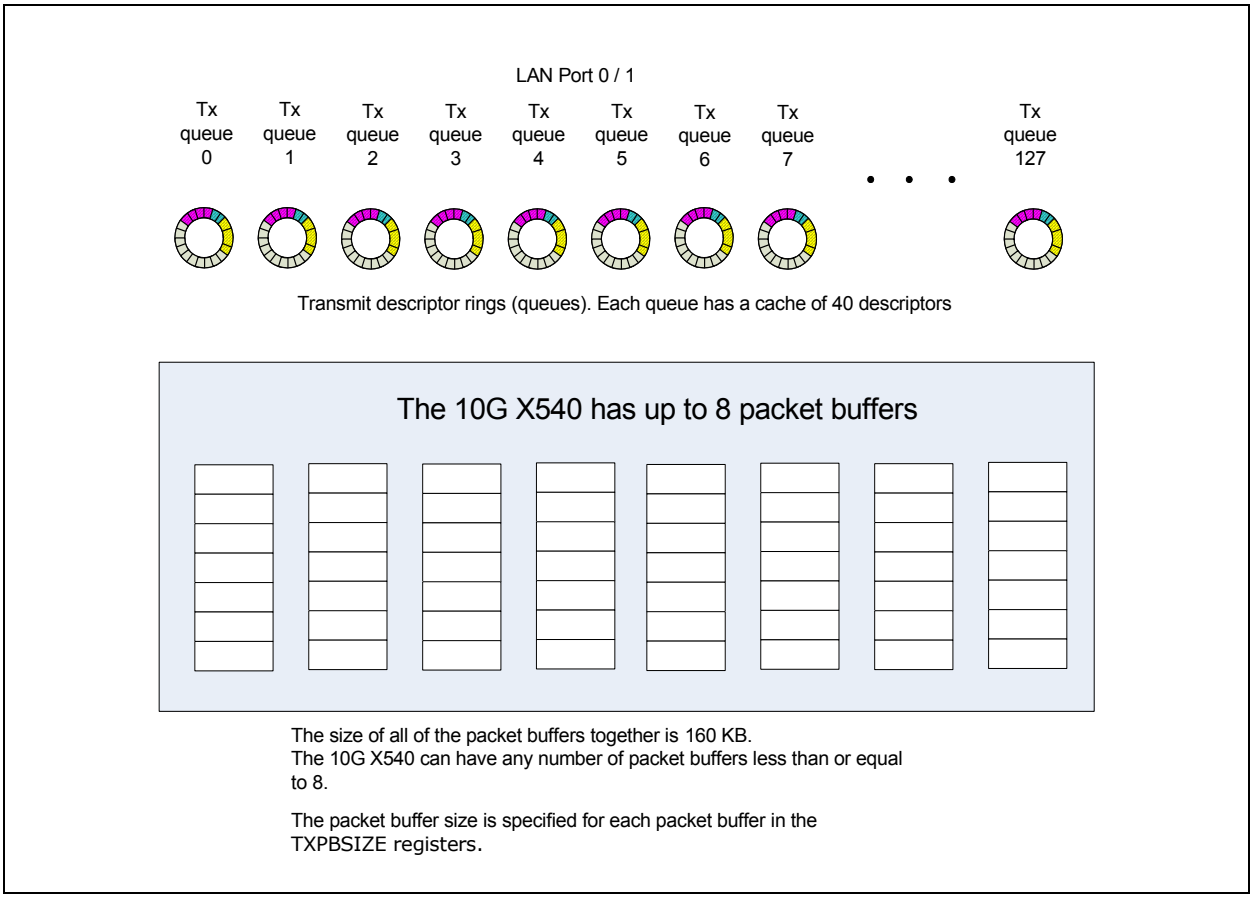


Figure 7-16 Tx Arbitration Schemes

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 register.

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 7.7.2.3. 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 7.7.2.3. 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 7.7.2.3.
- **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.

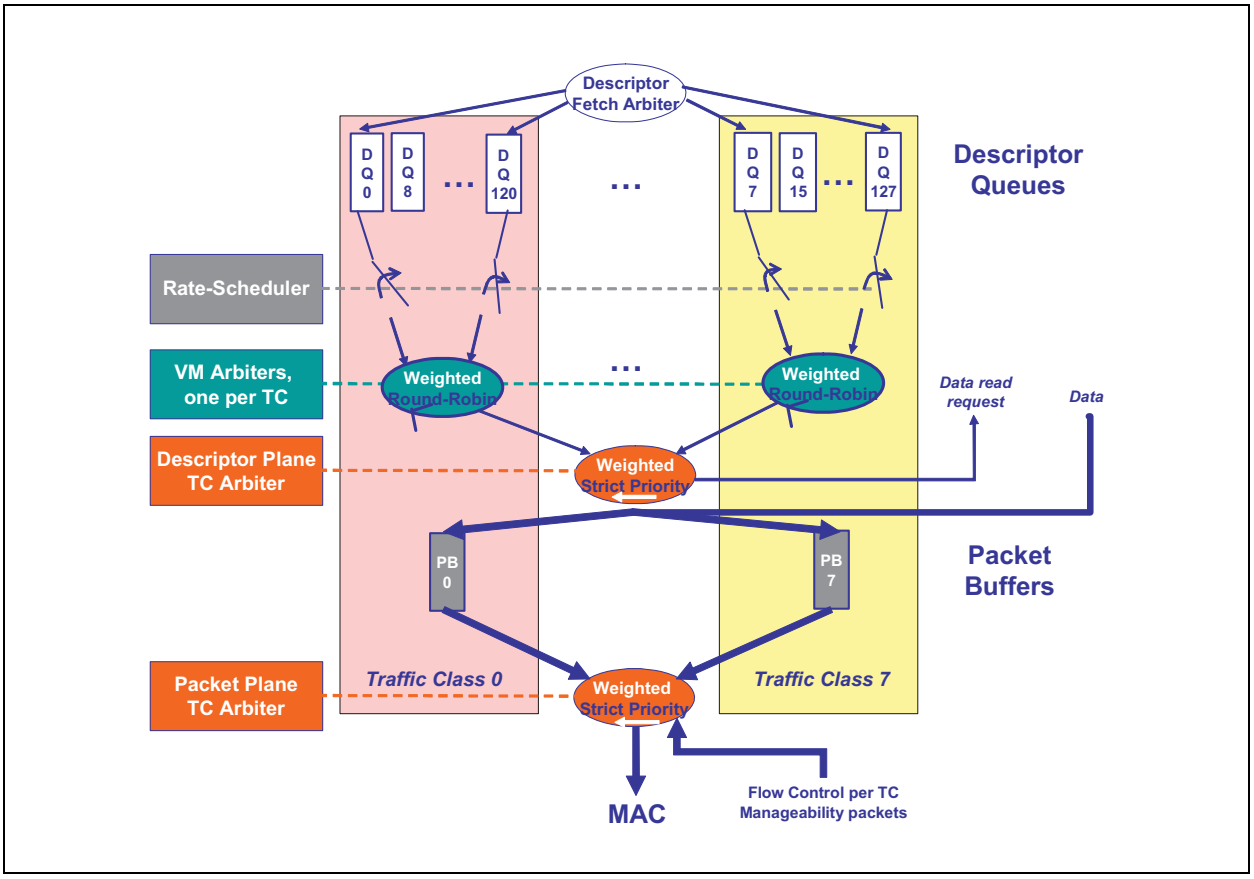


Figure 7-17 Transmit Architecture DCB-on / VT-on — Eight TCs Mode

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.

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 7-19, 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.

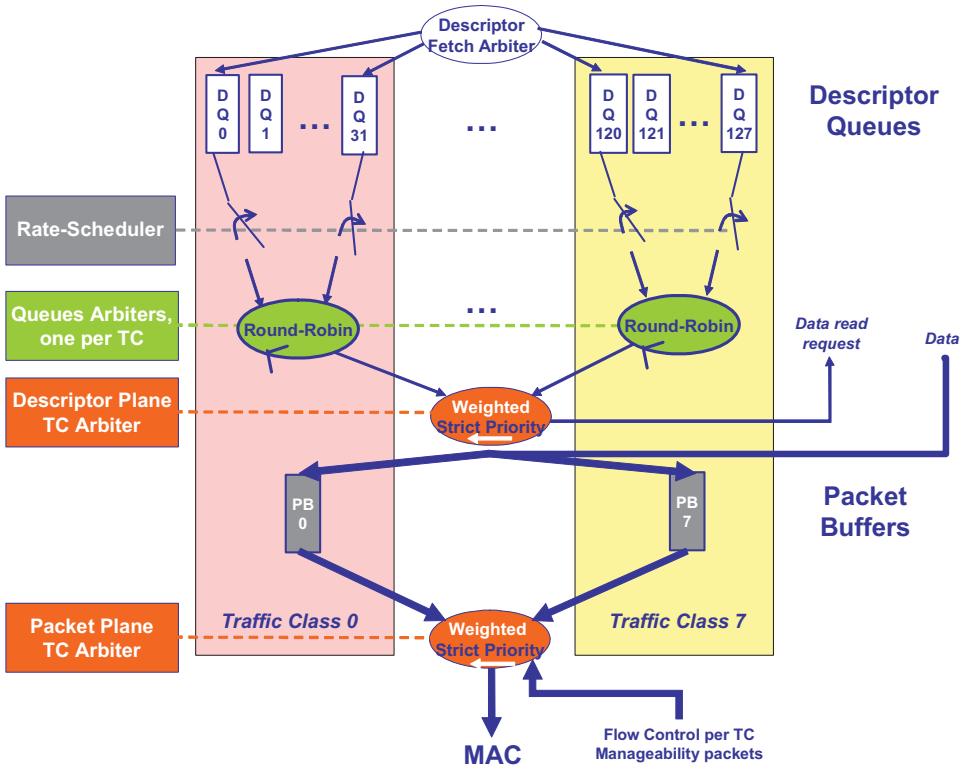


Figure 7-18 Transmit Architecture DCB-on / VT-off — Eight TCs Mode

DCB-off/VT-on

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).

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 7.7.2.3.

Packet Plane Arbiter:

- **Manageability** packets are inserted with strict priority over data packets.

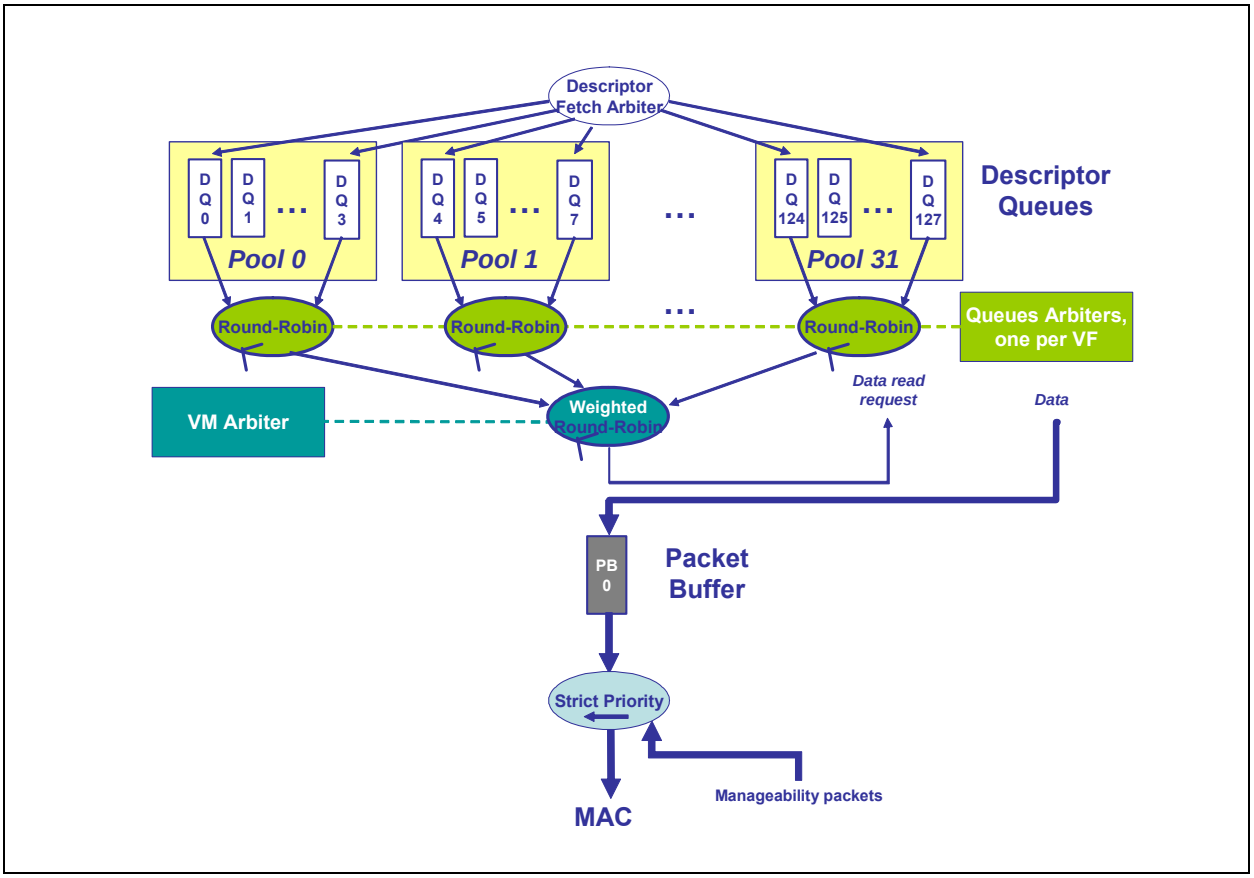


Figure 7-19 Transmit Architecture DCB-off / VT-on — 32 VFs

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.

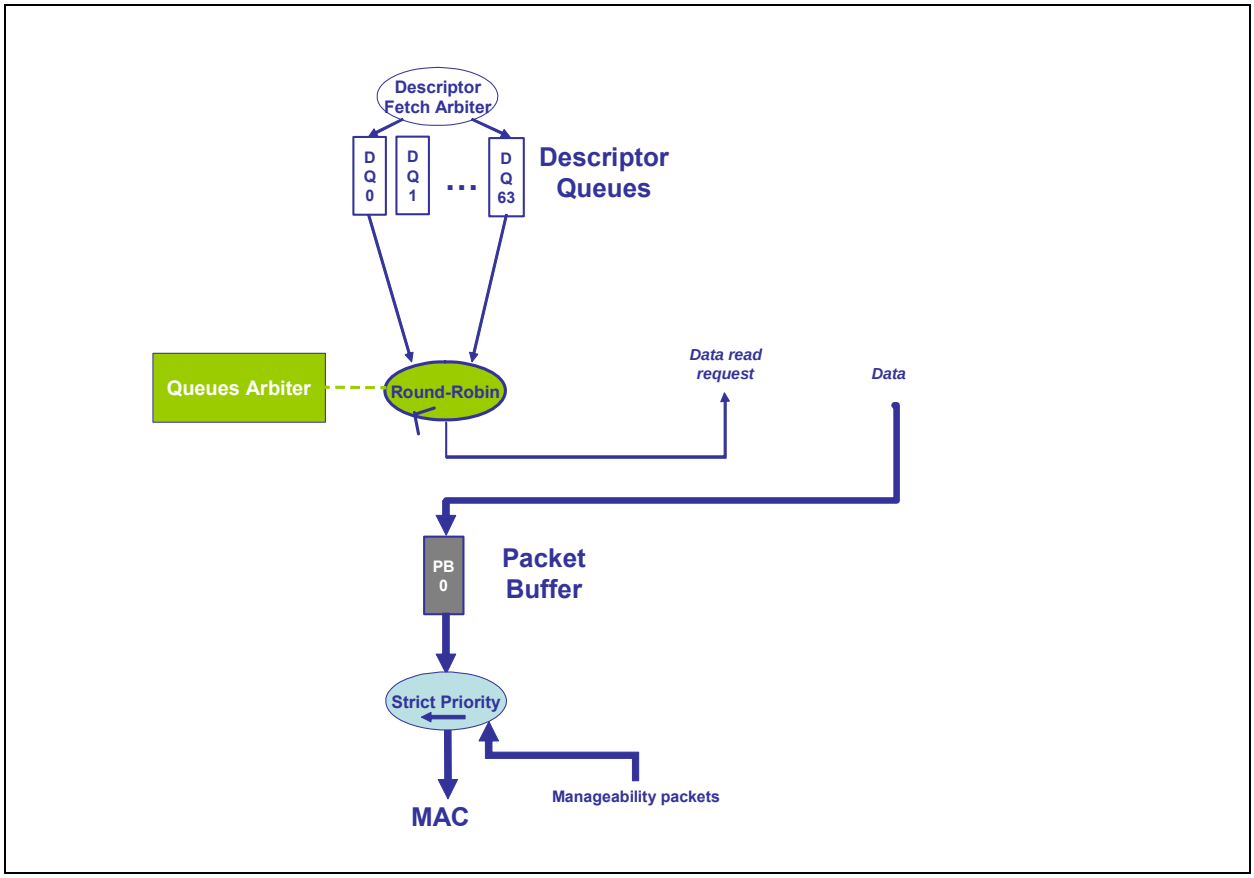


Figure 7-20 Transmit Architecture DCB-off/VT-off

7.2.2 Transmit Contexts

The X540 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 X540 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 X540 supports legacy descriptors and advanced descriptors.

Legacy descriptors are intended to support legacy drivers, in order 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, Rate Scheduler, MACsec, and IPsec. These modes are recognized by a dedicated enable bit for each.

In addition, the X540 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 7-29 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)	CFI	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 7-30. 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 7-30 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 7-31 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 unpadded 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 7-32.

Table 7-32 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 7-33 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 Tx 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 registers.
  - — TSO or TCP/IP checksum offload using a context descriptor.
  - — MACsec offload is requested.

Note that TSO and MACsec 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 *EOP* bit is meaningful in all descriptors.

Transmitted.Status — STA (4)

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 7-34 VLAN Field (TDESC.VLAN) Layout

15 13	12	11 0
PRI	CFI	VLAN

7.2.3.2.3 Advanced Transmit Context Descriptor

Table 7-35 Transmit Context Descriptor (TDESC) Layout — (Type = 0010)

	63 48	47 42	41 32	31 16	15 9	8 0
0	RSV	FCoEF	IPsec SA Index	VLAN	MACLEN	IPLEN/HEADLEN
8	MSS	L4LEN	RSV IDX	QCNLEN	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 7-49. HEADLEN does not include the MACsec header if it exists. HEADLEN is meaningful only if transmit FCoE offload is enabled by setting the *FCoE* bit in the TUCMD field. HEADLEN that matches Figure 7-49 equals 56 or 64 for packets without FC extended headers. The X540 supports FC extended headers only for single send. Segmentation offload can be used only when extended

  headers are not present.

MACLEN (7)

- For nonFCoE 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)/*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 Table 7-36 lists the value of MACLEN in the different cases.

Table 7-36 MACLEN Values

SNAP	Regular VLAN	Extended VLAN	MACLEN
No	By hardware or no	No	14
No	By hardware or no	Yes	18
No	By software	No	18
No	By software	Yes	22
Yes	By hardware or no	No	22
Yes	By hardware or no	Yes	26
Yes	By software	No	26
Yes	By software	Yes	30

- 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 7-49. The MACLEN that matches Figure 7-49 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 CFI and priority fields as listed in Table 7-34.

Note that the *VLAN* field is provided in network order. See Section 7.2.3.2.1.

Ipsec SA IDX (10) – IPsec SA Index. 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) — see the following:

- EOF (bits 1:0) — End of frame delimiter index.
- ORIE (bit 4) — Orientation relative to the last frame in an FC sequence.

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 7-37. The values EOF0...EOF3 are taken from the TEOFF register.

Table 7-37 EOF Codes in TSO

EOF Bits in the Context Descriptor	ORIE Bit in the Context Descriptor	Last Frame of the TSO or TSO in Single Frame	Other Frames of the TSO
00 (EOFn)	0 (not a sequence end)	EOF0 (EOFn)	EOF0 (EOFn)
00 (EOFn)	1 (sequence end)	EOF1 (EOFt)	EOF0 (EOFn)
Else = Reserved	N/A	N/A	N/A

- SOF (bit 2) — Start of frame delimiter index.
- ORIS (bit 5) — Orientation relative to the first frame in an FC sequence.

In a single packet send, SOF is taken from the data buffer. In TSO, hardware places the SOF in the transmitted packet replacing the data buffer content. The *SOF* and *ORIS* bits in the context descriptor define the SOF that is placed by the hardware as listed in Table 7-38. The values SOF0...SOF3 are taken from the SOFF register.

Table 7-38 SOF Codes

SOF bit (2)	ORIS bit (5)	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 start)	SOF2 (SOFn2)	SOF2 (SOFn2)	SOF2 (SOFn2)

- 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.

RSV(16)

Reserved

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 7.13.2 for a description of the offload provided by the hardware while transmitting a single frame and TSO.
- Encryption (bit 5) — ESP encryption offload is required. Meaningful only to packets for which the *IPSEC* bit is set in their advanced Tx data descriptor.
- IPSEC_TYPE (bit 4) — Set for ESP. Cleared for AH. Meaningful only to packets for which the *IPSEC* bit is set in their advanced Tx data descriptor.
- L4T (bit 3:2) — L4 Packet TYPE (00: UDP; 01: TCP; 10: SCTP; 11: RSV)
- IPV4(bit 1) — IP Packet Type: When 1b, IPv4; when 0b, IPv6
- SNAP (bit 0) — SNAP indication

DTYP (4)

This field is always 0010b for this type of descriptor.

RSV(1)

Reserved

DEXT (1) — Descriptor extension (one for advanced mode)

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 X540 supports only two context descriptors per queue, the two MS bits are reserved and should be set to 0b.

RSV(1)

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.

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]  + 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

FCoE Segmentation

The total length of each frame (or segment) excluding Ethernet CRC equals 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. The following table lists the required valid fields according to the different offload options.

Table 7-39 Valid Fields by Offload Option

	Context Fields ->	FCoE	FCoEF	VLAN	MACLEN	IPLEN/HEADLEN	L4LEN	SNAP	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	yes	0
	FCoE CRC	yes	yes		yes	yes	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a		yes	1
	FCoE Seg	yes	yes		yes	yes	n/a	n/a	n/a	n/a	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	n/a		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 7-40 Advanced Transmit Data Descriptor Read Format

0	Address[63:0]									
8	PAYLEN	POPTS	CC	IDX	STA	DCMD	DTYP	MAC	RSV	DTALEN
	63 46	45 40	39	38 36	35 32	31 24	23 20	19 18	17 16	15 0

Table 7-41 Advanced Transmit Data Descriptor Write-back Format

0	RSV		
8	RSV	STA	RSV
	63 36	35 32	31 0

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.

RSV(2)

Reserved

MAC (2) — see the following. This field is meaningful on the first descriptor of the packet(s).

- ILSec (bit 0) — Apply MACsec on packet. When set, hardware includes the MACsec header (SecTAG) and MACsec header digest (signature). The MACsec processing is defined by the *Enable Tx MACsec* field in the LSECTXCTRL register. The *ILSec* bit in the packet descriptor should not be set if MACsec processing is not enabled by the *Enable Tx MACsec* field. If the *ILSec* bit is set erroneously while the *Enable Tx MACsec* field is set to 00b, then the packet is dropped.
- 1588 (bit 1) — IEEE1588 time stamp packet.

DTYP (4)

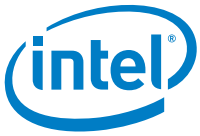
0011b for advanced data descriptor. DTYP should be valid in all descriptors of the packet(s).

DCMD (8) — see the following:

- 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 in order 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, then 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 7-39 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.

POPTS (6)

- Rsv (bit 5) — Reserved
- ISCO (bits 4:3) - iSCSI Orientation

  This field indicates that the TSO is the first/middle/last or first and last TSO of the iSCSI PDU that is being offloaded for CRC calculation.

  00b = First TSO of iSCSI PDU

01b = Middle TSO of iSCSI PDU

10b = Last TSO of iSCSI PDU

11b = First and Last - TSO contains the full iSCSI PDU

- IPSEC (bit 2) — Ipsec offload request.
- 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), Ethernet CRC or Ethernet padding. When MACsec offload is enabled, the PAYLEN field does not include the MACsec encapsulation. 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.

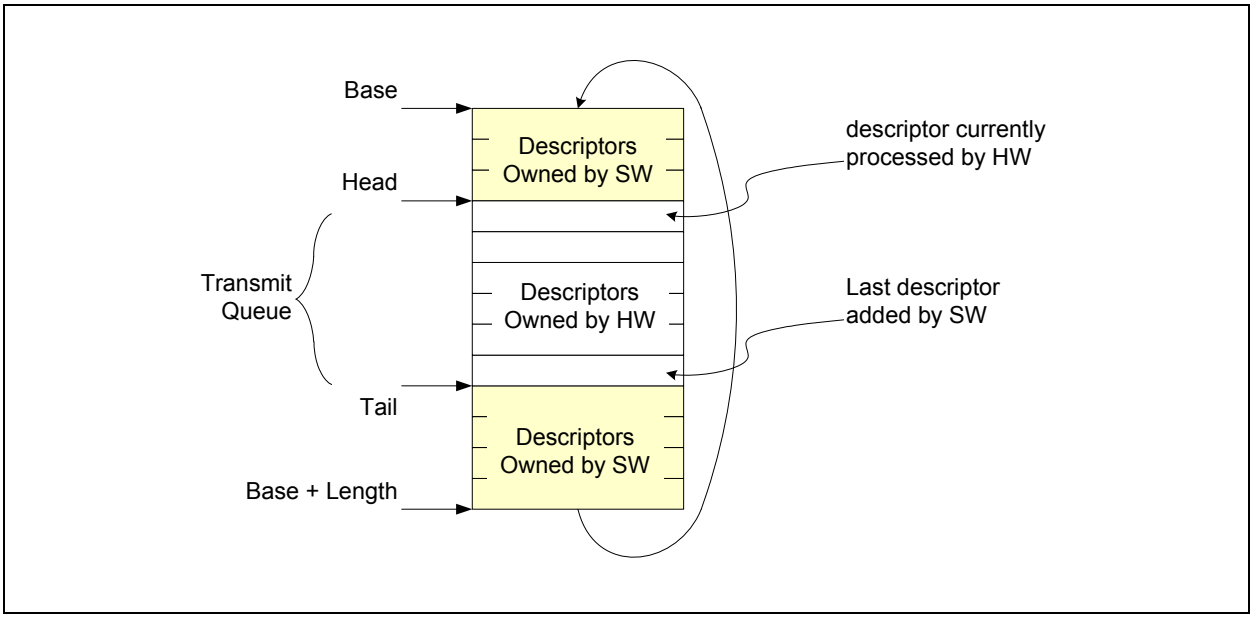


Figure 7-21 Transmit Descriptor Ring Structure

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 16-byte boundary and is stored in two consecutive 32-bit registers. Hardware ignores the lower four 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.

— For testability purpose only: If the tail pointer is larger then the ring length, then the X540 reads the descriptor ring in an endless loop until the queue is disabled. Prior to setting such a tail pointer value, it is required to initialize all the descriptors of the ring.

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 X540 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 then 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.
- In order 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).
- In order 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, in order 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'. In order 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 X540 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).

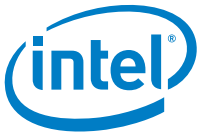
An additional mode in which transmit descriptors are not written back at all and the head pointer of the descriptor ring is written instead is described in the following section.

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. In order to improve the performance of this feature, the driver needs to program DCA registers to configure which CPU will be processing 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 X540 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 X540 saves significant CPU cycles. The device driver shares the additional tasks to support these options with the X540.

7.2.4.1 Assumptions and Restrictions

The following assumptions apply to the TCP / UDP segmentation implementation in the X540:

- 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 supported for segmentation with the following restriction. The location of the 802.3 length field in 802.3+SNAP packets is at MACLEN minus eight bytes (MACLEN is indicated in the context descriptor).
- IP tunneled packets are not supported for offloading under TSO operation.
- 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, optional IPSec 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 X540, 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 X540 enables interleaving between different TSO requests at an Ethernet packet level. In other words, the X540 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 X540 decides on the order of data fetched based on its QoS requirements (such as bandwidth allocation and priority).

In order 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.

— Since the header buffers are read multiple times, it is guaranteed on most platforms that by the second read, the data does not reside in the CPU caches any more. In that case, it is possible to avoid snooping the CPU cache during subsequent accesses to the same buffer. If the NoSnoop_LSO_hdr_buf CSR bit is set, the X540 sets the no snoop attribute in PCIe requests for header buffers belonging to second and later segments.

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

Typical TCP/IP transmit window size is 8760 bytes (about six full size frames). Today the average size on corporate Intranets is 12-14 KB, and normally the maximum window size allowed is 64 KB (unless Windows Scaling — RFC 1323 is specified). A TCP / UDP message can be as large as 256 KB and is generally fragmented across multiple pages in host memory. The X540 partitions the data packet into standard Ethernet frames prior to transmission. The X540 supports calculating the Ethernet, IP, TCP, and even UDP headers, include checksum, on a frame-by-frame basis.

Table 7-42 TCP/IP and UDP/IP Packet Format Sent by Host

Pseudo Header			Data
Ethernet	IPv4/IPv6	TCP/UDP	DATA (full TCP message)

Table 7-43 Packets Format Sent by Device

Pseudo Header (updated)	Data (first MSS)	FCS	...	Pseudo Header (updated)	Data (Next MSS)	FCS	...

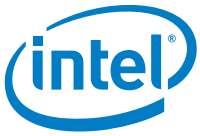
Frame formats supported by the X540 include:

- Ethernet 802.3
- IEEE 802.1Q VLAN (Ethernet 802.3ac)
- Ethernet Type 2
- Ethernet SNAP
- IPv4 headers with options
- IPv4 headers without options with one AH/ESP IPsec header
- IPv6 headers with extensions
- TCP with options
- UDP with options

VLAN tag insertion is 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 X540 has the ability to segment UDP traffic (in addition to TCP traffic); however, because UDP packets are generally fragmented at the IP layer, the X540'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 prepended. 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 TCP / 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 checksum 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 7-44 TCP Partial Pseudo-header Checksum for IPv4

IP Source Address		
IP Destination Address		
Zero	Layer 4 Protocol ID	Zero

Table 7-45 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 X540 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 X540 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 7-46 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 X540.

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.

MAC Header (for SNAP)

- Type/Len field = MSS + MACLEN + IPLEN + L4LEN — 14 - number of bytes added to the packet due to VLAN by software (4 bytes)

IPv4 Header

- IP Total Length = MSS + L4LEN + IPLEN
- Calculates the IP Checksum

IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension[1]

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.

MAC Header (for SNAP packets)

Type/Len field = MSS + MACLEN + IPLEN + L4LEN — 14 - number of bytes added to the packet due to VLAN by software (4 bytes)

IPv4 Header

- IP Identification: incremented 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]

1. IPV6_HDR_extension is calculated as IPLEN — 40 bytes.

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).

MAC Header (for SNAP packets)

- Type/Len field = Last frame payload bytes + MACLEN + IPLEN + L4LEN — 14 - number of bytes added to the packet due to VLAN by software (4 bytes)

IPv4 Header

- IP Total length = last frame payload bytes + L4LEN + IPLEN
- IP identification: incremented from last value (wrap around based on 16-bit width)
- Calculate the IP checksum

IPv6 Header

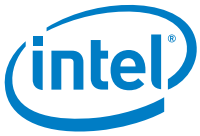
- Payload length = last frame payload bytes + L4LEN + IPV6_HDR_extension[1]

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.



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 X540'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.

As mentioned in Section 7.2.3, transmit descriptors, it is not necessary to set a new context for each new packet. In many cases, the same checksum context can be used for a majority of the packet stream. In this case, some performance can be gained by only changing the context on an as needed basis or electing to use the off-load feature only for a particular traffic type, thereby avoiding all context descriptors except for the initial one.

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.

Note: For IPsec packets offloaded by hardware in Tx, it is assumed that IPLEN provided by software in the Tx context descriptor is the sum of the IP header length and the IPsec header length. Thus, for the IPv4 header checksum offload, hardware could no longer rely on the *IPLEN* field provided by software in the Tx context descriptor, but should rely on the fact that no IPv4 options is present in the packet. Consequently, for IPsec offload packets, hardware computes IP header checksum over a fixed amount of 20 bytes.

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.

The 16-bit IPv4 header checksum is placed at the two bytes starting at MACLEN+10.

As mentioned in Section 7.2.3.2.3, transmit contexts, it is not necessary to set a new context for each new packet. In many cases, the same checksum context can be used for a majority of the packet stream. In this case, some performance can be gained by only changing the context on an as needed basis or electing to use the off-load feature only for a particular traffic type, thereby avoiding all context descriptors except for the initial one.

7.2.5.2 TCP and UDP Checksum

Three fields in the transmit context descriptor set the context of the TCP / UDP checksum offloading feature:

- MACLEN
- 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.

MACLEN+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+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

The following table 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 7-46 Checksums Supported by Packet Type

Packet Type	HW IP Checksum Calculation	HW 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

Table 7-46 Checksums Supported by Packet Type

Packet Type	HW IP Checksum Calculation	HW TCP/UDP/SCTP Checksum Calculation
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 IP options and no IPSec header (IP header is longer than 20 bytes)	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

7.2.6 Transmit Statistics

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 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 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.

Statistics Hierarchy:

The following diagram describes the relations between the packet flow and the different statistic counters.

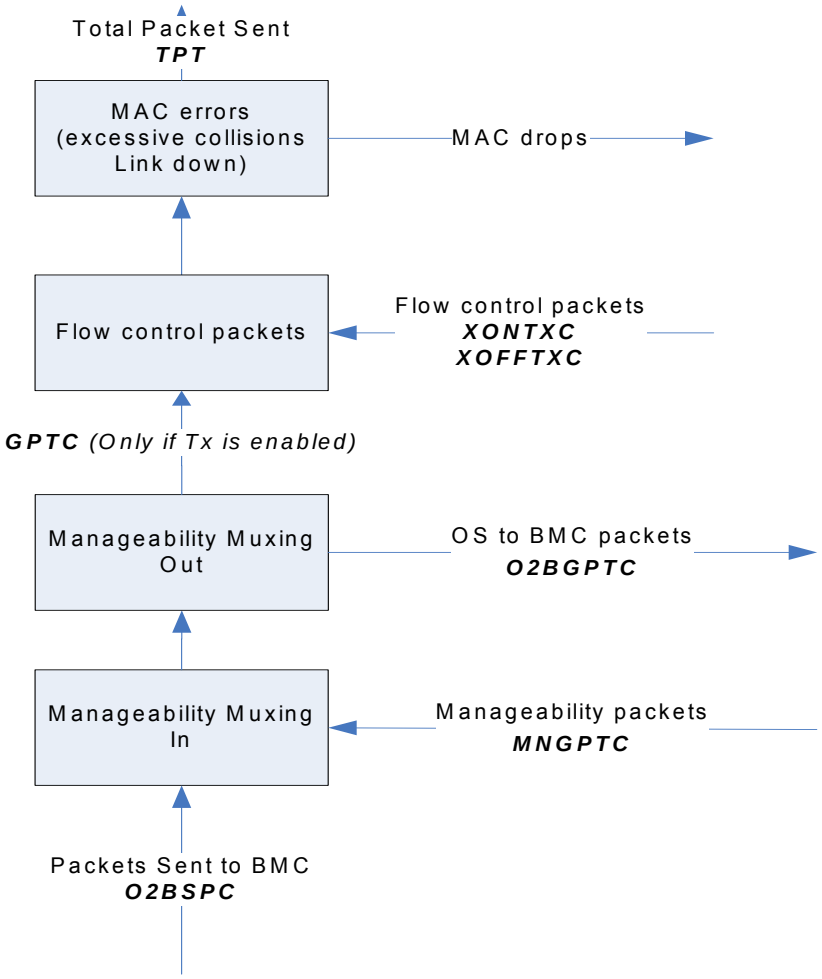


Figure 7-22 Transmit Flow Statistics

Note: This page intentionally left blank.

7.3 Interrupts

The X540 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

Physical Function (PF) Registers

The PF interrupt logic consists of the registers listed in the Table 7-47 followed by their description:

Table 7-47 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 and Low Latency Interrupt (LLI) setting]
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, EIAC, 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.

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 VFEIAC registers are not supported since 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.

7.3.1.1 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 or LLI as configured in the EITR register (LLI does not impact Tx). Following interrupt assertion, software cannot distinguish between ITR or LLI events.
- 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, then 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 Section 7.3.1.4).

7.3.1.2 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 either LLI or throttled interrupt depending on the GPIE.EIMEN setting: When the *EIMEN* bit is set, then setting the EICS register causes an LLI interrupt; When the *EIMEN* bit is cleared, then 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).

7.3.1.2.1 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 the same as any LLI Rx packet. Device behavior is described in Section 7.3.2.2.3 starting with the 2nd step.

7.3.1.3 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.

7.3.1.4 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 29:20 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.

7.3.1.5 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 29:20 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 X540 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 7.3.1.1.

The EITR registers include two types of throttling mechanisms: ITR and LLI. Both are described in the sections that follow.

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 timer, other and mail box (IOV mode) interrupts are not moderated. 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 1048.576 µs. These ITR interval times correspond to interrupt rates in the range of 488 K INT/sec to 953 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 or the last interrupt was LLI.
- **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 7.11.6). 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 LLI

LLI provides low latency service for specific packet types, bypassing the ITR latency. LLIs are bound by a credit-based throttling mechanism that limits the maximum rate of low latency events that require a fast CPU response. Low latency events are triggered by the write back of the LLI packets. It then generates an immediate interrupt if LLI credits are not exhausted. See more details on the credit mechanism in the Section 7.3.2.2.2. Note also that in the case of RSC, the interrupt is not immediate as described in Section 7.3.2.2.3.

7.3.2.2.1 LLI Filters and Other Cases

Following is a list of all Rx packets that are defined as low latency events (LLI packets):

- **LLI by 5-tuple / TCP flags / frame size** — The X540 supports a set of 128 filters that initiate LLI by a 5-tuple value, TCP flags, and frame size. An LLI is issued if any of the filters are set for LLI matches against the enabled fields of 5-tuple, TCP flags, and frame size. Configuration is done via the L34TIMIR register as follows per filter (more details about the 5-tuple filters can be found in Section 7.1.2.5):
- 5-tuple fields (protocol, IP address, port) and mask options for these fields
- Pool and pool mask
- LLITHRESH.SizeThresh — A frame with a length below this threshold triggers an interrupt. Unlike other fields, the *SizeThresh* field is shared by all filters (like there is a single copy of it). Matching the frame size is enabled by the *Size_BP* bit.
- *L34TIMIR[n].Size_BP* bit, when set to 0b, the size of the received packet is checked for LLI.
  - L34TIMIR[n].PSH_Bit bit, when set to 1b, the PSH flag is checked in the received packet for LLI.
  - L34TIMIR[n].LLI_EN bit, when set enables a Low Latency Interrupt if the following conditions are met: (1) The received packet matches the associated 5-tuple filter and... (2) If the Size_BP bit is cleared the packet length is smaller than the length defined by LLITHRESH.SizeThresh and... (3) If the PSH_Bit bit is set the PSH flag in the received packet is set
- **LLI field** — When set, an LLI is issued for packets that match the filter.
- **LLI by Ethertype** — The X540 supports eight Ethertype filters. Any filter has an LLI action defined by the *LLI* field in the ETQS registers.
- **LLI by VLAN priority** — The X540 supports VLAN priority filtering as defined in the IMIRVP register. Packets with VLAN header that have higher priority tagging than the one defined by IMIRVP register generates an LLI.
- **LLI by FCoE** — FCoE FCP_RSP packets can trigger LLI as defined in the FCRXCTRL.RSCINT bit. The X540 identifies FCoE packets by the Ethertype filters defined by the ETQF registers. FCP_RSP packets recognition is explained in Section 7.13.3.3.10.

The X540 might initiate an LLI 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.

7.3.2.2.2 LLI Parameters

LLI generation is based on the following parameters:

- **LLI Moderation** bit in the EITR registers — When the *LLI Moderation* bit is cleared, any low latency event generates an immediate interrupt. When set, LLI moderation is based on the LLI credit and LLI interval.
- **LLI Credit** field in the EITR register — LLI packets might generate immediate interrupts as long as the LLI credits counter is greater than one (positive credit).

  - The credit counter is incremented by one on each LL interval with a maximum ceiling of 31 credits. It then stops incrementing.
  - If an LLI packet is received and the credit counter is greater than zero, an immediate interrupt is triggered internally. The interrupt is asserted externally when an interrupt is enabled (EIMS setting) and PCI credits are available. Once the interrupt is asserted, the credit counter is decremented by one. Note that the counter never goes below zero.
  - LLI assertion might be delayed due to: interrupt enablement, lack of LLI credits or lack of PCI credits. Each time the interrupt is asserted, the LLI credit is decremented by one regardless of the number of received LLI packets and regardless if the ITR timer expires in the mean time.
  - If an LLI packet is received and the credit counter is zero (no credits), an interrupt can be asserted only on the next LL interval or when the ITR timer expires, whichever comes first.
  - The LLI credit counter is not affected by the ITR timer. Conversely, LLI assertion initializes the ITR timer to its timer interval.
  - Note that during nominal operation software may not need to access the LL credit field.

- **LL interval** is defined in units of 4 µs (at 1 Gb/s or 10 Gb/s link) in the GPIE register. At 100 Mb/s link speed, the LL interval is defined in units of 40 µs. This parameter defines the clock that increments the LLI credit counter. The maximum rate of the LLI interrupts per second is bound by the LL interval, which equals to 1/LL Interval. When LLI moderation is enabled, the ITR interval of the same interrupt must be greater than the LL interval.

7.3.2.2.3 LLIs Affect on RSC Functionality

LLI packet reception requires instant CPU processing. Software might be able to access a specific descriptor only if all its proceeding descriptors complete. If RSC's are enabled, some of the preceding descriptors might be incomplete at the time that the LLI packet is received. Hardware overcomes this problem by:

- Following LLI packet completion, all RSC's on the same queue are completed as well.
- Then, the associated 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 this case, it is guaranteed that an additional interrupt is asserted when the ITR expires.

7.3.3 TCP Timer Interrupt

7.3.3.1 Introduction

In order to implement TCP timers for IOAT, 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.

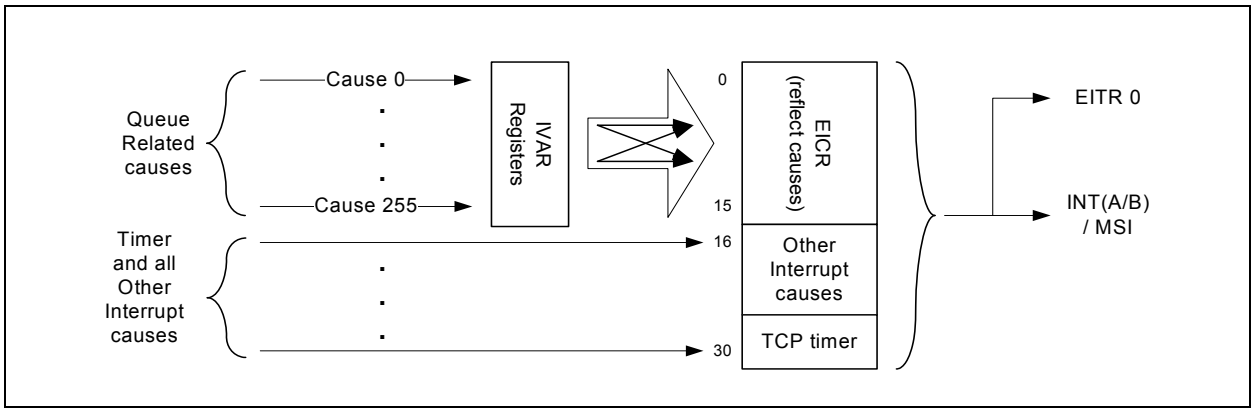


Figure 7-23 Cause Mapping in Legacy and MSI Modes

Mapping between the Tx and Rx queue to the IVAR registers is hardwired as shown in the Figure 7-24:

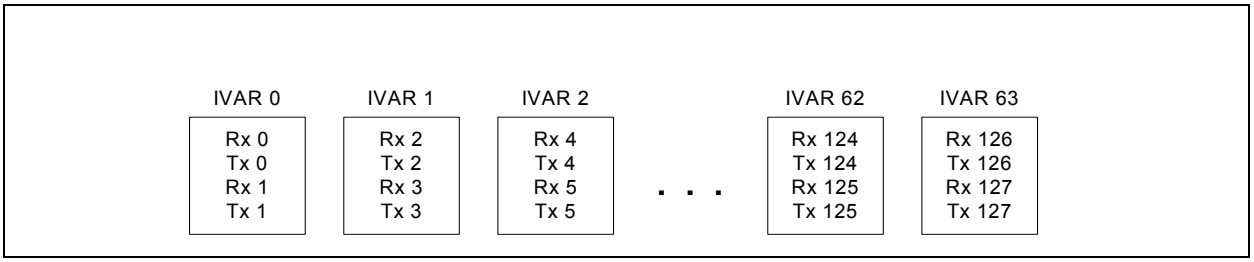


Figure 7-24 Rx and Tx Queue Mapping to IVAR Registers

7.3.4.2 MSI-X Mode in Non-IOV Mode

MSI-X defines a separate optional extension to basic MSI functionality. The number of requested MSI-X vectors is loaded from the MSI_X_N fields in the NVM up to maximum of 64 MSI-X vectors.

- Hardware indicates the number of requested MSI-X vectors in the table size in the MSI-X capability structure in the configuration space. This parameter is loaded from the *MSI_X _N* field in the NVM. 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 (n 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, EIAC, EIAM mirror the lower 16 bits of EICS(1), EIMS(1), EIMC(1), EIAC(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 then 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 SEIC, EICS... 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.

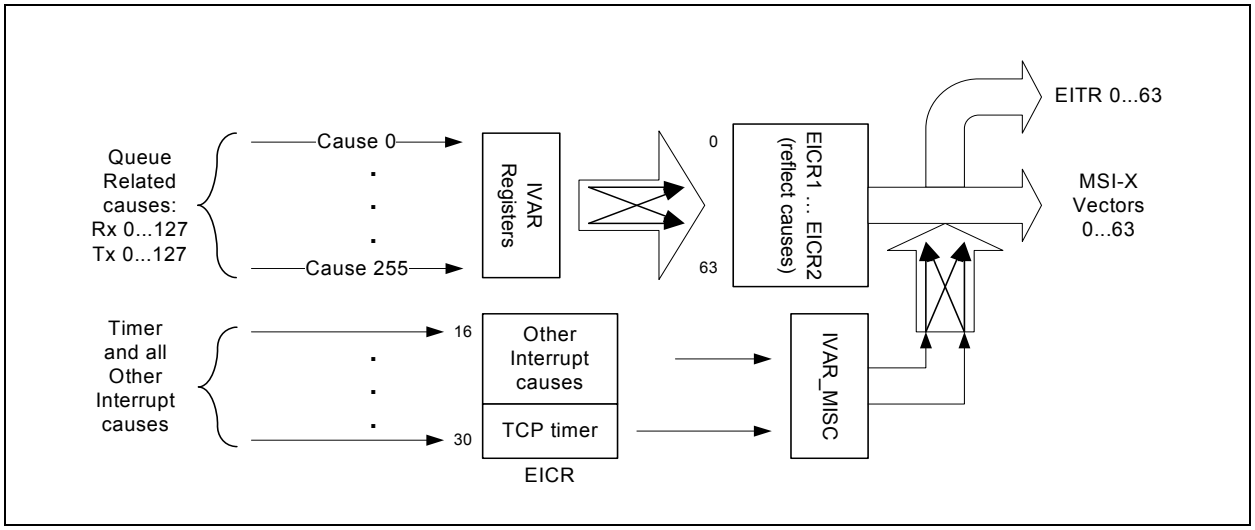
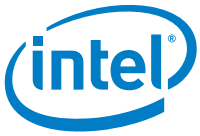


Figure 7-25 Cause Mapping in MSI-X Mode (non-IOV)



7.3.4.3 MSI-X Interrupts In IOV Mode

In IOV mode, interrupts must be implemented by MSI-X vectors. The X540 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 the *MSI-X Table* field in the NVM. 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 for the PF is loaded from the *MSI_X_N* fields in the NVM 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.

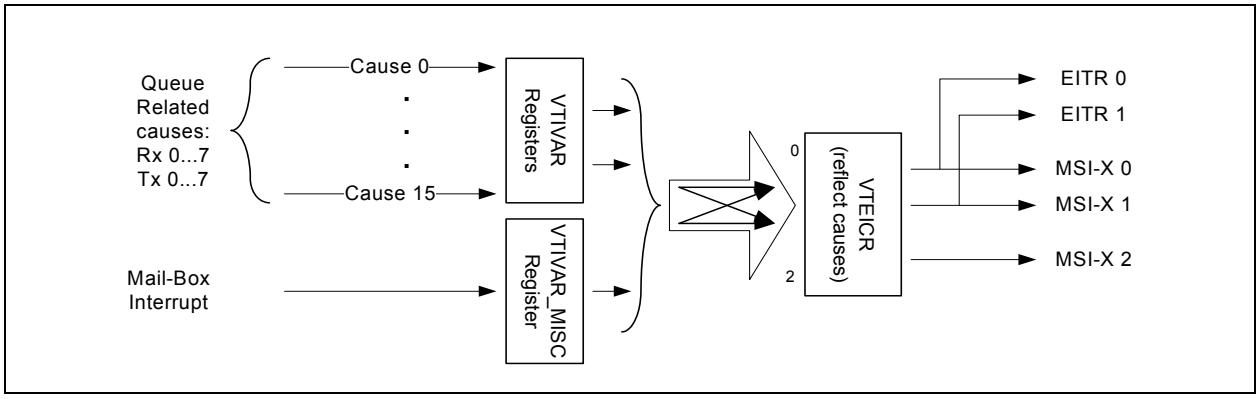


Figure 7-26 VF Interrupt Cause Mapping (MSI-X, IOV)

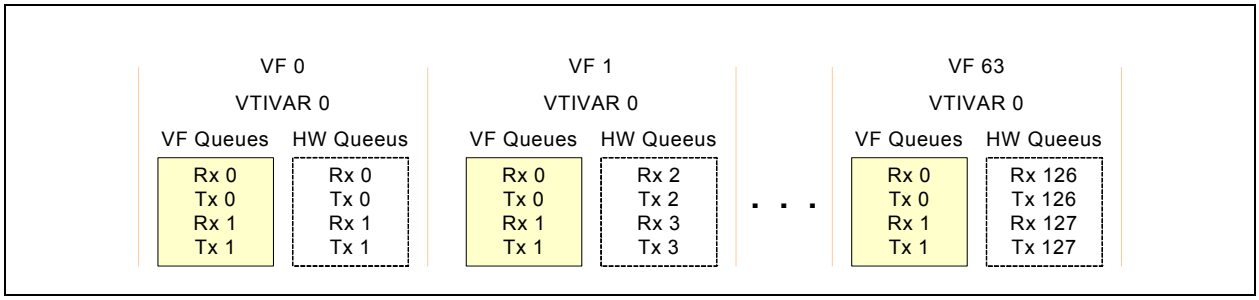


Figure 7-27 VF Mapping of Rx and Tx Queue to VFIVAR in 64 VF's Mode

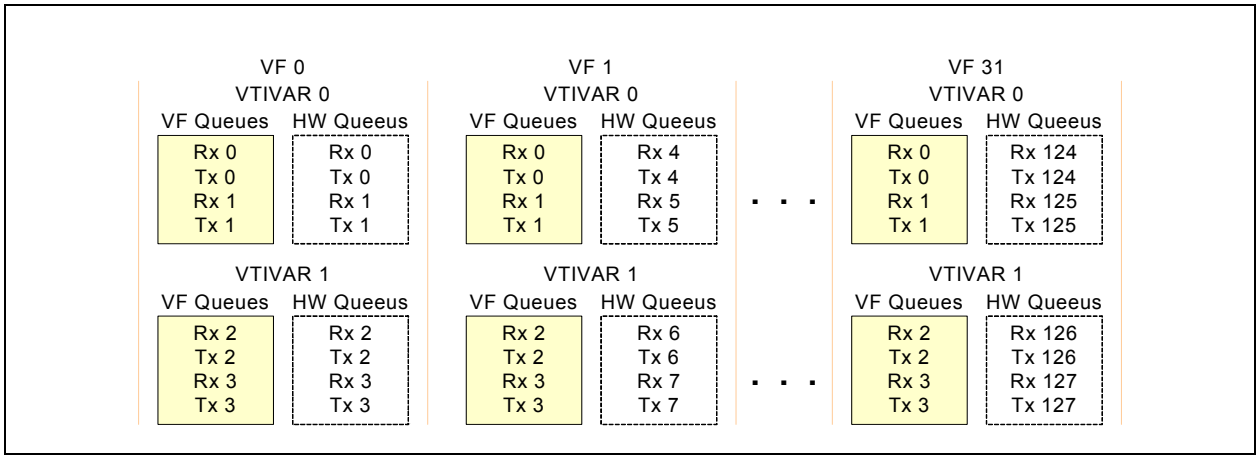


Figure 7-28 VF Mapping of Rx and Tx Queue to VFIVAR in 32 VF's Mode

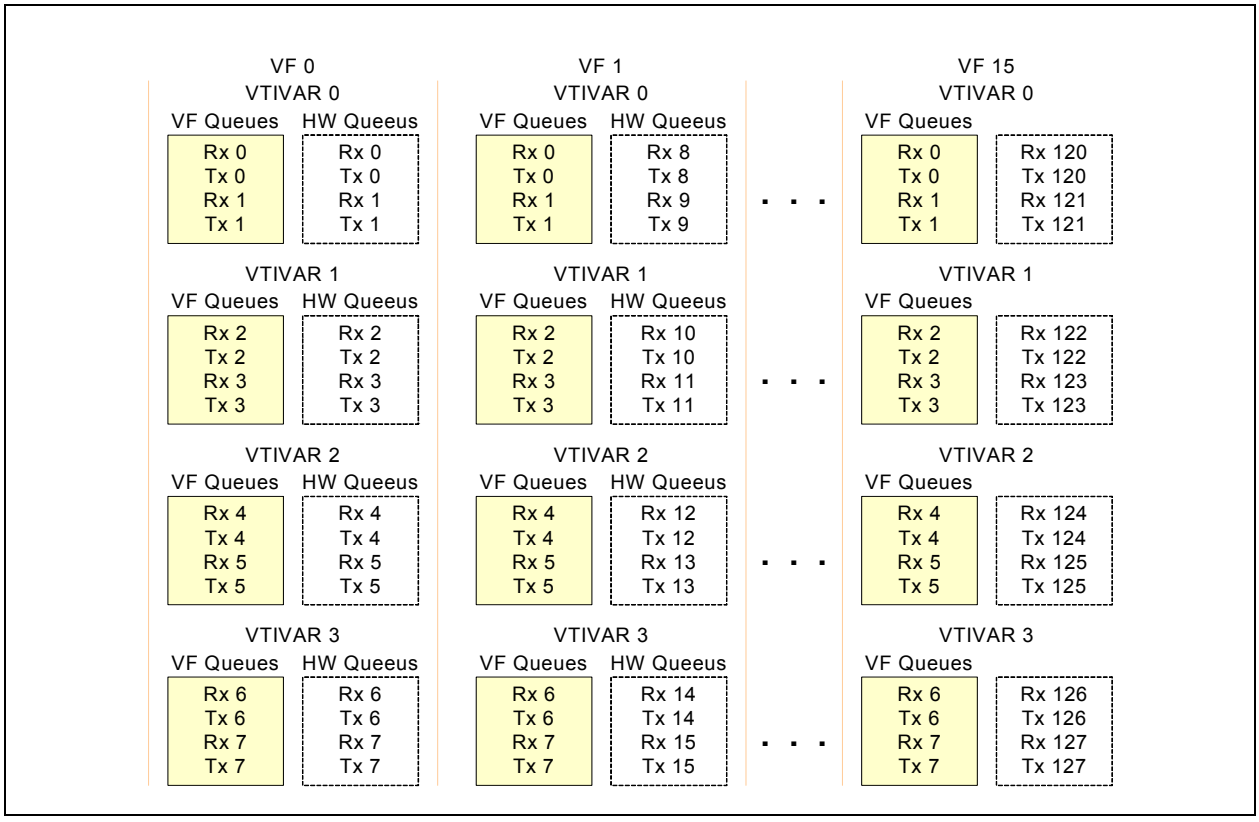


Figure 7-29 VF Mapping of Rx and Tx Queue to VFIVAR in 16 VF's Mode

7.3.4.4.3 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 7-48 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.

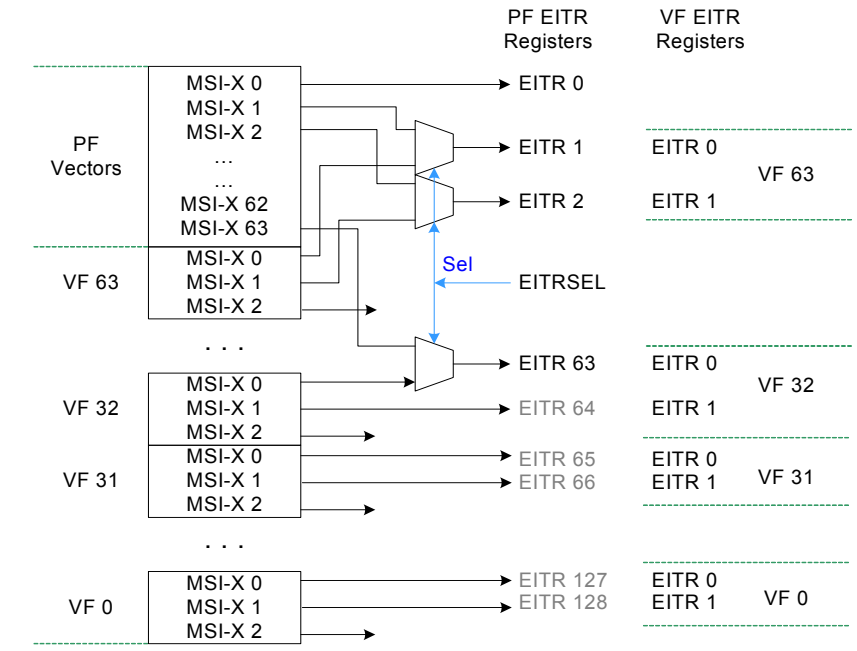
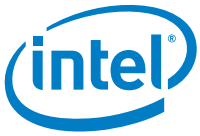


Figure 7-30 PF / VF MSI-X Vectors Mapping to EITR

Table 7-48 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 X540 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)
- Canonical Form Indicator (CFI). 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 *CFIEN* and *CFI* bits in the VLNCTRL
- VLAN Identifier (VID)

Octet 1								Octet 2							
UP			CFI	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 X540 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

Software might instruct the X540 to insert an 802.1q VLAN tag on a per-packet basis. If the *VLE* bit in the transmit descriptor is set to 1b, then the X540 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.

7.4.3.2 Stripping 802.1q Tags on Receives

Software might instruct the X540 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), then the X540 strips the 4-byte VLAN tag from the packet, and stores the TCI in the *VLAN Tag* field of the receive descriptor.

The X540 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.

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 then 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, *CFIEN* and *CFI*, are also used in conjunction with 802.1q VLAN filtering operations. *CFIEN* enables the comparison of the value of the *CFI* bit in the 802.1q packet to the Receive Control register *CFI* 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 X540 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 register.

Cross Functionality with Manageability

The X540 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 X540 is set to double VLAN mode as well as the VLAN Ethertype(s). When operating in a double VLAN mode, control packets sent by the manageability controller with no VLAN headers should not activate any hardware offload other than MACsec encapsulation.

Table 7-49 Transmit Handling of Packets with VLAN Header(s)

MAC Address	Outer VLAN	Inner VLAN	L2 Payload	Ethernet CRC

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.

Transmit Functionality on the Inner VLAN Header

- The inner VLAN header can be added by software in one of the following methods:
  - The header is included in the transmit data buffers.
  - The 16-bit portion of the header that includes the priority tag, CFI and VLAN ID are included in the transmit descriptor. The VLAN Ethertype is taken from the *VT* field in the DMATXCTL register.
  - In IOV mode, the priority tag, CFI and VLAN ID can be taken from the PFVMVIR (see details in Section 7.10.3.9.2)
- Hardware identifies and skips the VLAN header for parsing inner fields.
- DCB — The user priority of the packet is taken from the inner VLAN. 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 7.10.3.4 and Section 7.10.3.9.2.

7.4.5.1 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 other than MACsec processing
          }
          Else
          {
                    This is the case of no VLAN. Process the frame (ignore
                    outer and inner VLAN processing)
          }
}
```

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.

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.
- If the RXDCTL.VME is set, the inner VLAN is stripped by hardware while the priority tag, CFI and VLAN ID are indicated in the *VLAN Tag* field 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 7.10.3.3.
- DCB — The user priority of the packet is taken from the inner VLAN. In the absence of inner VLAN, the packet is assumed as user priority 0 (least priority). See Section 7.4.5.2 for the absence of any VLAN headers.

7.4.5.2 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.

Transmitted 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 other than MACsec.

Receive Functionality

DCB — Assume user priority 0 (lowest priority).

Receive offload functionality — pool and queue are selected by the Ethernet MAC address or ETQF/ETQS registers. MACsec offload is functional. Filtering to host and manageability remains functional.

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.

7.4.5.3 Packet Priority in Single and Double VLAN Modes

This section summarizes packet handling in both single and double VLAN modes. The user priority of a packet is meaningful in DCB mode when multiple traffic classes are enabled as well as LLIs. The user priority is extracted from the packets as listed in the following table.

Table 7-50 Packet Handling in Single and Double VLAN Modes

Packet type	Single VLAN	Double VLAN
Packet with no VLAN	User priority = 0	User priority = 0
Packet with 1 VLAN	The user priority field in the VLAN header in the packet	User priority = 0
Packet with 2 VLANs	Erroneous case: The user priority field in the **outer** VLAN header in the packet	The user priority field in the **inner** VLAN header in the packet

7.5 Direct Cache Access (DCA)

DCA is a method to improve network I/O performance by placing some posted inbound writes directly within CPU cache. DCA potentially eliminates cache misses due to inbound writes.

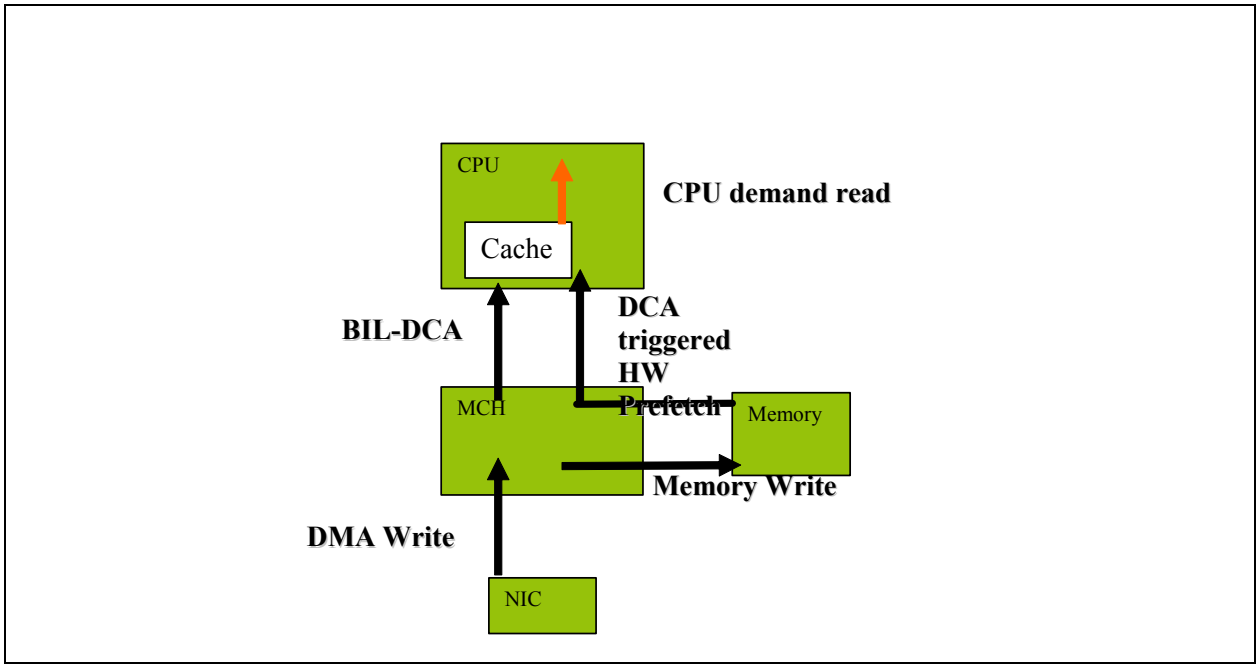


Figure 7-31 Diagram of DCA Implementation on FSB System

As Figure 7-31 illustrates, DCA provides a mechanism where the posted write data from an I/O device, such as an Ethernet NIC, can be placed into CPU cache with a hardware pre-fetch. This mechanism is initialized upon a power good reset. A device driver for the I/O device configures the I/O device for DCA and sets up the appropriate CPU ID and bus ID for the device to send data. The device then encapsulates that information in PCIe TLP headers, in the tag field, to trigger a hardware pre-fetch to the CPU cache.

DCA implementation is controlled by separate registers (DCA_RXCTRL and DCA_TXCTRL) for each transmit and receive queue. In addition, a DCA disable bit can be found in the DCA_CTRL register, and a DCA_ID register can be found for each port, in order to make the function, device, and bus numbers visible to the driver.

The DCA_RXCTRL and DCA_TXCTRL registers can be written by software on the fly and can be changed at any time. When software changes the register contents, hardware applies changes only after all the previous packets in progress for DCA have completed.

However, in order to implement DCA, the X540 has to be aware of the IOAT version used. The software driver initializes the X540 to be aware of the bus configuration. The *DCA Mode* field in the DCA_CTRL register defines the system configuration:

1. Legacy DCA: The DCA target ID is derived from CPU ID.
2. DCA 1.0: The DCA target ID is derived from APIC ID.

Both modes are described as follows.

7.5.1 PCIe TLP Format for DCA

Figure 7-32 shows the format of the PCIe TLP for DCA.

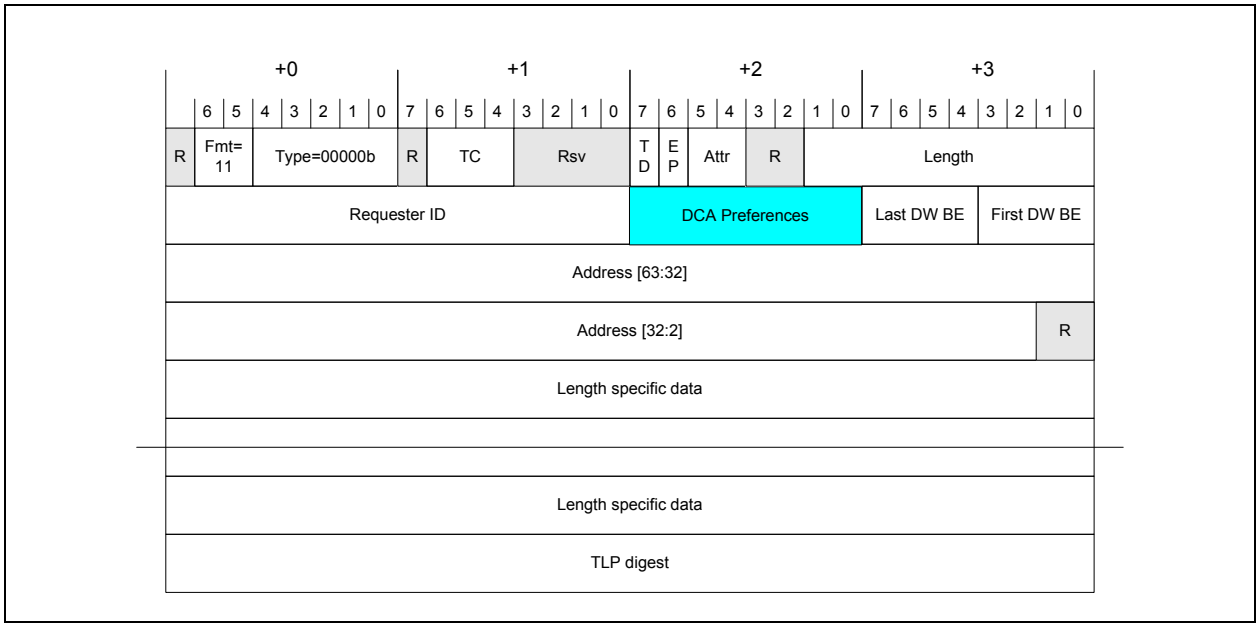


Figure 7-32 PCIe Message Format for DCA

The DCA preferences field has the following formats.

For legacy DCA systems:

Bits	Name	Description
0	DCA indication	0b = DCA disabled. 1b = DCA enabled.
4:1	DCA target ID	The DCA target ID specifies the target cache for the data.

For DCA 1.0 systems:

Bits	Name	Description
7:0	DCA target ID	0000.0000b: DCA is disabled. Other: Target core ID derived from APIC ID.

Note: All functions within a the X540 have to adhere to the tag encoding rules for DCA writes. Even if a given function is not capable of DCA, but other functions are capable of DCA, memory writes from the non-DCA function must set the tag field to 00000000b.

7.6 LEDs

The X540 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 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 section.

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) 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.

LINK/ACTIVITY mode functions slightly different than others irrespective of 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 following mapping is used to specify the LED control source (MODE) for each LED output:

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.
0110	LINK_100	Asserted or blinking according to the LEDx_BLINK setting when a 100M link is established and maintained.

0111b:1101b	Reserved	Reserved.
1110b	LED_ON	Always asserted or blinking according to the LEDx_BLINK setting.
1111b	LED_OFF	Always de-asserted.

7.7 Data Center Bridging (DCB)

See Section 4.6.11 for the DCB configuration sequence and Section 12.5 for the expected performance of DCB functionality.

7.7.1 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:

1. 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.
2. 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.
3. 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 X540 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.
4. 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.

5. 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 X540 are:

1. Multi-class priority arbitration and scheduling — The X540 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 X540, TCs and user priorities are processed on a packet-by-packet basis based on the 802.1p identifier in the 802.1Q-tag.
2. 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.
3. 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 X540 implements 128 descriptor queue in transmit and 128 descriptor queues in receive.
4. Multiple Buffers — The X540 implements separate transmit and receive packet buffers per TC.
5. 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 7-33 shows the latency requirements as previously defined.

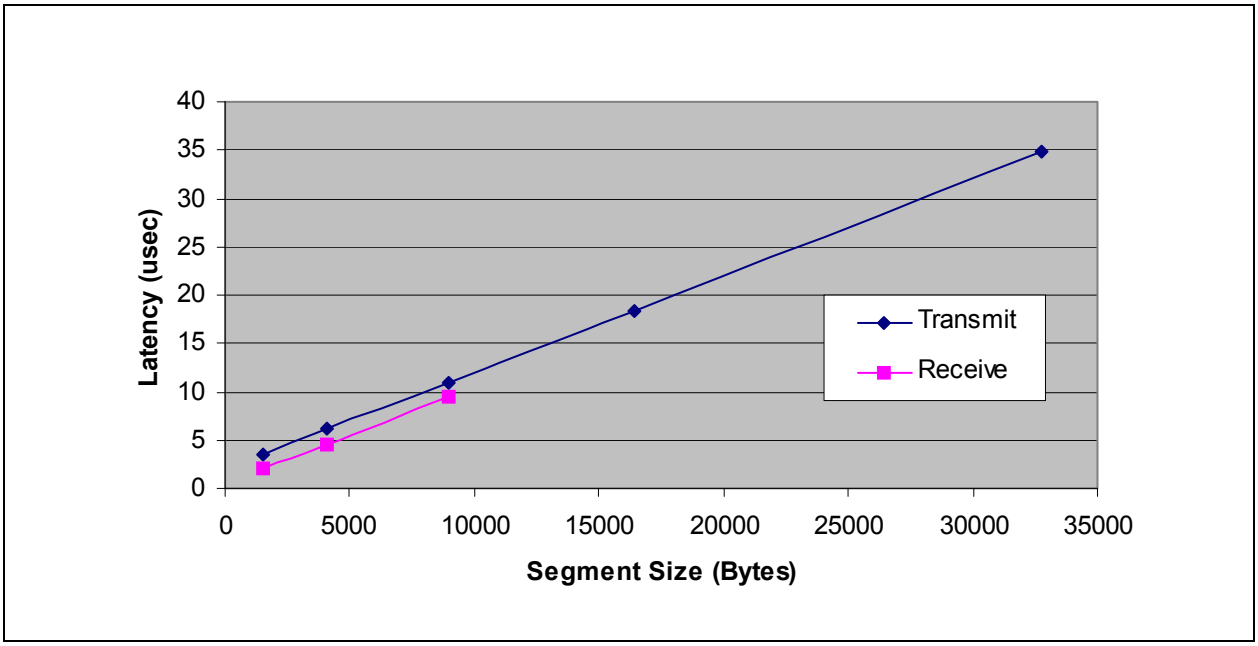


Figure 7-33 Latency Requirements

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 untagged. GMRP frames (special Ethernet MAC addresses starting with 0x0180C20000) must, however, go tagged. Untagged packets must be delivered to the host and are assumed to belong to User Priority 7.

Note that any QCN signaling is terminated at thenetwork's edge. At initialization, everycomponent exchanges itscapabilities with its peer via aCapability Exchange (DCX) protocol carried over dedicated Link Layer Discovery Protocol (LLDP) frames. Support for these protocols is transparent to the hardware implementation, and as a result, is not described in this document.

7.7.2 Transmit-side Capabilities

7.7.2.1 Transmit Rate Scheduler (RS)

Note: When configured for DCB mode or when using the Tx rate-limiting functionality, the X540 software driver should only use advanced transmit descriptors. Refer to Section 7.2.3.2.3 for more details.

7.7.2.1.1 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. 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 7-34 for visualizing the effect of MMW.

MMW_SIZE set to 0b must be supported as well.

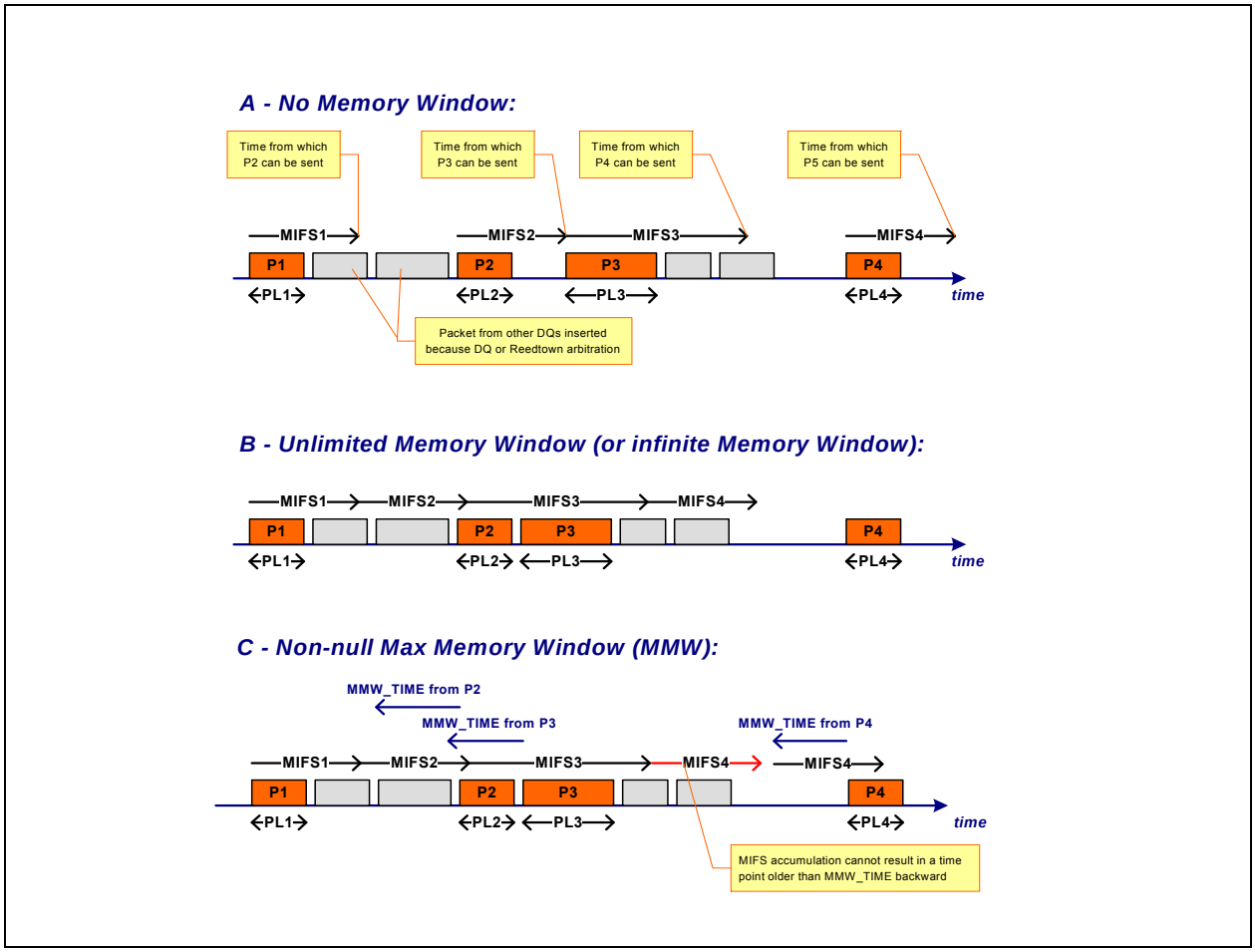


Figure 7-34 Minimum Inter-Frame Spacing for Rate-Controlled Frames (in Orange)

7.7.2.2 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 X540 is required to use the same UP-to-TC Tx mapping table that software is using. The RTTUP2TC register must be configured by software accordingly. Refer to Section 3.6.5.1.3 for details on priority flow control.

7.7.2.3 VM-Weighted Round-Robin Arbiters

The X540 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 7.10.

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.

7.7.2.3.1 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 X540 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 X540 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
† Due to a pipelined implementation, the VM credits range is enlarged by one MSS, beyond negative limits.			
†† 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 X540'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 X540'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 in order to guaranty transmission of at least one frame on each recycling round. The X540 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 — In order 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

Table 7-52 Registers Allocation for Tx VM Arbiters

Attribute	Tx VM Arbiter
VMC Status registers	RTTDT1S
VM credit refill	CRQ
VM credits	CCC

7.7.2.3.2 WRR Arbiter Algorithm

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.

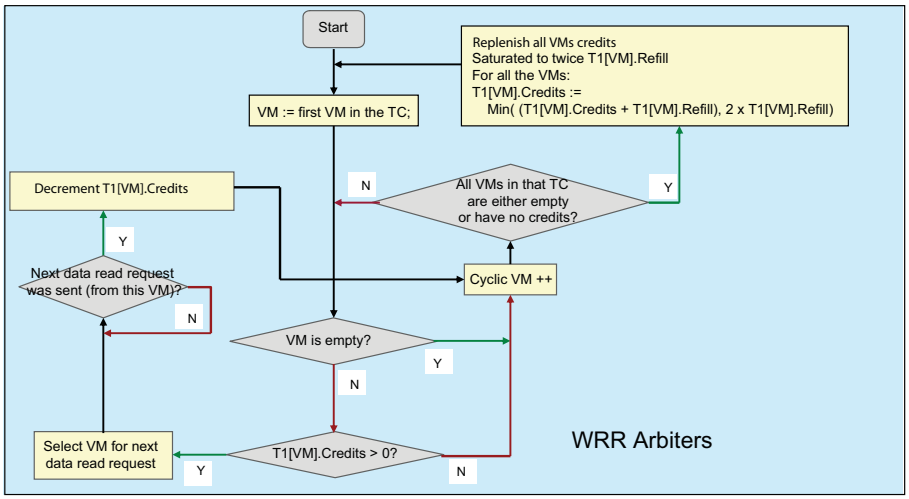


Figure 7-35 Tx VM WRR Arbiters Operation

7.7.2.4 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 X540 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 X540 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.

7.7.2.4.1 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 X540 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 X540 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 X540 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

† Due to a pipelined implementation, the TC credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.

†† 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 X540). TC7 is the highest priority TC.

BWG: Configuration — Traffic BWG that a TC belongs to.

TC Credit Refill: Configuration — The X540'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 X540'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 X540 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 — In order 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, then 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, then 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 TC's with bandwidth allocation and TC's 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 TC's. To guarantee correct bandwidth allocation, all TC's 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. TC's with the *LSP* bit set should be the first to be considered by the scheduler (the first TC's). For example, from queue 7 to queue 5 with the other five TC's 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 TC's with the unlimited bit set effectively has zero BWG credits since TC's with the LSP unlimited bit set should have TC credits set to zero. Use of LSP / TC.GSP should be restricted to UP/TC's 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 TC's 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				

† 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.

7.7.2.4.2 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

7.7.2.4.3 Tx TC WSP Arbitration Configurations

RR / WSP: Global Configuration bits — 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.

Recycle Mode: Global Configuration bits —

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, then 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.

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

7.7.2.4.4 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 can't 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.

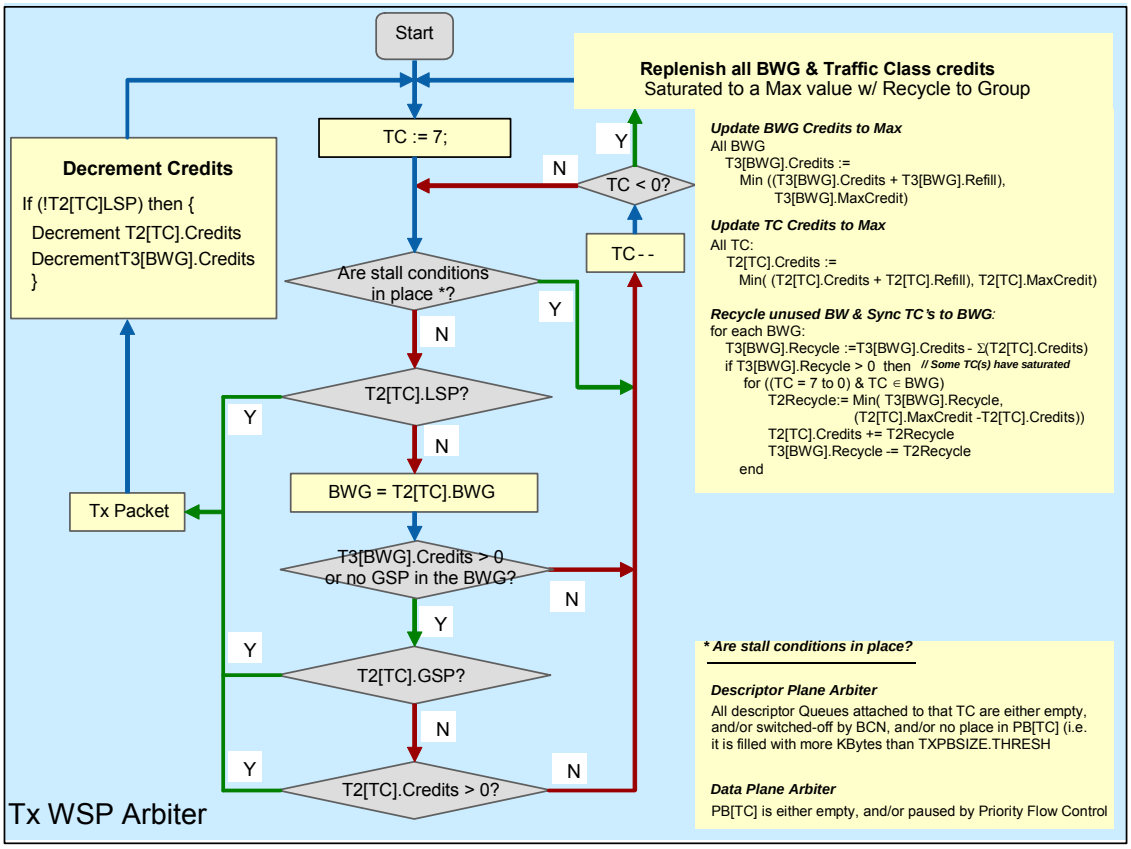


Figure 7-36 Tx TC WSP Arbiters Operation

7.7.3 Receive-side Capabilities

7.7.3.1 User Priority to Traffic Class Mapping

To enable different TC support for incoming packets and proper behavior per TC, the X540's receive packet buffer is segmented into several packet buffers. Refer to Section 3.6.5.3.5 for the different packet buffer configurations supported.

Incoming packets are transferred from the packet buffers into data buffers in system memory. Data buffers are arranged around descriptor rings described in Section 7.1.9. Each descriptor queue is assigned dynamically to a given TC (and therefore to a packet buffer) as described in Section 7.1.2.

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 X540 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.

7.7.3.2 Rx PB Weighted Strict Priority Arbiter

The X540'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 X540 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

† Due to a pipelined implementation, the PB credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.

†† 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	-/+ 2,048KB	IPC
1	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048KB	SAN
2	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048KB	LAN
3	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048KB	MGMT.
-				
-				
-				
7				

The attributes of the Rx packet arbiter are described in Section 7.7.2.4.2.

7.7.3.2.1 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 7.7.2.4.1.

7.7.3.2.2 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.

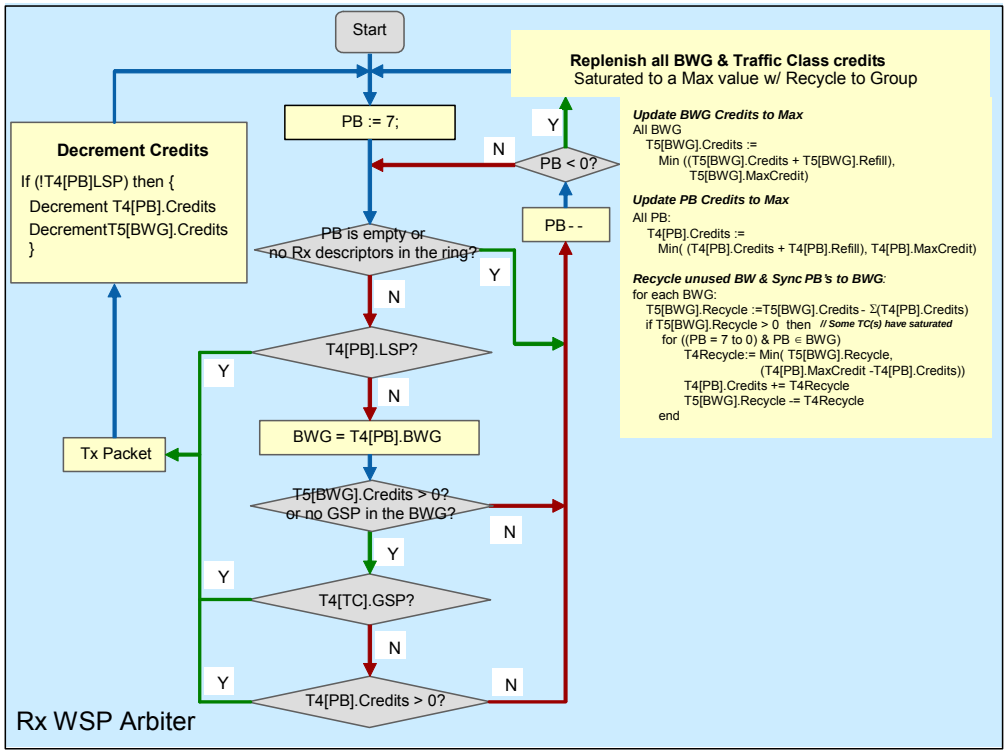


Figure 7-37 Rx Packet WSP Arbiter Operation

Note: This page intentionally left blank.

7.8 MACsec 802.1AE

MACsec, 802.1AE is a MAC-level encryption/authentication scheme defined in IEEE 802.1AE that uses symmetric cryptography. The 802.1AE defines AES-GCM 128-bit key as a mandatory cipher suite that can be processed by the X540. The MACsec implementation enabled as detailed in Section 4.6.12, supports the following:

- GCM AES 128-bit offload engine in the Tx and Rx data path that support 10 Gb/s wire speed.
- Both host and manageability controller traffic can be processed by the GCM AES engines.
- Support a single, secure Connectivity Association (CA):
  - Single Secure Connection (SC) on transmit data path.
  - Single SC on receive data path.
  - Each SC supports two Security Associations (SAs) for seamless re-keying.
- At any given time, either the manageability controller or the host can act as key agreement entity (KaY – in 802.1AE spec terminology). For example, control and access the offloading engine (SecY in 802.1AE specification terminology).
  - Arbitration semaphores indicate whether the manageability controller or the host acts as the KaY.
  - Tamper resistance — When the manageability controller acts as KaY it can disable accesses from the host to SecY's address space. When the host acts as the KaY no protection is provided.
- Provide statistic counters.
- Support replay protection with replay window equal to zero. Packets that fail replay validation are posted with a replay error in the Rx descriptor. The packets are posted to the host regardless of strict versus check mode described later on in this section.
- Receive memory structure:
  - New MACsec offload receive status indication in the receive descriptors. MACsec offload must not be used with the legacy receive format but rather use the extended receive descriptor format.
  - MACsec header/tag can be posted to the KaY for debug.
- Support VLAN header location according to IEEE 802.1AE (first header inner to the MACsec tag).
- When MACsec offload is enabled, Ethernet CRC must be enabled as well by setting both TXCRCEN and RXCRCSTRP bits in the HLREG0 register.

7.8.1 Packet Format

MACsec defines frame encapsulation format as follows.

Table 7-61 Legacy Frame Format

MAC DA, SA	VLAN (optional)	Legacy Type / Len	LLC data (might include IP/TCP and higher level payload)	CRC
	✍ — - — - — - — - — - — - — - — - — - — - User Data — - — - — - — - — - — - — - — - — - — - — - — - — - — - — - — - — - — - ⑦			

Table 7-62 MACsec Encapsulation

MAC DA, SA	MACsec header (SecTag)	User data (optional encrypted)	MACsec ICV (tag)	CRC

7.8.2 MACsec Header (SecTag) Format

Table 7-63 Sectag Format

MACsec Ethertype	TCI and AN	SL	PN	SCI (optional)
2 bytes	1 byte	1 byte	4 bytes	8 bytes

7.8.2.1 MACsec Ethertype

The MACsec Ethertype comprises octet 1 and octet 2 of the SecTAG. It is included to allow:

a. Coexistence of MACsec capable systems in the same environment as other systems.

b. Incremental deployment of MACsec capable systems.

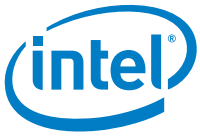
c. Peer SecY's to communicate using the same media as other communicating entities.

d. Concurrent operation of key agreement protocols that are independent of the MACsec protocol and the current cipher suite.

e. Operation of other protocols and entities that make use of the service provided by the SecY's uncontrolled port to communicate independently of the Key agreement state.

Table 7-64 MACsec Ethertype

Tag Type	Name	Value
802.1AE security TAG	MACsec Ethertype	88-E5



7.8.2.2 TCI and AN

Table 7-65 TCI and AN Description

Bit(s)	Description
7	**Version Number (V).** the X540 supports only version 0. Packets with other version value are discarded by the X540.
6	**End Station (ES).** When set, indicates that the sender is an end station. As a result, SCI is redundant and causes the SC bit to be cleared. Currently, should be always 0b.
5	**Secure Channel (SC).** Secure Channel (SC). Equals 1b when the SCI field is active. If the ES bit is set the SC must be cleared. Since only ES equals zero is supported, the SCI field must be active by setting the LSECTXCTRL.AISCI.
4	**Single Copy Broadcast (SCB).** Cleared to 0b unless SC supports EPON. Should always be 0b.
3	**Encryption (E).** Set to 1b when user data is encrypted. (see the note that follows).
2	**Changed Text (C).** Set to 1b if the data portion is modified by the integrity algorithm. For example, if non-default integrity algorithm is used or if packet is encrypted. (see the note that follows).
1:0	**Association Number (AN).** 2-bit value defined by control channel to uniquely identify SA (Keys, etc.).

Note: The combination of the *E* bit equals 1b and the *C* bit equals 0b is reserved for KaY packets. The MACsec logic ignores these packets on the receive path and transfers them to KaY as is (no MACsec processing and no MACsec header strip). the X540's implementation never issues a packet in which the *E* bit is cleared and the *C* bit is set, although can tolerate such packets on receive.

7.8.2.2.1 Short Length

Table 7-66 SL Field Description

Bit(s)	Description
7:6	Reserved, set to 0b.
5:0	**Short Length (SL).** Number of octets in the secure data field from the end of SecTag to the beginning of ICV if it is less then 48 octets, else SL value is 0b.

7.8.2.2.2 Packet Number (PN)

The MACsec engine increments it for each packet on the transmit side. The PN is used to generate the initial value (IV) for the crypto engines. When KaY is establishing a new SA, it should set the initial value of PN to 1b. See more details on PN exhausting in Section 7.8.5.1.

7.8.2.3 Secure Channel Identifier (SCI)

The SCI is composed of the Ethernet MAC address and port number as listed in the following table. If the *SC* bit in TCI is not set, the SCI is not encoded in the SecTag.

Table 7-67 SCI Field Description

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Source Ethernet MAC Address						Port Number	

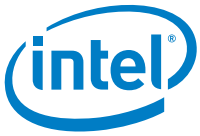
7.8.2.4 Initial Value (IV) Calculation

The IV is the initial value used by the Tx and Rx authentication engines. The IV is generated from the PN and SCI as described in the 802.1AE specification.

7.8.3 MACsec Management – KaY (Key Agreement Entity)

Kay management is done by the host or the manageability controller. The ownership of MACsec management is as follows:

1. Initialization at power up or after wake on LAN.
   - In most cases the manageability controller wakes before the host, so:
     - — If the manageability controller can be a KaY, it establishes an SC (authentication and key exchange).
     - — If the manageability controller cannot be a KaY the only way for it to communicate is through a dedicated Ethernet MAC address or VLAN. This means that the switch must support settings that enable specific Ethernet MAC Address or VLAN to bypass MACsec.
   - When the host is awake:
     - — If the manageability controller acted as KaY, the host should authenticate itself and transfer its ability to authenticate to the manageability controller in order for the manageability controller to transfer ownership over the MACsec hardware. At this stage, the system operates in proxy mode where the host manages the secured channel while the manageability controller piggybacks on it.
     - — If the manageability controller wasn't KaY, the host takes ownership over the MACsec hardware and establishes an SC (authentication and key exchange). The manageability controller mode of operation does not change and it continues to communicate through a dedicated Ethernet MAC address or VLAN.
2. Host in Sx state — manageability controller active:
   - — If the manageability controller is not Kay capable, then the SC should be reset by link reset or by sending a log-off packet (1af) and then the manageability controller can return to VLAN solution (or remain in such).



— If the manageability controller is KaY capable, the host should notify the manageability controller that it retires KaY ownership and the manageability controller should retake it.

7.8.4 Receive Flow

The X540 might concurrently receive packets that contain MACsec encapsulation as well as packets that do not include MACsec encapsulation. This section describes the incoming packet classification.

- Examine the user data for a SecTAG:
  - — If no SecTag, post the packet with a cleared *MACsec* bit in the *Packet Type* field in the receive descriptor.
- Validate frames with a SecTAG:
  - — The MPDU comprises at least 18 octets
  - — Octets 1 and 2 compose the MACsec Ethertype (88E5)
  - — The *V* bit in the TCI is cleared
  - — If the *ES* or the *SCB* bit in the TCI is set, then the *SC* bit is cleared
  - — Bits 7 and 8 of octet 4 of the SecTAG are cleared SL <= 0x3F
  - — If the *C* and *SC* bits in the TCI are cleared, the MPDU comprises 24 octets plus the number of octets indicated by the *SL* field if that is non-zero and at least 72 octets otherwise
  - — If the *C* bit is cleared and the *SC* bit set, then the MPDU comprises 32 octets plus the number of octets indicated by the *SL* field if that is non-zero and at least 80 octets otherwise
  - — If the *C* bit is set and the *SC* bit cleared, then the MPDU comprises 8 octets plus the minimum length of the ICV as determined by the cipher suite in use at the receiving SecY, plus the number of octets indicated by the *SL* field if that is non-zero and at least 48 additional octets otherwise
  - — If the *C* and *SC* bits are both set, the frame comprises at least 16 octets plus the minimum length of the ICV as determined by the cipher suite in use at the receiving SecY, plus the number of octets indicated by the *SL* field if that is non-zero and at least 48 additional octets otherwise
- Extract and decode the SecTAG as specified in Section 7.8.2.
- Extract the user data and ICV as specified section Section 7.8.1.
- Assign the frame to an SA:
  - — If a valid SCI, use it to identify the SC
  - — Select SA according to AN value
  - — If no valid SC or no valid SA found, drop the packet
  - — If SCI is omitted, use default SC
  - — Select SA according to AN value
  - — If no valid SC (or more then SC active) or no valid SA found drop packet

- Perform a preliminary replay check against the last validated PN
- Provide the validation function with:
  - The SA Key (SAK)
  - The SCI for the SC used by the SecY to transmit
  - The PN
  - The SecTAG
  - The sequence of octets that compose the secure data
  - The ICV
- Receive the following parameters from the cipher suite validation operation
  - A valid indication, if the integrity check was valid and the user data could be recovered
  - The sequence of octets that compose the user data
- Update the replay check
- Issue an indication to the controlled port with the DA, SA, and priority of the frame as received from the receive de-multiplexer, and the user data provided by the validation operation

Note: All the references to clauses are to the IEEE P802.1AE/D5.1 document from January 19, 2006.

7.8.4.1 Receive Modes

There are four modes of operation defined for MACsec Rx as defined by the LSECRXCTRL.LSRXEN field:

1. Bypass (LSRXEN = 00b) — in this mode, MACsec is not offloaded. There is no authentication or decrypting of the incoming traffic. The MACsec header and trailer are not removed and these packets are forwarded to the host or the manageability controller according to the regular L2 MAC filtering. The packet is considered as untagged (no VLAN filtering). No further offloads are done on MACsec packets.
2. Check (LSRXEN = 01b) — in this mode, incoming packets with matching key are decrypted and authenticated according to the MACsec tag. In this mode, both good and erroneous packets are forwarded to host (with the relevant error indication). The only case where packets are dropped are: erroneous encrypted packets (with the 'C' bit in the SecTag header is set) or erroneous packets with replay error if replay protection is enabled in the LSECRXCTRL registers. This mode is expected to be used mainly for debug purposes. In this mode, it may be useful to set also the "Post MACsec header" bit in the LSECRXCTRL register which controls both SecTag and ICV to be posted to host memory. Note that the header is not removed from KaY packets.
3. Strict (LSRXEN = 10b) — in this mode, incoming packets with matching key are decrypted and authenticated according to the MACsec tag. The MACsec header and trailer might be removed from these packets and the packets are forwarded to the host only if the decrypting or authentication was successful. Additional offloads are possible on MACsec packets. The header is not removed from KaY packets.
4. Drop (LSRXEN = 11b) — in this mode, MACsec is not offloaded and MACsec packets are dropped. There is no authentication or decrypting of the incoming traffic.

7.8.4.2 Receive SA Exhausting – Re-Keying

The seamless re-keying mechanism is explained in the following example.

KaY establishes SC0 SC and sets SA0 as the active SA by writing the key in register MACsec RX Key, writing the AN in LSECRXSA[0], and setting the *SA Valid* bit in the same register. This clears the *Frame Received* bit. On the first packet that arrived to SA0, the frame received automatically sets the *Frame Received* bit. Only at this time the KaY can and should initiate SA1 in the same manner as for SA0. When a frame of SA1 arrives, SA0 retires and can be used for the next SA.

Note: The same mechanism should be used for all RX SCs.

7.8.4.3 Receive SA Context and Identification

Upon arrival of a secured frame the context of the SecTag is verified. This context of the SecTag is described in Section 7.8.2. In order to process the secured frame it should be associated with one of the SA keys. The identification is done by comparing the SCI data with MACsec RX SC registers and the appropriate SC is selected. To ensure that the *SC* bit in the TCI of the frame is not set and more than one SC is valid belongs to the frame considered as erroneous and transferred to error handling if only one SC is valid, this SC is selected in this case SC. The incoming frame AN field is compared to the AN field of the Link RX SA register of the selected SC in order to select an SA. The selected SA PN (register MACsec RX SA PN) field is compared to the incoming PN which should be equal or greater than the MACsec RX SA PN value, otherwise this frame is dropped. On a match, the selected SA key is used for the secured frame processing.

7.8.4.4 Receive Statistic Counters

A detailed list and description of the MACsec Rx statistics counters can found in the MACSec Rx Port Statistics registers section.

7.8.5 Transmit Data Path

The X540 might concurrently transmit packets that contain MACsec encapsulation as well as packets that do not include MACsec encapsulation. This section describes the transmit packet classification, transmit descriptors and statistic counters.

Note: Since flow control (PAUSE) packets are part of the MAC service they should not go through the MACsec logic.

1. Assign the frame to an SA by adding the AN according to SA select bit in the LSECTXSA register.
2. Assign the next PN variable for that SA to be used as the value of the PN in the SecTAG based on the value in the appropriate (according to SA) LSECTXPN register.
3. Encode the octets of the SecTAG according to the setting in LSECTXCTRL register.
4. Provide the protection function of the current cipher suite with:
   a. The SA Key (SAK).

b. The SCI for the SC used by the SecY to transmit.

c. The PN.

d. The SecTAG.

e. The sequence of octets that compose the user data.

5. Receive the following parameters from the cipher suite protection operation:

   a. The sequence of octets that compose the secure data.

   b. The ICV.

6. Issue a request to the transmit multiplexer with the destination and source Ethernet MAC addresses, and priority of the frame as received from the controlled port, and an MPDU comprising the octets of the SecTAG, secure data, and the ICV concatenated in that order.

7.8.5.1 Transmit SA Exhausting – Re-Keying

the X540 supports a single SC on the transmit data path with a seamless re-keying mechanism. The SC might act with one of two optional SAs. The SA is selected statically by the *Active SA* field in the LSECTXSA register. Once the KaY entity (could be either software or hardware as defined by the *MACsec Ownership* field in the LSWFW register) changes the setting of the *SA Select* field in the LSEXTXSA register the *Active SA* field is getting the same value on a packet boundary. The next packet that is processed by the transmit MACsec engine uses the updated SA.

The KaY should switch between the two SAs before the PN is exhausted. In order to protect against such event, hardware generates a MACsec packet number interrupt to KaY when the PN reaches the exhaustion threshold as defined in the LSECTXCTRL register. The exhaustion threshold should be set to a level that enables the KaY to switch between SA's faster then the PN might be exhausted. If the KaY is slower than it should be, then the PN might be increment above planned. Hardware guarantees that the PN never repeats itself, even if the KaY is slow. Once the PN reaches a value of 0xFF…F0, hardware clears the *Enable Tx MACsec* field in the LSECTXCTRL register to 00b. Clearing the *Enable Tx MACsec* field, hardware disables MACsec offload before the PN could wrap around and then might repeat itself.

Note: Potential race conditions are possible as follows. the X540 might fetch a transmit packet (indicated as TxPacketN) from the host memory (host or manageability controller packet). KaY can change the setting of the Tx SA Index. The TxPacketN can use the new TX SA Index if the TX SA index was updated before the TxPacketN propagated to the transmit MACsec engine. This race is not critical since the receiving node should be able to process the previous SA as well as the new SA in the re-keying transition period.

7.8.5.2 Transmit SA Context

Upon transmission of a secured frame, the SA associated data is inserted into the *SecTag* field of the frame. The SecTag data is composed from the MACsec Tx registers. The SCI value is taken from MACsec TX SCI Low and High registers unless instructed to omit SCI. The AN value is taken from the active MACsec TX SA and the PN from the appropriate MACsec TX SA PN.

7.8.5.3 Transmit Statistic Counters

A detailed list and description of the MACsec TX statistics counters can be found in the MACSec Tx Port Statistics registers section.

7.8.6 MACsec and Manageability

See Section 11.4.

7.8.7 Key and Tamper Protection

MACsec provides the network administrator protection to the network infrastructure from hostile or unauthorized devices. Since the local host operating system can itself be compromised, hardware protects vital MACsec context from software access. There are two levels of protection:

- Disable host read access to the MACsec Keys (keys are write-only)
- Disable host access to MACsec logic while the firmware manages the MACsec SC.

7.8.7.1 Key Protection

The MACsec keys are protected against read accesses at all times. Both software and firmware are not able to read back the keys that hardware uses for transmit and receive activity. Instead, hardware enables the software and firmware reading a signature enabling to verify proper programming of the device. The signature is a byte XOR operation of the Tx and Rx keys readable in the LSECTXSUM and LSECRXSUM fields in the LSECCAP register.

7.8.7.2 Tamper Protection

In a scenario where the host failed authentication and as a result cannot act as the KaY, the manageability controller disables the host access to network and manages the MACsec channel while the host operating system is already up and running. In such cases, hardware provides the required hooks to protect MACsec connectivity against hostile software. The manageability controller firmware can disable write accesses generated by the host CPU (on the PCI interface) by setting the *Lock MACsec Logic* (bit 12) bit in the LSWFW register. Setting this bit can generate an interrupt to the host in case it is enabled by the host in the IMS register.

7.8.8 MACsec Statistics

7.8.8.1 Rx Statistics

After receiving a packet, one and only one of the statistics in Table 7-68 applies. The precedence order of the statistics is also defined in Table 7-68.

Table 7-68 Rx Statistics

Register Name	802.1ae Name	Priority	Notes
LSECRXBAD	InPktsBadTag	2	Packet is dropped in strict mode or in check mode when the *C* bit is one.
LSECRXUNSCI	InPktsUnknownSCI	3	Used only in check mode. Packet is forwarded to the host if the *C* bit is zero.
LSECRXNOSCI	InPktsNoSCI	3	Packet is dropped in strict mode or in check mode when the *C* bit is one.
LSECRXUNSA	InPktsUnusedSA	4	Packet is dropped in strict mode or in check mode when the *C* bit is one. Note: This statistic reflects the sum of InPktsUnusedSA for all SAs.
LSECRXNUSA	InPktsNotUsingSA	4	Used only in check mode. Packet is forwarded to the host if the *C* bit is zero. Note: This statistic reflects the sum of InPktsUnusedSA for all SAs.
LSECRXLATE	InPktsLate	5	
n/a	InPktsOverrun	n/a	The X540 supports wire-speed decryption and thus this statistic is not needed.
LSECRXNV[SA#]	InPktsNotValid	6	Packet is dropped in strict mode or in check mode when the *C* bit is one.
LSECRXINV[SA#]	InPktsInvalid	6	Used only in check mode. Packet is forwarded to the host if the *C* bit is zero.
LSECRXDELAY	InPktsDelayed	7	
GPRC	InPktsUnchecked	n/a	This statistic is relevant only in bypass mode. In this case, this statistic is reflected in the regular GPRC statistic.
LSECRXOK[SA#]	InPktsOK	8	

Note: This page intentionally left blank.

7.9 Time SYNC (IEEE1588 and 802.1AS)

7.9.1 Overview

Measurement and control applications are increasingly using distributed system technologies such as network communication, local computing, and distributed objects. Many of these applications are enhanced by having an accurate system-wide sense of time achieved by having local clocks in each sensor, actuator, or other system device. Without a standardized protocol for synchronizing these clocks, it is unlikely that the benefits are realized in the multi-vendor system component market. Existing protocols for clock synchronization are not optimum for these applications. For example, Network Time Protocol (NTP) targets large distributed computing systems with Millisecond (ms) synchronization requirements. The 1588 standard specifically addresses the needs of measurement and control systems:

- Spatially localized
- Microsecond (μs) to sub-μs accuracy
- Administration free
- Accessible for both high-end devices and low-cost, low-end devices

Note: The time sync mechanism activation is possible in full-duplex mode only and is not supported at 100 Mb/s link speed.

7.9.2 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 in order 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 7-38.

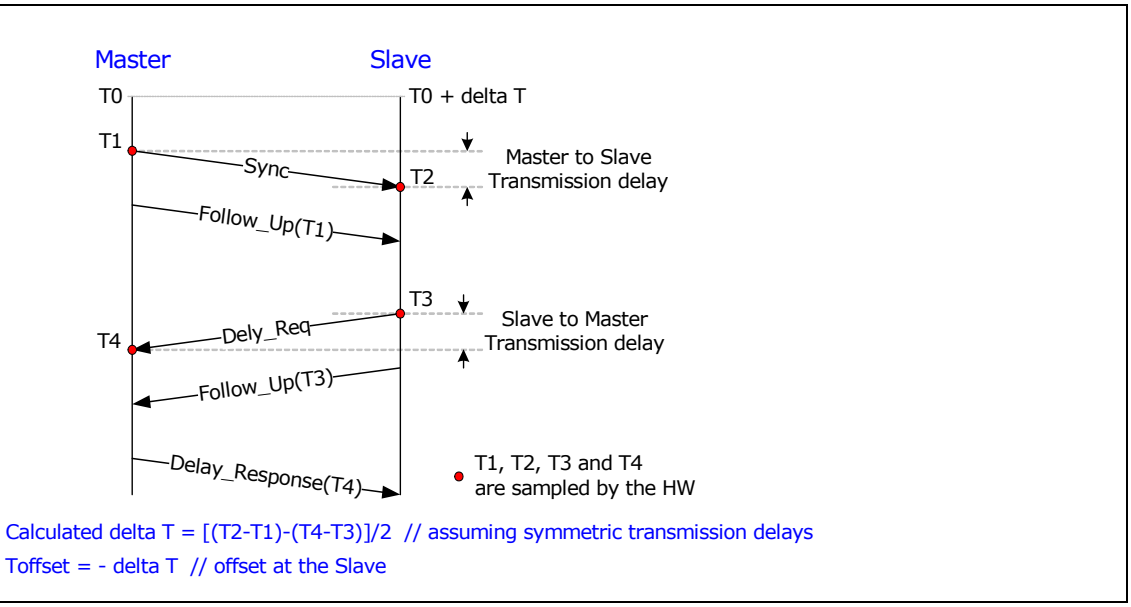


Figure 7-38 Sync Flow and Offset Calculation

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 and store the sourceID and sequenceID in registers (T2).	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 and store the sourceID and sequenceID in registers (T4).	Hardware	Master
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

7.9.2.1 TimeSync 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 don't 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 TS (bit 14) 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.

7.9.3 Hardware Time Sync Elements

All time sync hardware elements are reset to their initial values (as defined in Section 8.0) upon Master reset. The clock driving the time sync elements is the DMA clock which its frequency depends on the link speed (as shown in Table 7-69). Upon change in link speed some of the time sync parameters should be changed accordingly.

7.9.3.1 System Time Structure and Mode of Operation

The time sync logic contains an up counter to maintain the system time value. This is a 64-bit counter that is built from the SYSTIMEL and SYSTIMEH registers. When operating as a master, the SYSTIME registers are usually set once by software according to a general system clock. When operating as a slave, software should update the SYSTIME registers on every sync event as described in Section 7.9.3.3. Read access to the SYSTIMEH and SYSTIMEL registers should execute in the following manner:

1. Software reads register SYSTIMEL, at this stage hardware should latch the value of SYSTIMEH.
2. Software reads register SYSTIMEH, the latched value from the last read of SYSTIMEL.

Each DMA clock the system time registers are incremented by TIMEINCA.*incvalue*. The *incvalue* defines the granularity of the SYSTIME registers. For example, if the DMA clock cycle time were 16 ns and the *incvalue* was 16 then the time was represented in nanoseconds units, and if the *incvalue* was 160 then the time was represented in 0.1 ns units and so on. Table 7-69 lists an example for TIMEINCA.*incvalue* that provides high accuracy SYSTIME at convenient resolution. The table also shows the SYSTIME precision caused by the DMA clock as well as the Incvalue when synchronizing it with the master once per second.

Table 7-69 High Accuracy Incvalue Example

Link Speed	Clock Frequency	Incvalue Example	SYSTIME Resolution	Precision due to DMA Clock	Precision due to Incvalue	Overall Effective Precision
10 Gb/s	156.25 MHz (6.4 ns)	1280000000 (0x4C4B4000)	$5 * 10^{-9}$ nsec	± 3.2 ns	± 0.4 ns	± 3.22 ns
1 Gb/s	15.625 MHz (64 ns)	1280000000 (0x4C4B4000)	$50 * 10^{-9}$ nsec	± 32 ns	± 4 ns	± 32.2 ns

7.9.3.2 Time Stamping Mechanism

The time stamping logic is located as close as possible to the PHY. Figure 7-39 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. While the time stamp is sampled at a very late phase in the data path, the X540 does not insert it to the transferred packet. Instead, the X540 supports the two-step operation as follows for Tx and Rx.

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 7-38.

Rx Time Stamping

On the Rx, this logic parses the traversing frame. If it is matching the message type defined in RXMTRL register, the following packet's parameters are latched: The reception time stamp is stored in the RXSTMPL and RXSTMPH registers. The SourceuuID and SequenceID are stored in the RXSATRL and RXSATRH registers. In addition, two status bits are reported in the Rx descriptor: PTP packet indication (this bit is set only for L2 packets since on the UDP packets the port number direct the packet to the application) and the *TS* bit 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. In order 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 posed properly to the software.
Reception consecutive packets that are able to latch its reception time stamp are not supported by Niantic. The RXSATRL and RXSATRH registers may not contain sufficient information to identify uniquely a specific client. Therefore, master software must not initiate consecutive sync requests before the previous response is received.

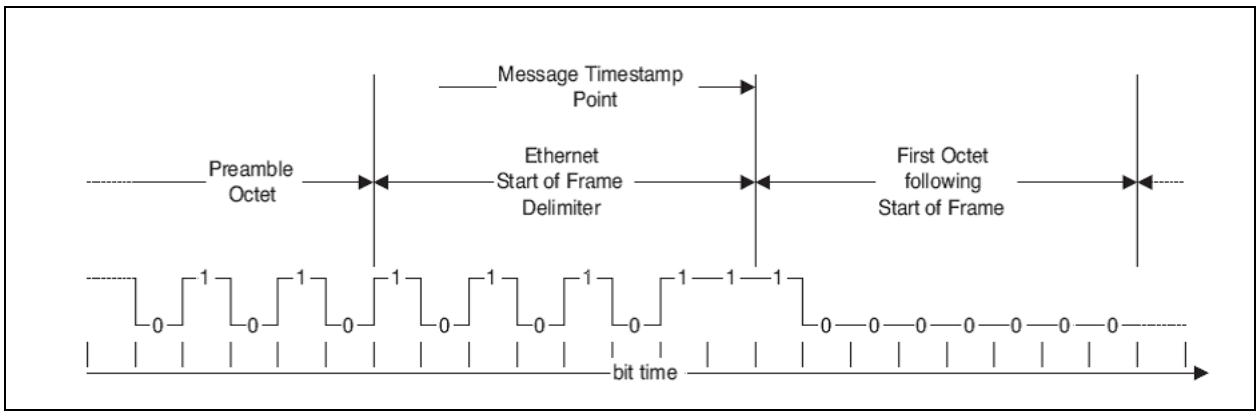


Figure 7-39 Time Stamp Point

7.9.3.3 Time Adjustment Mode of Operation

A node in a time sync network can be in one of two states: master or slave. When a time sync entity is in a master state, it should synchronize other entities to its system clock. In this case, no time adjustments are needed. When the entity is in slave state, it should

adjust its system clock by using the data arrived with the Follow_Up and Delay_Response packets and to the time stamp values of Sync and Delay_Req packets. When having all the values, the software on the slave entity can calculate its offset in the following manner.

After an offset calculation, the system time register should be updated. This is done by writing the calculated offset to TIMEADJL and TIMEADJH registers. The order should be as follows:

1. Write the lower portion of the offset to TIMEADJL.
2. Write the high portion of the offset to TIMEADJH to the lower 31 bits and the sign to the most significant bit.

After the write cycle to TIMEADJH the hardware adjusts the SYSTIME registers by the value of TIMEADJH and TIMEADJL.

Note: For normal operation, zero adjust is not expected and is not a supported operation.

7.9.4 Time Sync Related Auxiliary Elements

The time sync logic implements three types of auxiliary element using the precise system timer and SDPs: Event out signal at programmable Target Time; Sample Time Stamp on Input Event and Synchronized Clock Out. The following sections describe these auxiliary elements. These auxiliary can be provided on the SDP pins when they are set to native mode by the ESDP register. The table below shows the TimeSync signals mapping to the SDPs. Note that the Clock Out and Target Time 0 share the same SDPs while only one of them can be enabled as described in the following sections.

Reg / Pins	SDP0[1]	SDP1	SDP2	SDP3
ESDP Setting	SDP0_NATIVE = 1b SDP0_IODIR = 1b	SDP1_NATIVE = 1b SDP1_IODIR = 1b	SDP2_NATIVE = 1b SDP2_IODIR = 0b	SDP3_NATIVE = 1b SDP3_IODIR = 0b
SDP Functionality	Drive Target Time 0 / Clock Out	Drive Target Time 1	Sample Event on Time Stamp 0 register	Sample Event on Time Stamp 1 register

1. It is assumed that bit 15 of NC-SI Configuration 2 word in the NVM is cleared. Otherwise, SDP0 pins are used as input pins that encode the NC-SI package ID of the X540.

7.9.4.1 Event Out on Target Time

The target time is a mechanism used to generate an event out on the SDP pins (see SDP mapping in Section 7.9.4). The Target Time registers are 64 bit wide, structured the same as the SYSTIME registers. There are two sets of Target Time registers: TRGTTIME(0) and TRGTTIME(1). The software can arm an event out by setting the TRGTTIME(0) or TRGTTIME(1) registers to the desired time and then setting the EN_TT0 or EN_TT1 in the TSAUXC register respectively. When SYSTIME gets equal or cross the value of the enabled TRGTTIME, a Target Time event occurs with the following actions:

- The matched SDP changes its output level
- The matched EN_TT bit is cleared blocking further events. If additional event is required, the software should set the TRGTTIME and re-arm (set) the EN_TT bit.
- The TimeSync interrupt cause in the EICR register is set if enabled by the SDP0_INT or SDP1_INT bit in the TSAUXC register for Target Time 0 and Target Time 1 respectively.

When the software adjusts the SYSTIME as part of the synchronization protocol, the adjusted value may be equal or cross the value of the TRGTTIME(0). The hardware responds differently if such event happen following a forward or backward SYSTIME adjustment as described below:

- Forward SYSTIME adjustment - Cross/Equal the TRGTTIME(0) is treated as nominal TRGTTIME event described in this section
- Backward SYSTIME adjustment - Cross/Equal the TRGTTIME(0) is ignored avoiding false event if the SYSTIME rolled over during the update.

Note: The Target Time 0 and Clock out share the same logic and SDPs. Therefore, the software should not enable both of them in the TSAUXC register.

Note: In some applications it may be required to initialize the SDP to a known state before arming the Target Time. In such cases, there is a need to set the SDP momentarily to generic software controlled I/O mode enabling its level setting and then revert it back to the native mode. This procedure is programmed by the ESDP register.

7.9.4.2 TimeSync Clock Output

The TimeSync Clock Out is a configurable frequency clock, synchronized to SYSTIME. The Clock can be driven on SDP0 (see SDP mapping in Section 7.9.4) and enabled by the EN_CLK bit in the TSAUXC register. The duration of the clock out (its half clock cycle time) is configured by the CLKTIMEL and CLKTIMEH registers. These registers are 64 bit wide structured the same as the SYSTIME registers (as well as the TRGTTIME registers). Note that *CLKTIME* must never be smaller than: [(2 * IncValue) + (the biggest expected adjustment to SYSTIME)]. Otherwise, the clock output might stop for an entire wrap around of the SYSTIME.

The clock out logic shares the same resources as the Target Time 0 with the following additive functionality:

- As long as the EN_CLK is cleared the clock out signal is kept at its level (it is at low level following device initialization). Setting the EN_CLK clears the output signal and then enables the clock out.

- The initial value of the Target Time 0 is loaded by the hardware in one of two options:
  - If the TSAUXC.SYNCLK is cleared when setting the TSAUXC.EN_CLK the TRGTTIME(0) is set to SYSTIME + CLKTIME enabling immediate operation of the Clock Out. The first clock transition to high level will happen within half clock cycle time.
  - If the TSAUXC.SYNCLK is set when setting the TSAUXC.EN_CLK the TRGTTIME(0) is not modified by the hardware. The first transition of the Clock Out to high level will happen when SYSTIME gets equal or cross the value of the TRGTTIME(0). This options enables the software to define a specific time on which the clock will start.
- When SYSTIME gets equal or cross the value of the TRGTTIME(0), the following actions occurs:
  - The matched SDP changes its output level
  - The TRGTTIME(0) is auto-incremented by CLKTIME value enabling additional Clock Out transition at the next half clock cycle time.
  - The TimeSync interrupt cause in the EICR register is set if enabled by the SDP0_INT bit in the TSAUXC register.

When the software adjusts the SYSTIME as part of the synchronization protocol, the adjusted value may be equal or cross the value of the TRGTTIME(0). The Clock Out logic handles it the same as the Target Time logic described in Section 7.9.4.1. When using the Clock Out, the software should never adjust the SYSTIME by larger values than the Incvalue. It is a critical rule to avoid potential lost or extra clock transitions that may be caused by too large forward or backward adjustment respectively.

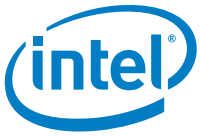
Note: The Target Time 0 and Clock out share the same logic. Therefore, the software should not enable both of them in the TSAUXC register

7.9.4.3 Time Stamp Events

Two Time Stamp inputs can be enabled by the TSAUXC.EN_TS bits sampled on SDP2 and SDP3 (see mapping to the SDP in Section 7.9.4). Following a level change of the enabled SDP pins, the system time is captured to the AUXSTMP registers and the matched AUTT bit in the TSAUXC register is set. Reading the AUXSTMP registers by software, clears the matched AUTT bit, enabling the next event. Note that the Time Stamp inputs on SDP2 and SDP3 can generate also a TimeSync interrupt if enabled by the SDP2_INT bit or the SDP3_INT bit in the TSAUXC register respectively.

7.9.5 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 X540 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]	***messageId***
1		Reserved	***versionPTP***
2	versionNetwork	messageLength	
3			
4	Subdomain	SubdomainNumber	
5		Reserved	
6		flags	
7			
8		Correction Field	
9			
10			
11			
12			
13			
14			
15			
16		reserved	
17			
18			
19			
20	messageType	***Source Port ID*** (only part of the field is captured in the RXSATRL and RXSATRH registers)	
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 RXMTRL register, 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

MessageId	Message Type	Value (hex)
PTP_MANAGEMENT_MESSAGE	General	D
Unused		E-F

If V2 mode is configured in TSAUXC register, then 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.

Note: This page intentionally left blank.

7.10 Virtualization

7.10.1 Overview

I/O virtualization is a mechanism that can be used to share I/O resources among several consumers. For example, in a virtual system, multiple operating systems are loaded and each operates as though the entire system's resources were at its disposal. However, for the limited number of I/O devices, this presents a problem because each operating system might be in a separate memory domain and all the data movement and device management has to be done by a Virtual Machine Monitor (VMM). VMM access adds latency and delay to I/O accesses and degrades I/O performance. Virtualized devices are designed to reduce the burden of the VMM by making certain functions of an I/O device shared among multiple guest operating systems or a Virtual Machine (VM), thereby allowing each VM direct access to the I/O device.

The X540 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 X540 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 X540 provides for these modes.

This section assumes a single-root implementation of IOV and no support for multi-root.

7.10.1.1 Direct Assignment Model

The direct assignment support in the X540 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.

7.10.1.1.1 Rationale

The direct assignment model enables each of the VMs to receive and transmit packets with minimum of overhead. Non time-critical operations such as initialization and error handling can be done via the PF driver. In addition, it is important that the VMs can operate independently with minimal disturbance. It is also preferable that the VM interface to hardware should be as close as possible to the native interface in non-virtualized systems in order 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. In order to support this, the DMA accesses of the queues associated to a VM should be identified as such on the PCIe using a different requester ID.
- 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.

7.10.1.2 System Overview

The following drawings show the various elements involved in the I/O process in a virtualized system. Figure 7-40 shows the flow in software VMDq2 mode and Figure 7-41 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 X540 or any other means provided by the VMM vendor.

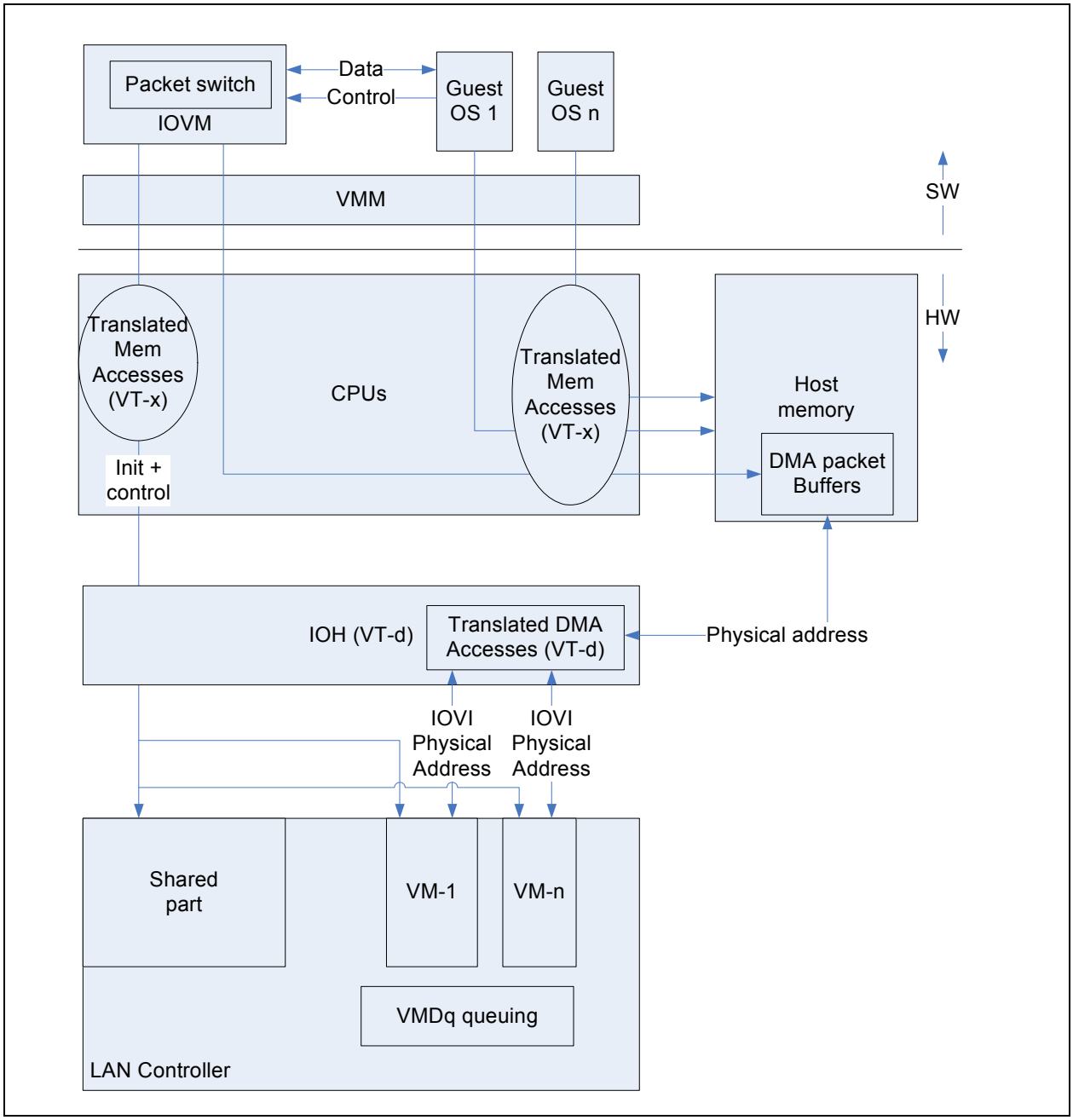


Figure 7-40 System Configuration for VMDq2 Mode

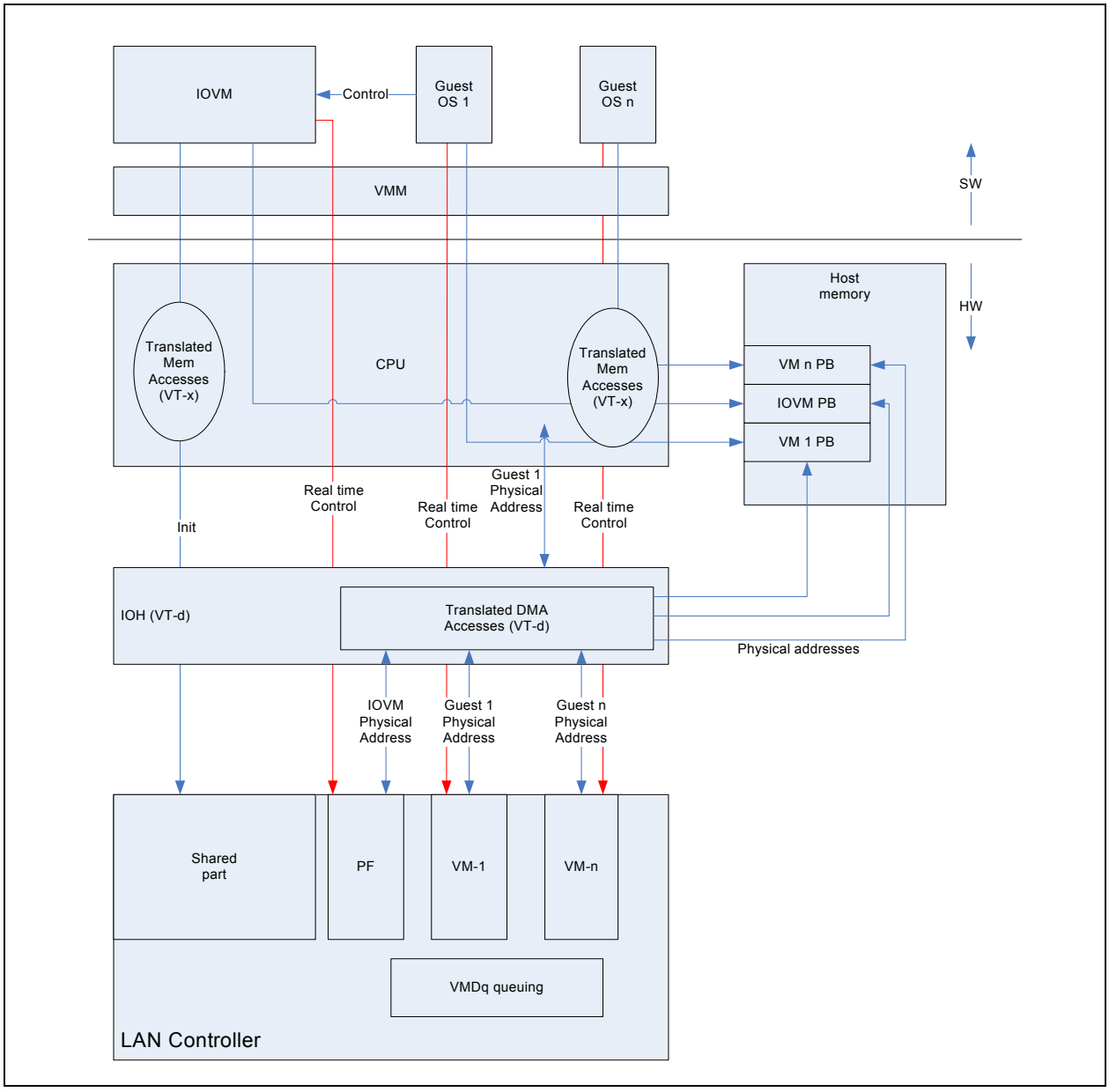


Figure 7-41 System Configuration for IOV Mode

7.10.2 PCI-SIG SR-IOV Support

7.10.2.1 SR-IOV Concepts

SR-IOV defines the following entities in relation to I/O virtualization:

- Virtual Image (VI): Part of the I/O resources are assigned to a 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 X540. 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.

7.10.2.2 Configuration Space Replication

The SR-IOV specification defines a reduced configuration space for the virtual functions. Most of the PCIe configuration of the VFs comes from the PF.

This section describes the expected handling of the different parts of the configuration space for virtual functions. It deals only with the parts relevant to the X540.

Details of the configuration space for virtual functions can be found in Section 9.4.5.

7.10.2.2.1 Legacy PCI Configuration Space

The legacy configuration space is allocated to the PF only and emulated for the VFs. A separate set of BARs and one bus master enable bit is allocated in the SR-IOV capability structure in the PF and is used to define the address space used by the entire set of VFs.

All the legacy error reporting bits are emulated for the VF. See Section 7.10.2.4 for details.

7.10.2.2.2 Memory BARs Assignment

The SR-IOV specification defines a fixed stride for all the VF BARs, so that each VF can be allocated part of the memory BARs at a fixed stride from the a basic set of BARs. In this method, only two decoders per replicated BAR per PF are required and the BARs reflected to the VF are emulated by the VMM.

The only BARs that are useful for the VFs are BAR0 and BAR3, so only those are replicated. The following table lists the existing BARs and the stride used for the VFs:

Table 7-70 BARs in X540 (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 9.4.4.8 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 7.10.2.7.

The following figure shows the different BARs in an IOV-enabled system:

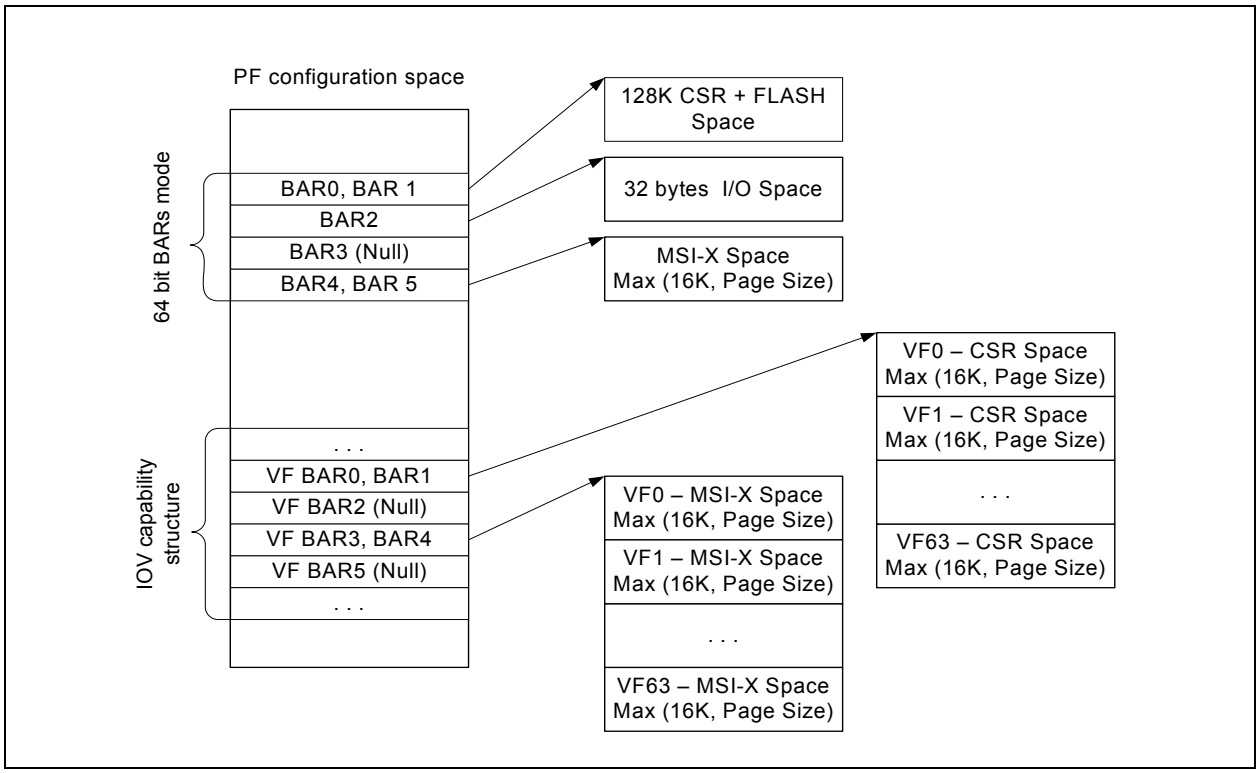


Figure 7-42 BARs in an IOV-enabled System

7.10.2.2.3 PCIe Capability Structure

The PCIe capability structure is shared between the PF and the VFs. The only relevant bits that are replicated are:

1. Transaction pending

2. Function Level Reset (FLR). See Section 7.10.2.3 for details.

7.10.2.2.4 MSI and MSI-X Capabilities

Both MSI and MSI-X are implemented in the X540. MSI-X vectors can be assigned per VF. MSI is not supported for the VFs.

See Section 9.3.8.1 for more details of the MSI-X and PBA tables implementation.

7.10.2.2.5 VPD Capability

VPD is implemented only once and is accessible only from the PF.

7.10.2.2.6 Power Management Capability

The X540 does not support power management per VF. The power management registers exist for each VF, but only the D0 power state is supported.

7.10.2.2.7 Serial ID

The serial ID capability is not supported in VFs.

7.10.2.2.8 Error Reporting Capabilities (Advanced and Legacy)

All the bits in this capability structure are implemented only for the PF. Note that the VMs see an emulated version of this capability structure. See Section 7.10.2.4 for details.

7.10.2.3 FLR Capability

The *FLR* bit is required per VF. Setting of this bit resets only a part of the logic dedicated to the specific VF and does not influence the shared part of the port. This reset should disable the queues, disable interrupts and the stop receive and transmit process per VF.

Setting the PF *FLR* bit resets the entire function.

7.10.2.4 Error Reporting

Error reporting includes legacy error reporting and Advanced Error Reporting (AER) or role-based capability.

The legacy error management includes the following functions:

1. 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:
   a. SERR# Enable
   b. Parity Error Response

c. Correctable Reporting Enable

d. Non-Fatal Reporting Enable

e. Fatal Reporting Enable

f. UR Reporting Enable

2. Error status in the configuration space. These should be set separately for each VF. This includes:

a. Master Data Parity Error

b. Signaled Target Abort

c. Received Target Abort

d. Master Abort

e. SERR# Asserted

f. Detected Parity Error

g. Correctable Error Detected

h. Non-Fatal Error Detected

i. Unsupported Request Detected

AER capability includes the following functions:

1. 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:

a. Uncorrectable Error Mask Register

b. Uncorrectable Error Severity Register

c. Correctable Error Mask Register

d. ECRC Generation Enable

e. ECRC Check Enable

2. Non-Function Specific Errors Status in the configuration space.

a. Non-Function Specific Errors are logged in the PF

b. Error logged in one register only

c. VI avoids touching all VFs to clear device level errors

d. 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

3. Function Specific Errors Status in the configuration space.
   a. Allows Per VF error detection and logging
   b. Help with fault isolation
   c. 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
4. Error logging. Each VF has it's own header log.
5. Error messages. In order 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.

7.10.2.5 Alternative Routing ID (ARI) and IOV Capability Structures

In order 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 9.4.3 and Section 9.4.4. Refer to the following section for details on the Requester ID (RID) allocation to VFs.

7.10.2.6 RID Allocation

RID allocation of the VF is done using the *Offset* field in the IOV structure. This field should be replicated per VF and is used to do the enumeration of the VFs.

Each PF includes an offset to the first associated VF. This pointer is a relative offset to the Bus/Device/Function (BDF) of the first VF. The *Offset* field is added to PF's requester ID to determine the requester ID of the next VF. An additional field in the IOV capability structure describes the distance between two consecutive VF's requester IDs.

7.10.2.6.1 BDF Layout

7.10.2.6.1.1 ARI Mode

ARI allows interpretation of the device ID part of the RID as part of the function ID inside a device. Thus, a single device can span up to 256 functions. In order 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. The following table lists the VF RIDs.

The layout of RID's used by the X540 is reported to the operating system via the PCIe IOV capability structure. See Section 9.4.4.6.

Table 7-71 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	0	B,16,0	B,10000,000	Offset to first VF from PF is 128.
1	0	B,16,1	B,10000,001	
0	1	B,16,2	B,10000,010	
1	1	B,16,3	B,10000,011	
0	2	B,16,4	B,10000,100	
1	2	B,16,5	B,10000,101	
...				
0	63	B,31,6	B,11111,110	
1	63	B,31,7	B,11111,111	Last

7.10.2.6.1.2 Non-ARI Mode

When ARI is disabled, non-zero devices in the first bus cannot be used, thus a second bus is needed to provide enough RIDs. In this mode, the RID layout is as follows:

Table 7-72 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	

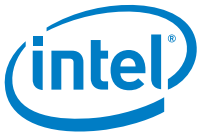


Table 7-72 RID Per VF — Non-ARI Mode

Port	VF#	B,D,F	Binary	Notes
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.

7.10.2.7 Hardware Resources Assignment

The main resources to allocate per VM are queues and interrupts. The assignment is static. If a VM requires more resources, it might be allocated to more than one VF. In this case, each VF gets a specific Ethernet MAC address/VLAN tag in order to enable forwarding of incoming traffic. The two VFs are then teamed in software.

7.10.2.7.1 PF Resources

A possible use of the PF is for a configuration setting without transmit and receive capabilities. In this case, it is not allocated to any queues but is allocated to one MSI-X vector.

The PF has access to all the resources of all VMs, but it is not expected to make use of resources allocated to active VFs.

7.10.2.7.2 Assignment of Queues to VF

See Section 7.2.1.2.1 for allocating Tx queues.

See Section 7.1.2.2 for allocating Rx queues.

The following table lists the Tx and Rx queues to VF allocation.

Table 7-73 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	...	...

Table 7-73 Queue to VF Allocation

VF	Queues in 16 VMs Mode	Queues in 32 VMs Mode	Queues in 64 VMs Mode
...		...	...
31		124-127	...
...			...
63			126-127

7.10.2.7.3 Assignment of MSI-X Vector to VF

See Section 7.3.4.3 for allocating MSI-X vectors in IOV mode.

7.10.2.8 CSR Organization

CSRs can be divided into three types:

- Global Configuration registers that should be accessible only to the PF. For example, link control and LED control. These types of registers also include all of the debug features such as the mapping of the packet buffers and is responsible for most of the CSR area requested by the X540. 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, DCA tag, etc. These parameters are used by both a VF in an IOV system and by the PF in a non-IOV mode.

In order 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:
  a. Fields related to the shared resources are reserved.
  b. 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 the VF Device Registers section.

7.10.2.9 SR IOV Control

In order 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 7.10.3.10.
- PFVFLREC register indicating that a VFLR reset occurred in one of the VFs (bitmap).

7.10.2.9.1 VF-to-PF Mailbox

The VF drivers and the PF driver require some means of communication between them. This channel can be used for the PF driver to send status updates to the VFs (such as link change, memory parity error, etc.) or for the VF to send requests to the PF (add to VLAN).

Such a channel can be implemented in software, but requires enablement by the VMM vendors. In order to avoid the need for such an enablement, the X540 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. 512 bytes is provided per port. The RAM is organized as follows:

Table 7-74 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 in order 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 7-75 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.		
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.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 7-76 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.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.ACK bit.		

Table 7-76 VF-to-PF Messaging Flow

Step	PF Driver	Hardware	VF #n Driver
10		Indicate an interrupt to VF #n.	
11			Clear VFMailbox.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.

7.10.2.10 DMA

7.10.2.10.1 RID

Each VF is allocated a RID. Each DMA request should use the RID of the VM that requested it. See Section 7.10.2.6 for details.

7.10.2.10.2 Sharing of the DMA Resources

The outstanding requests and completion credits are shared between all the VFs. The tags attached to read requests are assigned the same way as in a non-virtualized setting, although in VF systems tags can be re-used for different RIDs. See Section 3.1.3.1.

7.10.2.10.3 DCA

The DCA enable is common to all the devices (all PFs and VFs). Given a DCA enabled device, each VM might decide for each queue, on which type of traffic (data, headers, Tx descriptors, Rx descriptors) the DCA should be asserted and what is the CPU ID assigned to this queue.

Note: There are no plans to virtualize DCA in the IOH. Thus, the physical CPU ID should be used in the programming of the CPUID field.

7.10.2.11 Timers and Watchdog

7.10.2.11.1 TCP Timer

The TCP timer is available only to the PF. It might indicate an interrupt to the VFs via the mailbox mechanism.

7.10.2.11.2 IEEE 1588

IEEE 1588 is a per-link function and thus is controlled by the PF driver. The VMs have access to the real time clock register.

7.10.2.11.3 Watchdog

The watchdog was originally developed for pass-through NICs where virtualization is not a viable. Thus, this functionality is used only by the PF.

7.10.2.11.4 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.

7.10.2.12 Power Management and Wake Up

Power management is a PF resource and is not supported per VF.

7.10.2.13 Link Control

The link is a shared resource and as such is controllable only by the PF. This includes interface settings, speed and duplex settings, flow control settings, etc. The flow control packets are sent with the station Ethernet MAC address stored in the NVM. The watermarks of the flow control process and the time-out value are also controllable by the PF only. In a DCB environment, the parameters of the per TC flow control are also part of the PF responsibilities.

MACsec is a per-link function and is controlled by the PF driver.

Double VLAN is a network setting and as such should be common to all VFs.

7.10.2.13.1 Special Filtering Options

Pass bad packets is a debug feature. As such, pass bad packets is available only to the PF. Bad packets are passed according to the same filtering rules of the regular packets.

Note: Pass bad packets might cause guest operating systems to get unexpected packets. As a result, it should be used only for debug purposes of the entire system.

Receiving long packets is enabled separately per Rx queue in the RXDCTL registers. As this impacts the flow control thresholds, the PF should be made aware of the decision of all the VMs. Because of this, the setup of TSO packets is centralized by the PF and each VF might request this setting.

7.10.3 Packet Switching

7.10.3.1 Assumptions

The following assumptions are made:

- The required bandwidth for the VM-to-VM loopback traffic is low. That is, the PCIe bandwidth is not congested by the combination of the VM-to-VM and the regular incoming traffic. This case is handled but not optimized for. Unless specified otherwise, Tx and Rx packets should not be dropped or lost due to congestion caused by loopback traffic.
- If the buffer allocated for the VM-to-VM loopback traffic is full, it is acceptable to back pressure the transmit traffic of the same TC. This means that the outgoing traffic might be blocked if the loopback traffic is congested.
- The decision on local traffic is done only according to the Ethernet DA address and the VLAN tag. There is no filtering according to other parameters (IP, L4, etc.). The switch has no learning capabilities. In case of double VLAN mode, the inner VLAN is used for the switching functionality.
- The forwarding decisions are based on the receive filtering programming.
- No packet switching between TCs.
- Coexistence with IPSEC offload: Any loopback VM-to-VM traffic should not use the IPSEC offload (the *IPSEC* bit must be cleared in the advanced Tx data descriptor). IPsec processing of Tx packets destined to a local VM must be handled by software.
- 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.

7.10.3.2 Pool Selection

Pool selection is described in the following sections. A packet might be forwarded to a single pool or replicated to multiple pools. Multicast and broadcast packets are cases of replication, as is mirroring.

The following capabilities determine the destination pools of each packet:

- 128 Ethernet MAC address filters (RAH/RAL registers) for both unicast and multicast filtering. These are shared with L2 filtering. For example, the same Ethernet MAC addresses are used to determine if a packet is received by the switch and to determine the forwarding destination.

- 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

7.10.3.3 Rx Packets Switching

Rx packet switching is the second of three stages that determine the destination of a received packet. The three stages are defined in Section 7.1.2.

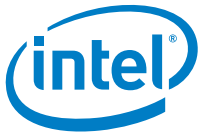
As far as switching is concerned, it doesn't matter whether the X540's virtual environment operates in IOV mode or in VMDq2 mode.

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.

7.10.3.3.1 Replication Mode Enabled

When replication mode is enabled, each broadcast/multicast packet can go to more than one pool. Finding the pool list of any packet is provided in the following steps:

1. **Exact unicast or multicast match** — If there is a match in one of the exact filters (RAL/RAH), for unicast or multicast packets, use the *MAC Pool Select Array* (MPSAR[n]) bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters.
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, add pools for which their PFVML2FLT.BAM bit (*Broadcast Accept Mode*) is set.
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 pools for which their PFVML2FLT.ROPE bit (*Accept Unicast Hash*) is set.
5. **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.
6. **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.
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. **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, untagged 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, then 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 queuing 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. 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.

12. **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.

    d. 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 in order to handle mirroring cases.

7.10.3.3.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode, the pool list of any packet always contains one pool only according to the following steps:

1. **Exact unicast or multicast match** — If the packet DA 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, 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 7.

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, then 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 and was not forwarded in step 1 and 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.

7.10.3.4 Tx Packets Switching

Tx switching is used only in a virtualized environment to serve VM-to-VM traffic. Packets that are destined to one or more local VMs are directed back (loopback) to the Rx packet buffers. Enabling Tx switching is done by setting the PFDTXGSWC.LBE bit. Tx to Rx switching always avoids packet drop as if flow control is enabled. Therefore, the software must set the FCRTH[n].RTH fields regardless if flow control is activated on the X540.

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.

Note: When Tx switching is enabled, the host shall avoid sending packets longer than 9.5KB

7.10.3.4.1 Replication Mode Enabled

When replication mode is enabled, the pool list for any packet is determined according to the following steps:

1. **Exact unicast or multicast match** — If there is a match in one of the exact filters (RAL/RAH), for unicast or multicast packets, take the MPSAR[n] bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters.
2. **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, add pools for which their PFVML2FLT.BAM bit (*Broadcast Accept Mode*) is set.
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 pools for which their PFVML2FLT.ROPE bit (*Accept Unicast Hash*) is set.
5. **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.
6. **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.
7. **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.
8. **Filter source pool** — The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.LLE bit is set.
9. **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.
10. **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.

11. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list (pre mirroring step).

12. **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 VMVLAN register.

    c. **Downlink port mirroring** — PFMRCTL.DPME is set and the packet is sent to the network.

13. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list (post mirroring step).

7.10.3.4.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode the pool list for any packet always contains one pool only according to the following steps:

1. **Exact unicast or multicast match** — If the packet DA matches one of the exact filters (RAL/RAH), take the MPSAR[n] bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters.

2. 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, 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.

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.

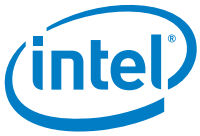
    b. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set.

    c. If there is no match, the pool list should be empty.

5. **Multicast or Broadcast** — If the packet is a multicast or broadcast packet and was not forwarded in step 1 and 2, set the default pool bit in the pool list (from PFVTCTL.DEF_PL).

6. **Filter source pool —** The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.LLE bit is set.

7. **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.

8. **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.

7.10.3.5 Mirroring Support

The X540 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:

1. **Pool mirroring** — reflect all the packets received to a pool from the network.
2. **Uplink port mirroring** — reflect all the traffic received from the network.
3. **Downlink port mirroring** — reflect all the traffic transmitted to the network.
4. **VLAN mirroring** — reflect 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: 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:

- **PFMRCT**L — 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.

7.10.3.6 Offloads

The general rule is that offloads are executed as configured for the pool and queue associated with the receive packet. Some special cases:

- If a packet is directed to a single pool, then offloads are determined by the pool and queue for that packet.

- If a packet is replicated to more than one pool, then 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.

7.10.3.6.1 Local Traffic Offload

The following capabilities are not supported on the loopback path:

- The Ethertype filters do not apply.
- Padding to a legal packet size is not supported.
- The following offload capabilities are only supported if XSUM offload is provided on the Tx path for the packet: RSS, 5-tuple filters, VLAN strip. The reason is that when XSUM is not offloaded, software does not provide the necessary offload offsets with the Tx packet.
- Header split/replication is not supported for NFS.
- Receive Side Coalescing (RSC) is not supported.
- FCoE offloads are not supported.

7.10.3.6.2 Rx Traffic Offload

- Security offloads (MACsec, IPsec) are managed globally and not per pool.
- CRC offload is a global policy. CRC strip is enabled or disabled for all received packets.

7.10.3.7 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 X540.

The X540 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.

7.10.3.8 Tx Queue Arbitration and Rate Control

In order to guarantee each pool with adequate bandwidth, a per-pool bandwidth control mechanism is added to the X540. 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 7.7.

7.10.3.9 Security Features

The X540 allows some security checks on the inbound and outbound traffic of the switch.

7.10.3.9.1 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.

7.10.3.9.2 Outbound Security

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.

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, then 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.

VLAN Tag Validation

In PCI-SIG IOV scenarios the driver might be malicious, and thus might fake a VLAN tag. The X540 provides the ability to override the VLAN tag inserted by 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.

Note: The VLAN insertion settings should be done before any of the queues of the VM are enabled.

Note: When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, VLAN validation should not be enabled.

7.10.3.10 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 — contains the following fields:

- Replication Enable (Rpl_En) — 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.
- Default Pool (DEF_PL) — defines the target pool for packets that passed L2 filtering but didn't pass any of the pool filters. This field is invalid when the Dis_Def_Pool bit is set.
- Disable Default Pool (Dis_Def_Pool) — disables acceptance of packets that failed all pool filters.
- PFVFRE — 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 (LBE) — VMDQ loopback enables switching of Tx traffic to the Rx path for VM-to-VM communication.
- PFVFSPOOF — MAC Anti-spoof Enable (MACAS) — enables filtering of Tx packet for anti-spoof.
- Local Loopback Enable (LLE) — defines whether or not to allow loopback of a packet from a certain pool into itself.
- Queue Drop Enable (PFQDE) 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 — Receive Overflow Multicast Packets (ROMPE) — accept multicast hash — Defines whether or not a pool accepts packets that match the multicast MTA table.
- Receive MAC Filters Overflow (ROPE) — accept unicast hash — Defines whether or not a pool accepts packets that match the unicast PFUTA table.
- Broadcast Accept (BAM) — Defines whether or not a pool accepts broadcast packets.
- Multicast Promiscuous (MPE) — Defines whether or not a pool accepts all multicast packets.
- Accept Untagged Packets Enable (AUPE) — Defines whether or not a pool accepts untagged VLAN packets.
- Mirror Control — See Section 7.10.3.5.
- PFVFTE — 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 — 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.
- Unicast Table Array (PFUTA) — 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.

- Multicast Table Array (MTA) — 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.

7.10.4 Virtualization of Hardware

This section describes additional features used in both IOV and VMDq2 modes.

7.10.4.1 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.

7.11 Receive Side Coalescing (RSC)

The X540 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The X540 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 X540 digests received packets and categorizes them by their TCP/IP connections (flows). For each flow, hardware coalesces the packets as shown in Figure 7-43 and Figure 7-44 (the colored parameters are explained in the RSC context table and receive descriptor sections). The X540 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 X540 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.

Figure 7-43 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.

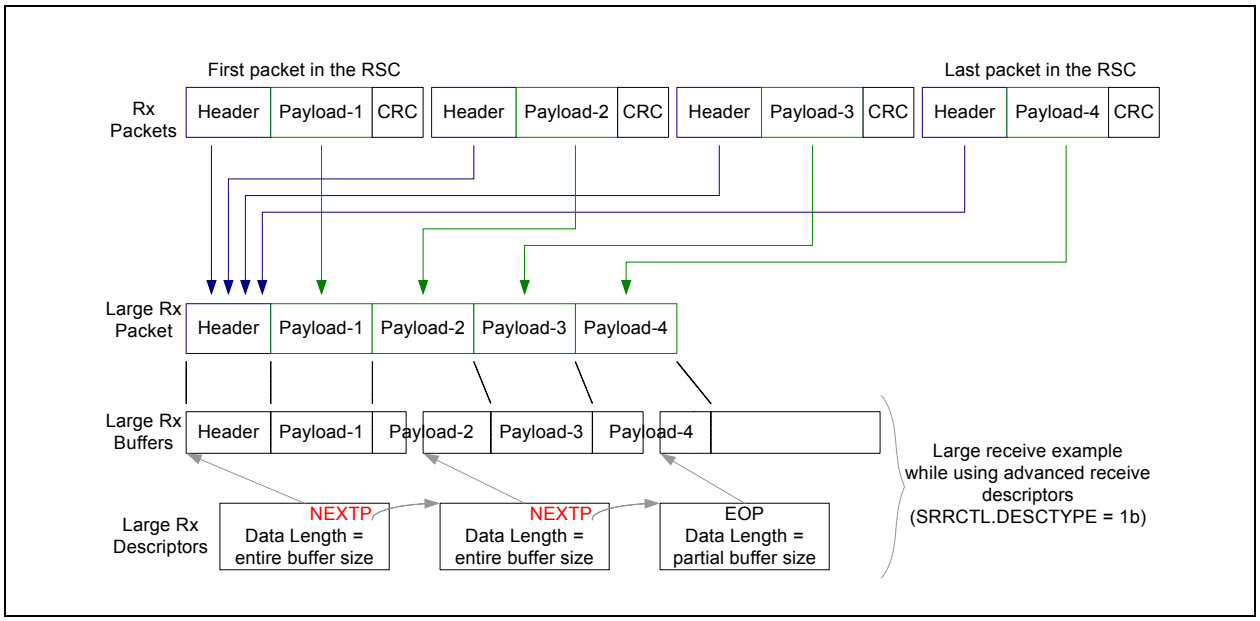


Figure 7-43 RSC Functionality (No Header Split)

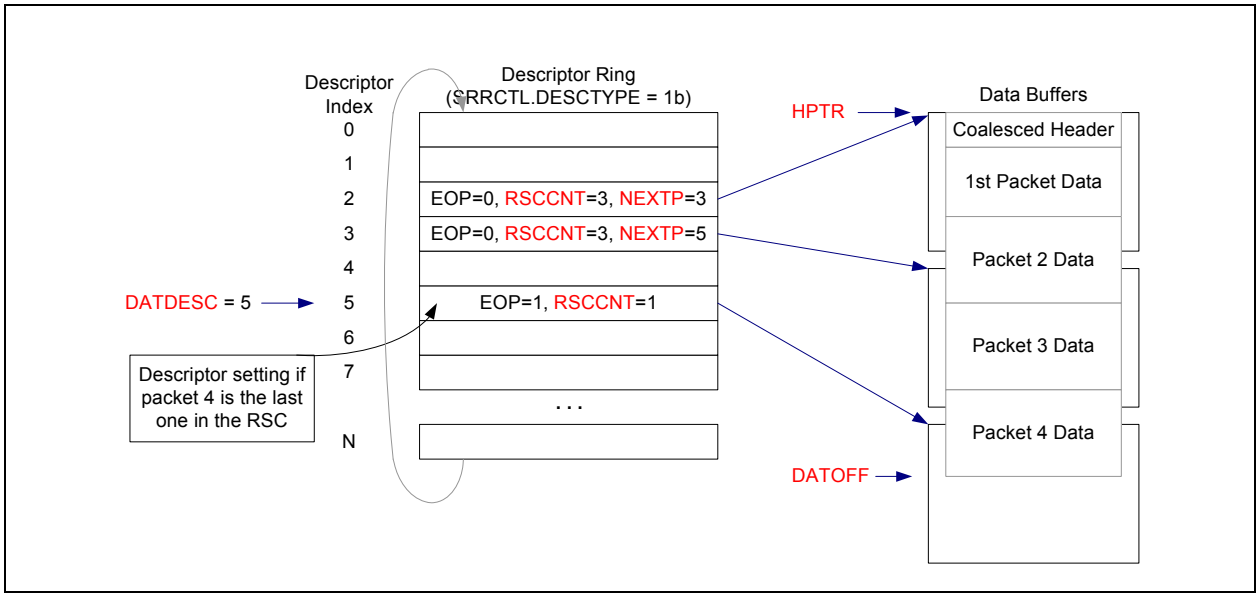


Figure 7-44 RSC Functionality (No Header Split)

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).

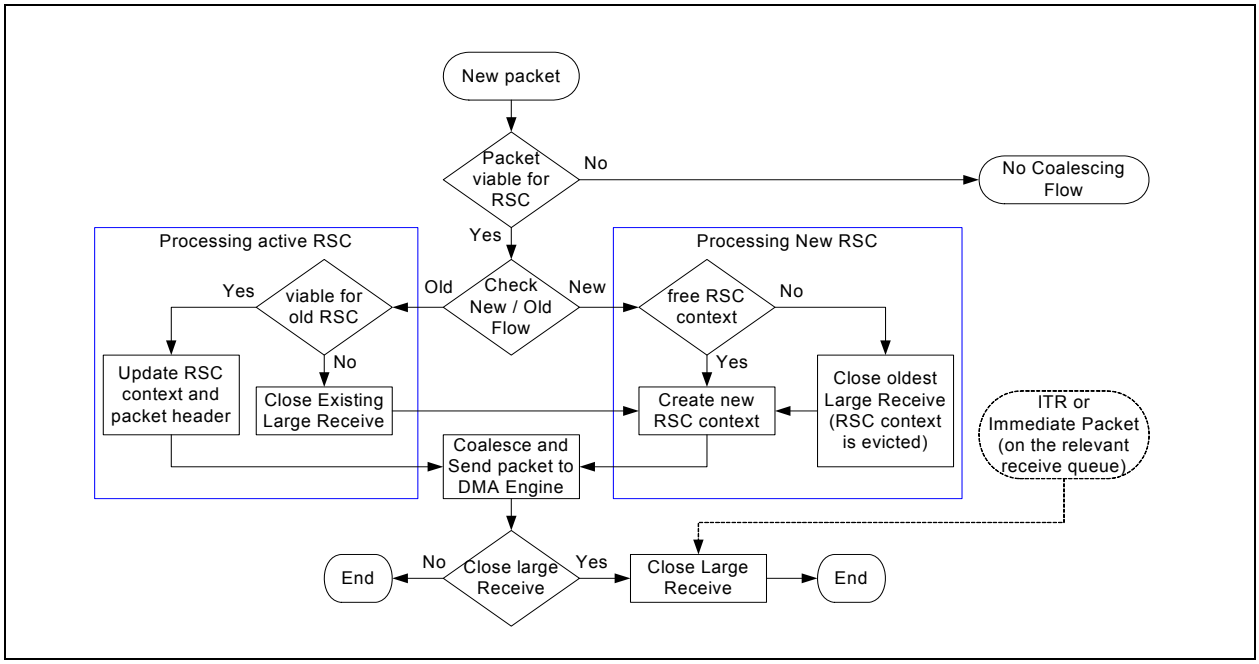


Figure 7-45 RSC Event Flow

7.11.1 Packet Viability for RSC Functionality

Incoming packets can be good candidates for RSC offload when the following conditions are met. If any of the these conditions are not met, 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 SRRCTL[n].BSIZEPACKET (packet buffer size) must be 2 KB at minimum.
- The received packet has no MAC errors and no TCP/IP checksum errors. MAC errors are: CRC error or undersize frame received or oversize frame received or error control byte received in mid-packet or 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) then the received packet is not a candidate for RSC.
- If the packet carries MACsec encapsulation, the MACsec offload is activated on the packet with no errors.
- The packet type is TCP/IPv4 (non-SNAP) with optional VLAN header.

- IP header does not carry any option headers.
- NFS packets can be coalesced only if NFS filtering is disabled by setting both RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits to 1b. Furthermore, the PSR_type1 bit (header split on NFS) must be turned off in all PSRTYPE[n] registers.
- If NFS coalescing is not required, software should set both RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits to 0b.
- The packet does not carry IPsec encapsulation (regardless if IPsec offload is enabled).
- 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 7-77).
- The *ECT* and *CE* bits in the *TOS* field in the IP header are not equal to 11b (see the flags in Table 7-78).
- 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, then a few additional conditions should be met as listed in Section 7.11.4.

The supported packet format is as follows:

Size	Packet Fields
6 bytes	Destination Ethernet MAC address.
6 bytes	Source Ethernet MAC address.
[8 / 16 bytes]	Optional MACsec header (supported by RSC only if MACsec offload is enabled and the hardware extracts this header).
[4 bytes]	Optional VLAN.
[4 bytes]	Optional 2nd VLAN (double 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).
[8 / 18 bytes]	Optional MACsec Integrity Checksum Value — ICV (supported by RSC only if MACsec offload is enabled and the hardware extracts this field).

Table 7-77 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 7-78 IP TOS Field — Bit Map

7	6	5	4	3	2	1	0
TOS (DS)						ECT	CE

Table 7-79 TCP Time-Stamp Option Header (RFC 1323)

1 byte: First on the Wire	1 Byte	4 Byte	4 Bytes: Last on the Wire
Kind = 0x8	Length = 10	TS Value (TSval)	TS Echo Reply (TSecr)

7.11.2 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 7-80). 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 7.11.3.
- If the packet's flow matches an active RSC context then the packet might be appended to the existing RSC as described in Chapter 7.11.4.

Table 7-80 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.
8 bytes		Total.

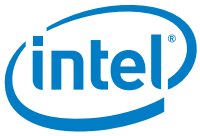


Table 7-80 RSC Context (Continued)

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 7.11.5.2).
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.

7.11.3 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.

7.11.3.1 RSC Context Setting

The X540 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.

7.11.4 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 bit in the TCP header is 1b or equals to the RSCACK bit in the RSC context (an active RSCACK context and inactive received ACK bit is defined as no match).

- The packet type remains the same as indicated by the RSCACKTYPE bit in the RSC context. Packet type can be either 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 ACK sequence number in the TCP header is greater than the RSCSEQ number in the RSC context. Note that the X540 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 X540 closes the old large receive it is guaranteed that there is at least one free context.

If the received packet meets all the above conditions, the X540 appends this packet to the active large receive and updates the context as follows. The packet is then DMA'ed to the RSC buffers (as described in Section 7.11.5).

- Update the TCP ACK: The RSCACK in the large receive context gets the value of the ACK bit in the TCP header in the received packet.
- 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 incremented 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 incremented 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.
- Note that MACsec encapsulation (if it exists) is stripped first by hardware. In this case, hardware also strips the Ethernet padding (if it exists).
- 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)} while...

  — 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 incremented 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.

7.11.5 Packet DMA and Descriptor Write Back

The Figure 7-46 shows a top view of the RSC buffers using advanced receive descriptors and header split descriptors.

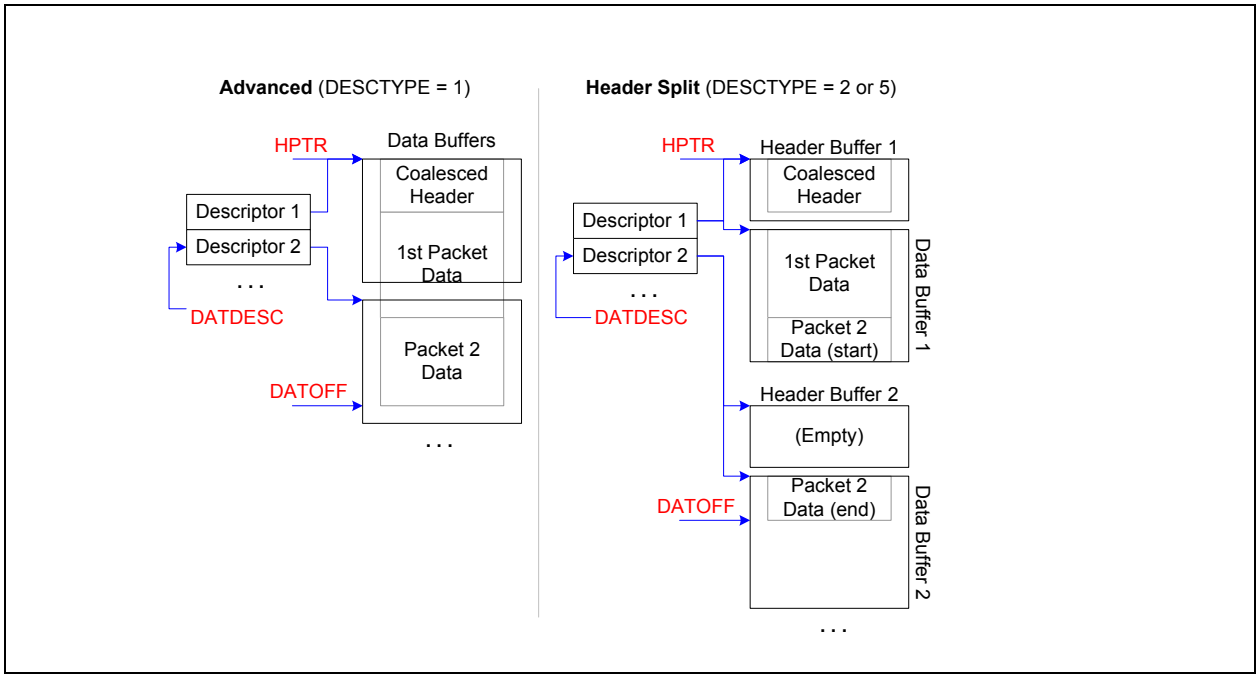


Figure 7-46 RSC — Header and Data Buffers

7.11.5.1 RSC Descriptor Indication (Write Back)

After receiving each packet, the X540 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 7.11.5.1 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.	
RSCCNT: indicates the number of coalesced packets in this descriptor.	

7.11.5.2 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 7.11.5.3 and Section 7.11.5.4.

7.11.5.3 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 7-46 for more details.

The packet’s header is posted to host memory after it is updated by the RSC context as follow:

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.

7.11.5.4 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 7.11.5.1 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 then 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, then the X540 completes the current large receive while the new packet starts a new large receive.

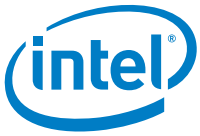
7.11.6 RSC Completion and Aging

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 X540 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 7.11.4). 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.
- LLI packet reception — All active RSC's on the same Rx queue complete and then the interrupt is asserted. Hardware then triggers RSC completion on all other queues associated with this interrupt.
- 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 an LLI event is generated that affects RSC completion as well.



- Interrupt assertion by setting the EICS register has the same impact on packet reception as described in Section 7.3.1.2.1.Auto RSC Disable — When the interrupt logic triggers RSC completion it might also auto-disable further coalescing by clearing the RSCINT.RSCEN bit. Auto RSC disablement is controlled by the RSCINT.AUTORSC bit. In this mode, hardware also re-enables RSC (by setting the RSCINT.RSCEN bit back to 1b) when the interrupt is re-enabled by the EIMS (either by software or hardware as described in Section 7.3.1.3 and Section 7.3.1.5).

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.

SW 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. The hardware disables further coalescing by clearing the RSCINT.RSCEN bit when the RSCINT.AUTORSC bit is set. When the software re-enables the interrupt (at the end of the interrupt handler), the hardware sets the RSCINT.RSCEN bit back to '1' if the RSCINT.AUTORSC bit is set. Note that the receive interrupt may be a result of either EITR expiration or a reception of a packet that generates a Low Latency Interrupt (LLI).

Note: This page intentionally left blank.

7.12 IPsec Support

7.12.1 Overview

This section defines the hardware requirements for the IPsec offload ability included in the X540. 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, in order 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 X540 processes it like it does for any other unsupported Layer4 protocol, and without performing on it any layer4 offload.

Refer to section Section 4.6.12, "Security Initialization" on page 159, 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)
- MACsec offload:
  - A device interface is operated in either MACsec offload or IPsec offload mode, but not both of them altogether
  - If both IPsec and MACsec encapsulations are required on the same packets, the X540's interface is operated in MACsec offload mode, while IPsec is performed by the operating system

- 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 X540 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), then 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, then 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 the X540, the driver should make sure it does not accept offload requests for flows to the BMC. In order 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 16.5.

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 X540 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 by hardware 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 X540 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 Section 7.12.4.2.

Shaded fields in the tables that follow correspond to fields that need to be updated per each segment.

Table 7-81 IPv4 TSO ESP Packet Provided by Software

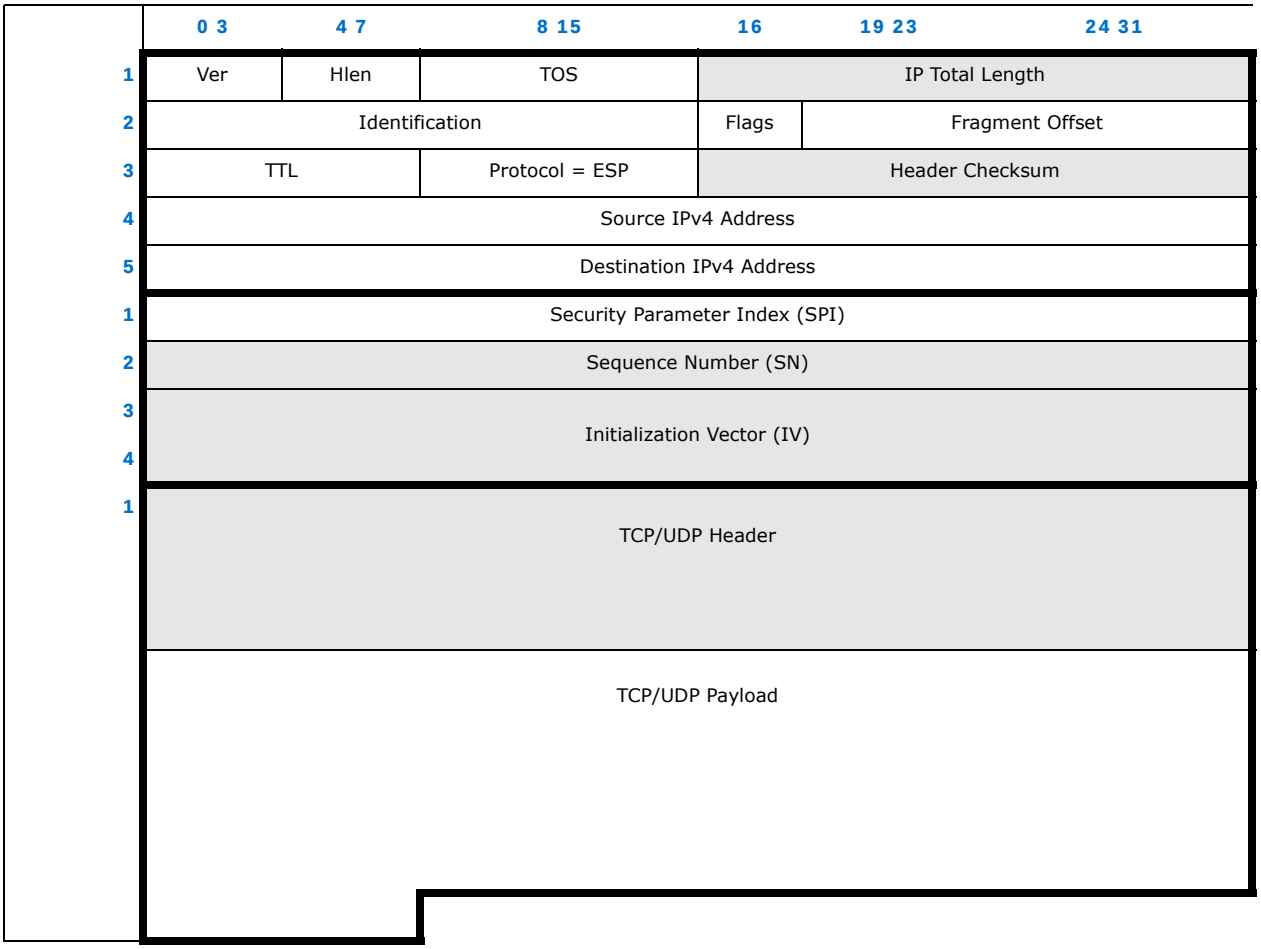


Table 7-82 IPv6 TSO ESP Packet Provided by Software

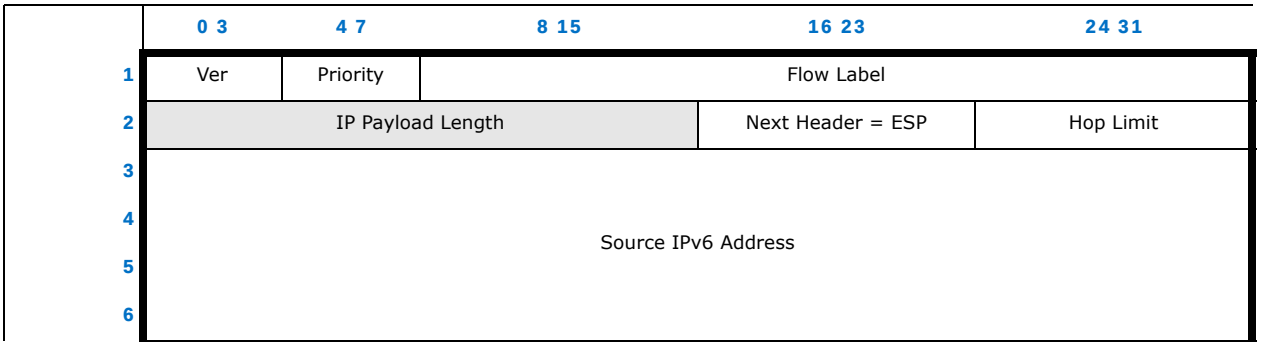
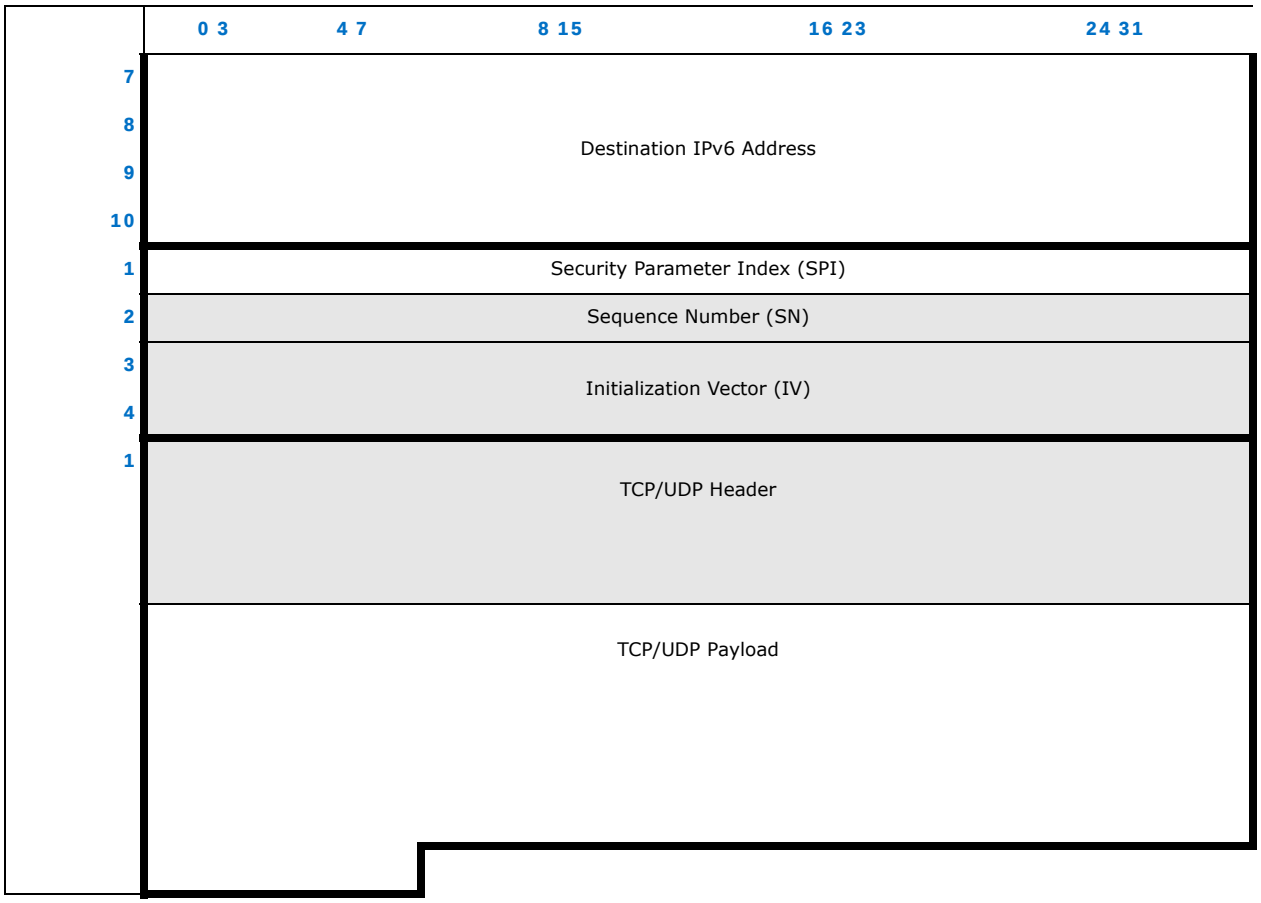


Table 7-82 IPv6 TSO ESP Packet Provided by Software (Continued)



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 7-83 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 the X540. Packets that have already started to be transmitted on the wire are sent with a wrong CRC.

Upon reset, the X540 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), then 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 16.5, 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 16.5.

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), then 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 16.5, 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 16.5.

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), then 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 (excepted to 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 16.5, 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 16.5.
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 the same AES-128-GCM crypto engine used for MACsec. It 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 7-47. Refer to the GCM specification for the internal details of the engine. The difference between IPsec and MACsec, and between the different IPsec modes reside in the set of inputs presented to the box.

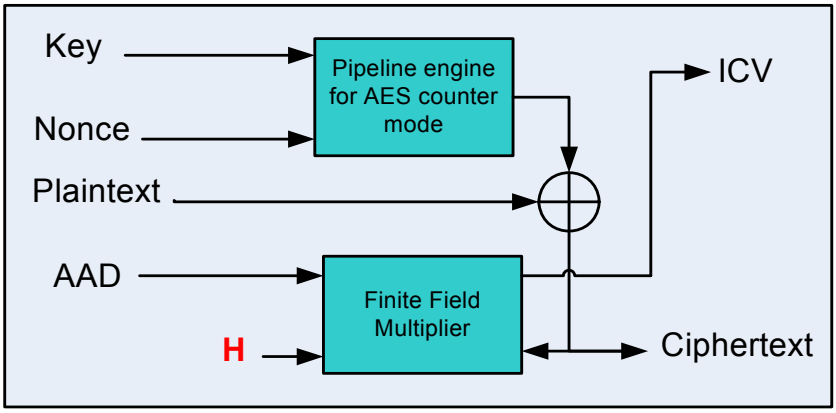


Figure 7-47 AES-128 Crypto Engine Box

- **Key** — 128-bits AES-128 KEY field (secret key) stored for that IPsec flow in the Tx SA table:

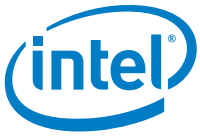
  ```
  Key = AES-128 KEY
  ```

- **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 16.5.2.

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 16.5.2.

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 16.5.1. 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 7.1.6 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 X540:

- 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 7-84 IP Address Table

IP Address
16 bytes

Table 7-85 SPI Table

SPI	IP Index (points to IP address table)
4 bytes	1 bytes

Table 7-86 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 Addr. 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, then 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 the X540.

Upon reset, the X540 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), then 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, then 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, then 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 whether or not 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 16.5, 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 16.5, 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 16.5. 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 16.5.2. 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

The following table lists how IPsec packets are handled according to some of their characteristics.

Table 7-87 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.

7.13 Fibre Channel over Ethernet (FCoE)

7.13.1 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 7-48 shows a connection between an FCoE initiator and legacy FC targets.

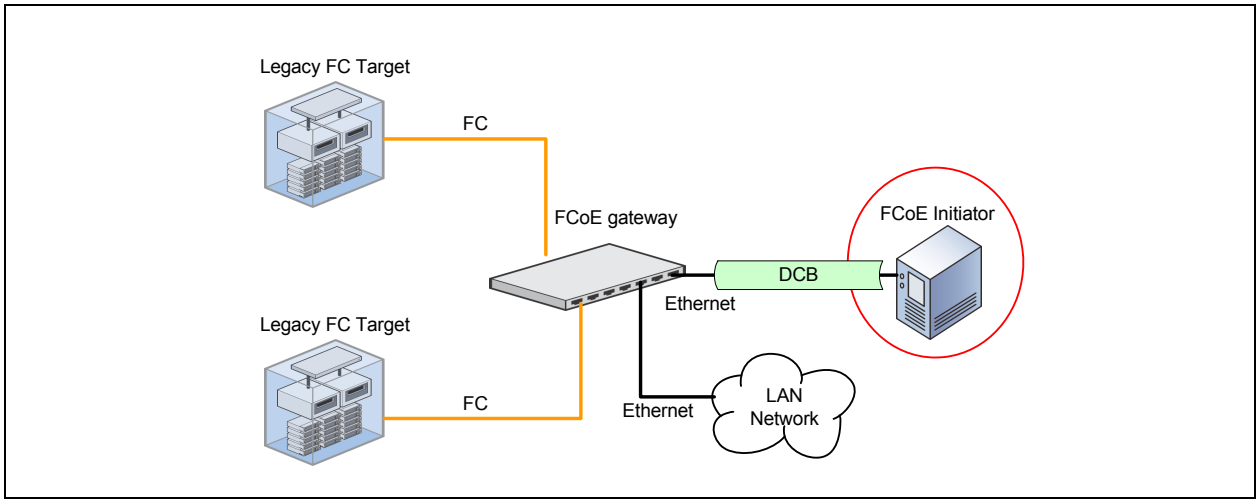


Figure 7-48 Connecting an FCoE Initiator to FC Targets

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. In order to compete with this market, the X540 offloads the main data path of I/O Read and Write commands to the storage target.

7.13.1.1 FC Terminology

Useful background on FC framing and its Ethernet encapsulation can be found in Section 16.6. 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, but they are supported by the X540.

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.

7.13.2 FCoE Transmit Operation

Transmit FCoE offload is enabled by setting the TUCMD.FCoE bit in the transmit context descriptor. The X540 supports the following offload capabilities: FC CRC calculation and insertion, FC padding insertion and FC segmentation. These capabilities are described in the following sections.

7.13.2.1 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.
MACsec offload	MACsec encapsulation covers the entire Ethernet packet payload (it includes both FCoE content and Ethernet padding). When packets carry MACsec encapsulation on the wire, MACsec offload by hardware should be activated.
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.

Cross Function	Requirements
SNAP packet	The X540 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 X540 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.

7.13.2.2 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, then 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)}. While PAYLEN is defined in the Tx data descriptor and MACLEN is defined in the Tx context descriptor.

The X540 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.

7.13.2.3 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 7-88. The *ORIS* bit defines if the TSO starts an FC sequence or if the first frame on the FC sequence is already sent.

Table 7-88 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)

7.13.2.4 EOF Insertion

The X540 automatically inserts the *End of Frame* field when the TUCMD.FCoE bit in the transmit context descriptor is set.

The EOF codes that are inserted into the transmitted packets are stored in the TEOFF register. The TEOFF register contains four EOF codes named EOF0...EOF3 that are supported by the transmit FCoE offload. By default, these values are programmed into the following values: EOF0 = EOFn; EOF1 = EOFt; EOF2 = EOFni; EOF3 = EOFa. The EOF flag and *Orientation End (ORIE*) bit in the *FCoEF* field in the transmit context descriptor define an index value. This index is used to extract the EOF code that is inserted to the packet as listed in Table 7-89. The *ORIE* bit determines if the packet or TSO ends an FC sequence or if there are more packet(s) to follow on the same FC sequence.

Table 7-89 EOF Codes in Single Send and TSO

EOF Bits in the Context Descriptor	ORIE Bit in the Context Descriptor	EOF Code in a — - — Single Send	EOF Codes in a TSO		
			Last Frame of the TSO	Other Frames of the 'TSO	TSO = Single Packet
00 (EOFn)	0 (not a sequence end)	EOF0 (EOFn)	EOF0 (EOFn)	EOF0 (EOFn)	EOF0 (EOFn)
00 (EOFn)	1 (sequence end)	EOF0 (EOFn)	EOF1 (EOFt)	EOF0 (EOFn)	EOF1 (EOFt)
01 (EOFt)	1 (don't care)	EOF1 (EOFt)	n/a	n/a	EOF1 (EOFt)
10 (EOFni)	1 (don't care)	EOF2 (EOFni)	n/a	n/a	EOF2 (EOFni)
11 (EOFa)	1 (don't care)	EOF3 (EOFa)	n/a	n/a	EOF3 (EOFa)

7.13.2.5 FC CRC Insertion

FC CRC calculation is one of the most CPU intensive tasks in large transactions. The X540 offloads the FC CRC calculation when the *FCoE* bit is set in the *TUCMD* field within the transmit context descriptor. The X540 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 7-49 specifies the FCoE frame and the relevant parameters to CRC calculation.

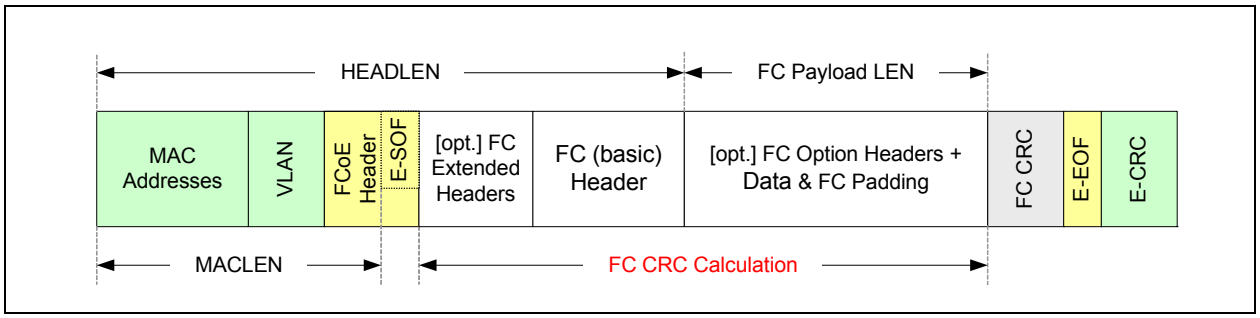


Figure 7-49 FCoE Frame and Relevant Transmit Descriptor Parameters

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 7-49 (eight bytes before the Ethernet CRC).

7.13.2.6 Host Data Buffers Content for a Single Packet Send

The Table 7-90 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 7-90 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 Figure 7-90.

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 X540 calculates and inserts the FC CRC bytes.

FC-Padding — The X540 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.

MACsec Header and Digest — When the link is secured by MACsec, then MACsec offload must be enabled and the MACsec encapsulation is added by hardware.

7.13.2.7 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 X540 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 7-49.
- The *SOF* and *EOF* fields are defined by the SOF, ORIS, EOF and ORIE fields in the context descriptor as described in Section 7.13.2.3 and Section 7.13.2.4.
- 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.

7.13.2.7.1 Host Data Buffers Content for TSO Offload

The Figure 7-50 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.

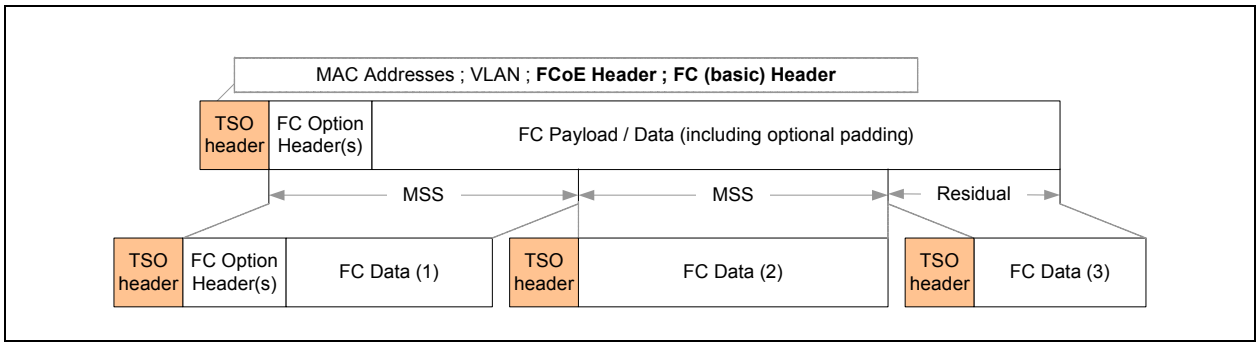


Figure 7-50 FCoE TSO Provided by the FCoE Driver

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 7-50) 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 16.6.2.3.

FCoE TSO — Payload

- FC option headers as described in Section 16.6.2.5.
- 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 7.13.2.3)

F_CTL Table 7-91 lists those fields in the F_CTL that are modified between consecutive frames of a TSO (see Section 16.6.2.3 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 7-91 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
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 7-92 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 7-92 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 incremented 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 incremented 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 Section 7.13.2.5.

FC_Padding Calculated the number of required padding bytes and inserted them on the last frame as described in section Section 7.13.2.2.

EOF The EOF flags are defined by the *EOF* field and the *ORIE* bit in the context descriptor (more details in Section 7.13.2.4).

7.13.3 FCoE Receive Operation

the X540 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.

7.13.3.1 FCoE Receive Cross Functionality

FCoE receive offload capabilities coexist with other functions in the X540 are listed as follows:

Table 7-93 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.
MACsec offload	MACsec encapsulation covers the entire Ethernet packet payload. If the traffic includes MACsec, hardware must process first the MACsec encapsulation uncovering the FCoE plain text to the FCoE offload logic. If the MACsec processing is not enabled and the packets include MACsec encapsulation, then the packets are posted to the matched legacy receive queue. If the MACsec processing is enabled but fails for any reason, the packet can still be posted to the matched legacy queue according to save bad frames policy.
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 X540 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 X540 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.
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, plus optional MACsec encapsulation packet might exceed the 2200 bytes. In order 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.

7.13.3.2 FC Receive CRC Offload

FC CRC calculation is one of the most CPU intensive tasks in TSO transactions. The X540 offloads the receive FC CRC integrity check while trashing the CRC bytes and FC padding bytes.

The X540 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 X540 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
- If the received frame includes MACsec encapsulation then MACsec offload must be enabled and MACsec integrity is found OK.

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.

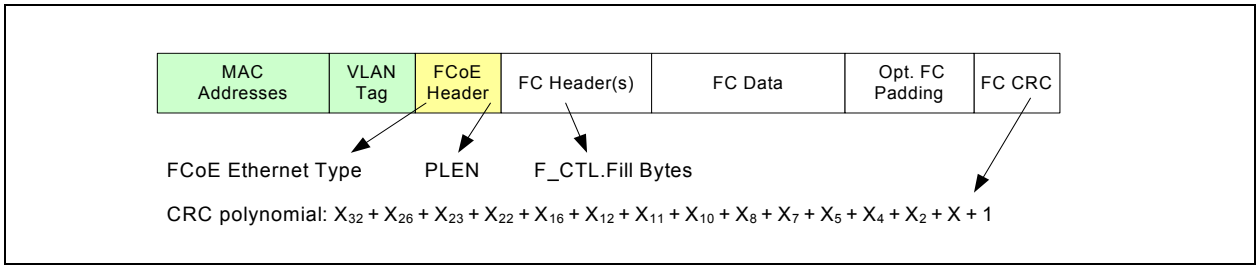


Figure 7-51 Relevant FCoE and FC Fields for CRC Receive Offload

7.13.3.3 Large FC Receive

Large FC receive includes two types of offloads. The X540 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 X540 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 7-52 shows the mapping of received FCoE frames to the legacy Rx queue and the user buffers. Figure 7-53 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 7.13.3.3.1
- FC read exchange flow — Section 7.13.3.3.2
- FC write exchange flow — Section 7.13.3.3.3
- FCoE receive filtering (Frame types and rules) — Section 7.13.3.3.5, Section 7.13.3.3.10 and Section 7.13.3.3.12
- User descriptors — Section 7.13.3.3.7
- Header posting to the legacy receive queues and FC exceptions — Section 7.13.3.3.13 and Section 7.13.3.3.14

- Interrupts — Section 7.13.3.3.15

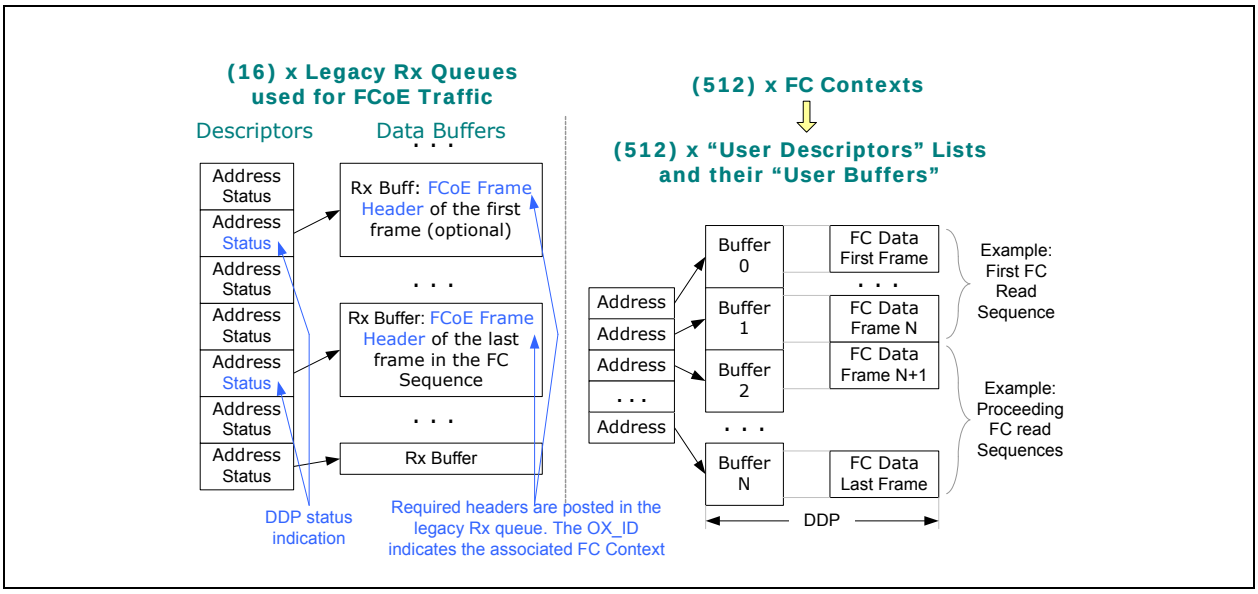


Figure 7-52 Large FC Reception to User Buffers and Legacy Rx Queue

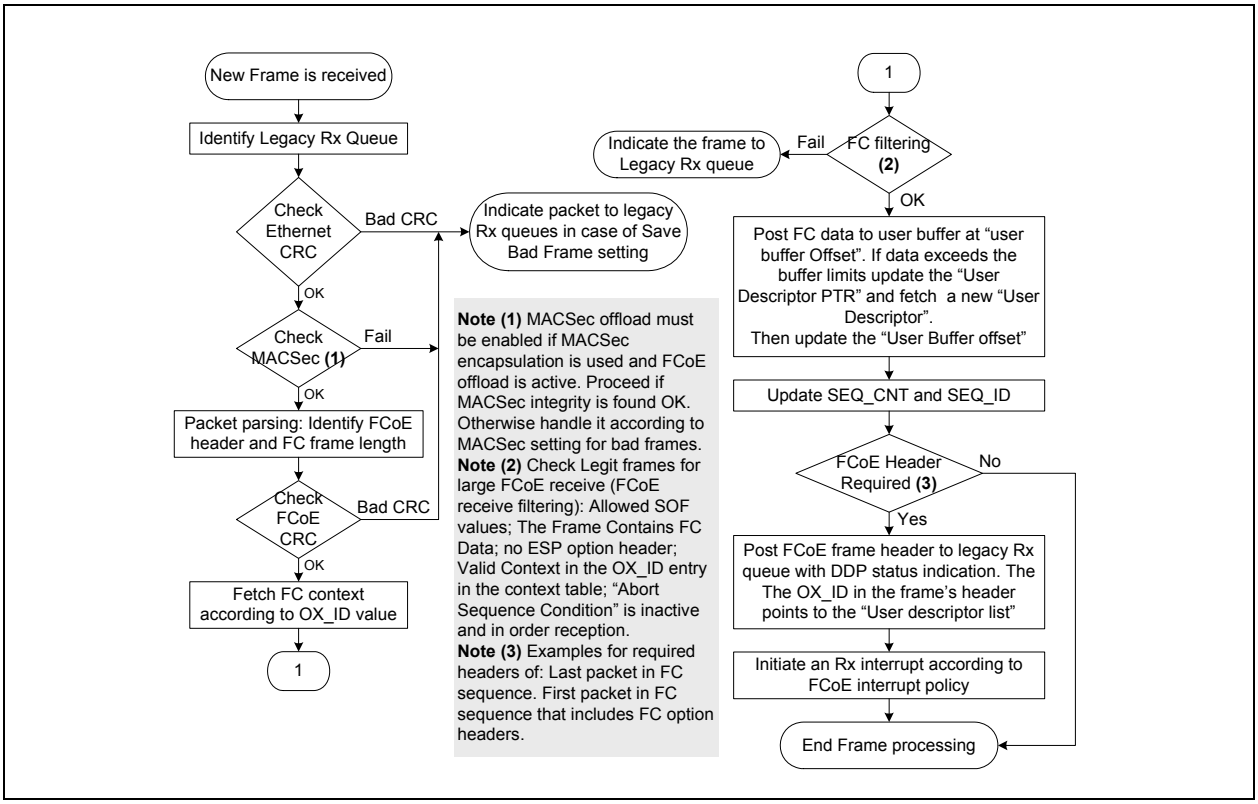


Figure 7-53 FCoE Large Receive Flow Diagram

7.13.3.3.1 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 7.13.3.3.5 for details on context invalidation). For more details on FCoE initialization flow see Section 4.6.9.

7.13.3.3.2 FC Read Exchange Flow

Figure 7-54 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 7.13.3.3.5. Section 7.13.3.3.12 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 7-54 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 7-54. 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 7.13.3.3.13). Software indicates the read completion to the application.

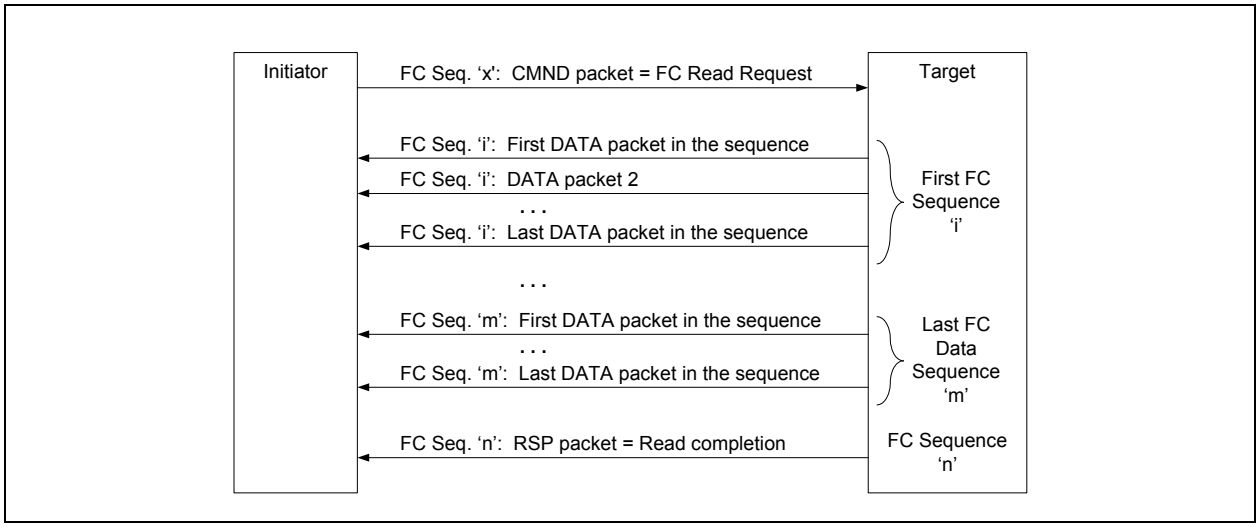


Figure 7-54 Example for FC Class 3 Read Exchange Flow

7.13.3.3.3 FC Write Exchange Flow

Figure 7-55 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 7.13.3.3.5.
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 7.13.3.3.13). 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.

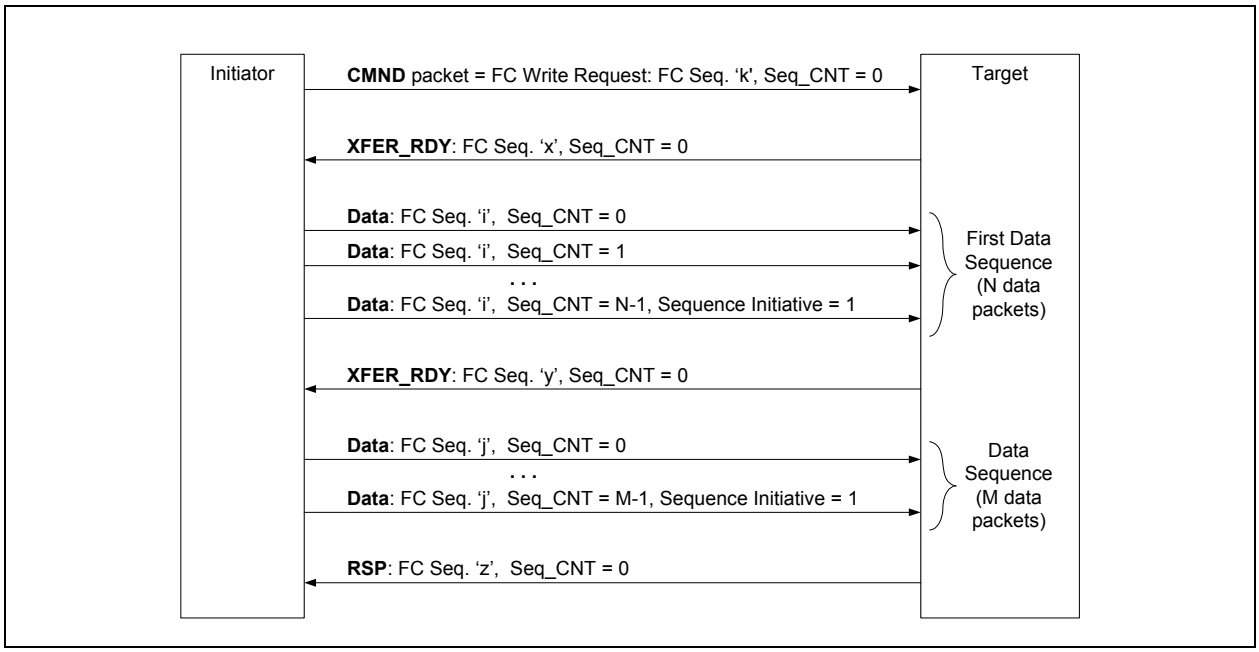


Figure 7-55 Example for FC Class 3 Write Exchange Flow

7.13.3.3.4 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.

7.13.3.3.5 FCoE Receive Filtering

Received FCoE frames are associated to one of the legacy receive queues according to the scheme described in Section 7.1.2. 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 7.13.3.3.13).

FCoE frames that carry FC class 3 or class 2 data can be posted to large receive buffers if they meet the following conditions:

- If the received packet carries MACsec encapsulation it must be offloaded (and de-capsulated) by hardware.
- 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).

- 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 7.13.3.3.5.
- 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 any 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 7.13.3.3.7 and does not carry any exception errors as defined in Section 7.13.3.3.14.
- 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.

7.13.3.3.6 DDP Context

Hardware can provide DDP offload for up to 512 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 512 contexts for the 512 concurrent outstanding exchanges. Using exchange ID values between 0 to 511, 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 9 LS bits of the exchange ID is the DDP context Index. In addition, read DDP context is defined by the 9 LS bits of the OX_ID and write DDP context is defined by the 9 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 and FCFLTRW register) as listed in Table 7-94 and shown in Figure 7-56.

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 7.13.3.3.5. Initialization values and the updated ones are listed in this section.

Table 7-94 Large FC Context Table

Exchange ID	DMA Context (FCPTRL, FCPTRH, FCBUFF, FCDMARW)		Filter Context (FCFLT and FCFLTRW)	
	DMA Flags	User Descriptor	Filter Flags	In Order Reception
0	Valid, First, Count	Size, Offset, Pointer	Valid, First	Seq_ID, Seq_CNT,PARAM
1	Valid, First, Count	Size, Offset, Pointer	Valid, First	Seq_ID, Seq_CNT,PARAM
2	Valid, First, Count	Size, Offset, Pointer	Valid, First	Seq_ID, Seq_CNT,PARAM
. . .	. . .		. . .	
511	Valid, First, Count	Size, Offset, Pointer	Valid, First	Seq_ID,Seq_CNT,PARAM

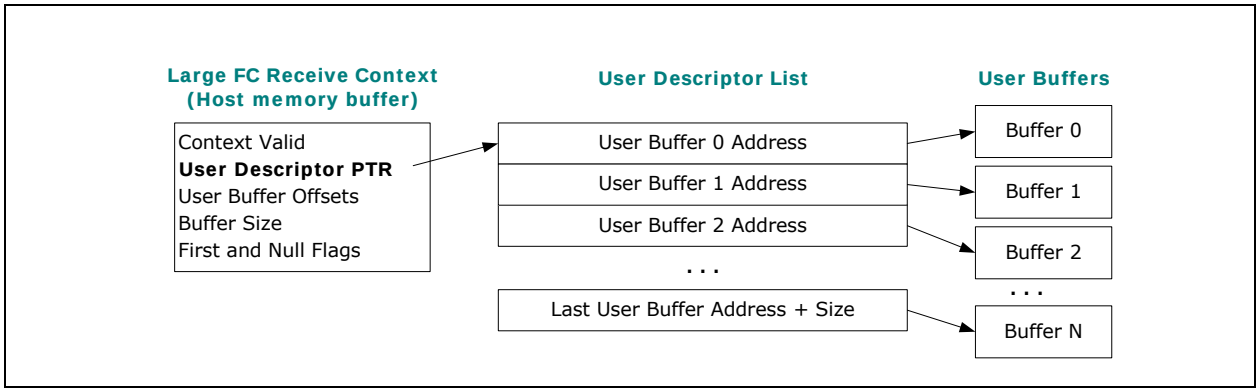


Figure 7-56 Large FC Receive Context Related to the User Buffers

DMA Context Valid (1 bit) and **Filter Context Valid (1 bit)** — These bits indicates the validity of this context.

Note: 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 7.13.3.3.10.

Filter First (1 bit) and DMA First (1 bit) — 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 7.13.3.3.14). The *Filter First* flag should be cleared by software when programming the context. Hardware sets this bit when the filter unit recognizes the first packet that matches a valid context. The *DMA First* bit should be cleared by software when programming the context. Hardware sets this bit when the DMA unit received packet that matches a valid context and marked as first by the filter unit.

Buffer Count (8 bit) — This field defines the number of remaining user buffers in the list. At programming time, software sets the buffer count to the number of the allocated user buffers. During reception, hardware decrements the buffer count as each of them completes. The number of active buffers equals the buffer count value while 0x00 equals 256.

Buffer Size (2 bit) — This field defines the user buffer size used in this context. It can be 4 KB, 8 KB, 16 KB or 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.

User Buffer Offset (16 bit) — This field defines the byte offset within the current buffer to which the next packet should be posted. At context programming, the software sets the user buffer offset to the beginning of the first buffer. During reception, hardware updates this field at the end of each packet processing for the next received packet.

Last User Buffer Size (16 bit) — This field defines the size of the last user buffer in byte units.

User Descriptor PTR (8 byte) — The user buffers are indicated by a list of pointers named as user descriptors (see Section 7.13.3.3.9 for a description of the user descriptors). The user descriptor PTR in the FC context is a pointer to the user descriptor list. At programming time, software sets the user descriptor PTR to the beginning of the user descriptor list. During reception, 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 bit) — 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_ID for checking in-order reception as described in Section 7.13.3.3.7. Hardware updates the SEQ_ID in the context table according to the value of the SEQ_ID in the incoming frame. The initialization value during programming could be of any value. For future compatibility software should set it to zero.

SEQ_CNT (16 bit) — SEQ_CNT is an index of the expected FC frames within a sequence or within the entire exchange depending on the target implementation. Hardware uses the SEQ_CNT for checking in-order reception as described in Section 7.13.3.3.7. On read context, software should initialize SEQ_CNT to zero. On write context, software should initialize SEQ_CNT to SEQ_CNT + 1 of the last packet of the same exchange received from the initiator. For each packet received in-order, the hardware sets the SEQ_CNT in the context to the value of the received SEQ_CNT + 1.

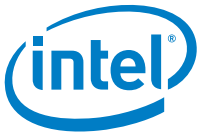
PARAM (32 bit) — The *PARAM* field in the FC header may indicate the data offset within the FC IO exchange. It is indicated as an offset by the *Relative Offset Present* bit in the F_CTL field in the FC header. In this case, the *PARAM* field indicates the expected value of the next received packet. At programming time, software should initialize it to zero. During reception, 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.

7.13.3.3.7 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) in order 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.



7.13.3.3.8 Accessing the Large FC Receive Context

The X540 supports a large number of FC contexts while each context contains about 16 bytes. In order 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.

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

DMA Context Read — Software should program the FCDMARW register as follows and then read the context on the FCPTRL, FCPTRH, FCBUFF and FCDMARW registers

- Software should initiate two consecutive write cycles to the FCDMARW register with the following setting: FCoESEL should be set to the required FCoE read index while both *WE* and *RE* should be set to 0b.
- 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.

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.

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.

7.13.3.3.9 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 7-95 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.	

7.13.3.3.10 Invalidating FC Receive 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 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.

7.13.3.3.11 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 7.13.3.3.10.

7.13.3.3.12 OX_ID and RX_ID Pool Management

As previously indicated, hardware enables Large FC receive offload for up to 512 concurrent outstanding read or write requests. In some cases more than 512 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 7-96 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 7-96 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
. . .		. . .		. . .		. . .
. . .		. . .		. . .		510
. . .		. . .		. . .		511
. . .		510		510		44
. . .		511		511		21
. . .				44		9
. . .				21		0
. . .				9		75
510				0		10
511						38

SW 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 511) only for those ones.

7.13.3.3.13 Packets and Headers Indication in the Legacy Receive Queue

The following packets or packets' 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 7.13.3.3.14
- Headers of packets posted to the user buffers by the DDP logic that contain meaningful data (as detailed in Section 7.13.3.3.2 and Section 7.13.3.3.3)

There are a few new fields in the receive descriptor dedicated to FCoE described in Section 7.1.6.2:

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.

7.13.3.3.14 Exception Handling

Table 7-97 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 then the EOF exception, any high priority exception hides all other ones with a lower priority.

Table 7-97 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 regardless of CRC correctness (independent of FCRXCTRL.SavBad setting). If the packet matches the FCoE redirection table, the packet is posted to Rx queue index defined by the FCRETA[0]. RDESC.STATUS.FCSTAT = 00b. RDESC.ERRORS.FCERR = 100b.
Rx packet with ESP option header.	If it matches the DDP context then 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 then 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 then 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 (see note 2).	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. 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 (see note 3).	Filter context parameters are updated while DMA context parameters are left intact. RDESC.STATUS.FCSTAT = 01b / 10b / 11b. RDESC.ERRORS.FCERR = 011b or 101b.

Table 7-97 Exception Error Table (Continued)

Event Description	Actions and Indications
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.
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 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.

Note (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.

Note (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.

7.13.3.3.15 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.

The X540 enables LLI for FCP_RSP frames or last FC data frame in a sequence with active *Sequence Initiative* flag. Any such frames can generate an LLI interrupt if the *FCOELLI* bit in the FCRXCTRL register is set.

Similarly, reducing the latency associated with processing FC write exchange can increase responder performance. During an FC write exchange, the originator handles the initiative to the responder after it sends all the data that the responder is ready to receive. Therefore, the X540 enables LLI after receiving the last packet in a sequence with the *Sequence Initiative* bit set in the F_CTL field. The LLI is enabled by the same *FCOELLI* bit in the FCRXCTRL register previously indicated.

7.14 Reliability

7.14.1 Memory Integrity Protection

All the X540 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 X540 halts the traffic and sets the ECC error interrupt. Software is then required to initiate a complete initialization cycle to resume nominal operation.

7.14.2 PCIe Error Handling

For PCIe error events and error reporting see Section 3.1.7.

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

8.0 Programming Interface

8.1 General

The X540's address space is mapped into four regions with PCI Base Address Registers listed in Table 8-1 and explained more in Section 9.3.6.1 and Section 9.3.6.2.

Table 8-1 X540 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	64 KB - 8 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

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 128 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 6.2.1. 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.

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 in 32-bit addressing or BAR0/BAR1 in 64-bit addressing). The Flash is only accessible if enabled through the NVM Initialization Control Word. For accesses, the offset from the CSR BAR minus 128 KB corresponds to the physical address within the external Flash device.

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 MSIX registers are described in the MSI-X Table Registers section.

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:

Offset	Abbreviation	Name	RW	Size
0x00	IOADDR	Internal Register and Internal Memory. 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.	N/A	N/A

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.

Because only a particular range is addressable, the upper bits of this register are hard coded to zero. Bits 31 through 20 are not write-able and always read back as 0b.

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 X540 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; 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 X540 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 8-1.

Table 8-1. 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:
   a. Bit 31 (*IOADDR.Configuration IO Access Enable*) of the IOADDR register should be set to 1b.
   b. 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 un-intentional CSR read operations (that might cause clear by read) by other applications scanning the configuration space.

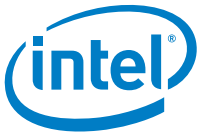
Software reads data from an internal CSR via the configuration space in the following manner:

1. CSR address is written to the IOADDR register where:
   a. Bit 31 (*IOADDR.Configuration IO Access Enable*) of the IOADDR register should be set to 1b.
   b. 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.
   a. 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 IO_Sup bit (bit 12) in PCIe Control 3 word (offset 0x07) must be cleared in the NVM when the IO_cfg_en bit is set, and vice versa.

8.1.4 Register Terminology

Shorthand	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.
RO	Read Only. If a register is read only, writes to this register have no effect.
WO	Write Only. Reading this register might not return a meaningful value.
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.
RC	Read Clear. A register bit with this attribute is cleared after read. Writes have no effect on the bit value.
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.
Reserved	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.

8.1.5 MSI-X BAR Register Summary PF

See Section 9.3.6.1 for the MSI-X BAR offset in 32-bit and 64-bit BAR options.

Category	BAR 3 Offset	Alias Offset	Abbreviation	Name	RW
MSI-X	0x0000 — (N-1)*0x10	N/A	MSIXTADD	MSIX Table Entry Lower Address	RW
MSI-X	0x0004 — (N-1)*0x10	N/A	MSIXTUADD	MSIX Table Entry Upper Address	RW
MSI-X	0x0008 — (N-1)*0x10	N/A	MSIXTMSG	MSIX Table Entry Message	RW
MSI-X	0x000C — (N-1)*0x10	N/A	MSIXTVCTRL	MSIX Table Vector Control	RW
MSI-X	0x2000 — 0x200C	N/A	MSIXPBA	MSI-X Pending Bit Array	RO

8.1.6 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.

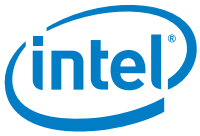
Section 7.10.2.7.2 defines the correspondence of queue indices between the PF and the VFs. For example, when in configuration for 32 VFs, queues 124-127 in the PF correspond to queues [3:0] of VF# 31.

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, then 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).

8.1.7 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 (like 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.

8.1.8 MSI—X Register Summary VF — BAR 3

Virtual Address	Physical Address Base (+ VFn *0x30)	Abbreviation	Name
0x0000 + n*0x10, n=0...2	0x00010	MSIXTADD	MSIX table entry lower address
0x0004 + n*0x10, n=0...2	0x00018	MSIXTUADD	MSIX table entry upper address
0x0008 + n*0x10, n=0...2	0x00028	MSIXTMSG	MSIX table entry message
0x000C + n*0x10, n=0...2	N/A	MSIXTVCTRL	MSIX table vector control
Max(Page Size, 0x2000)	N/A	MSIXPBA	MSI-X Pending bit array

8.1.8.1 MSI—X Table Entry Lower Address — MSIXTADD (BAR3: 0x0000 + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

8.1.8.2 MSI—X Table Entry Upper Address — MSIXTUADD (BAR3: 0x0004 + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

8.1.8.3 MSI—X Table Entry Message — MSIXTMSG (BAR3: 0x0008 + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

8.1.8.4 MSI—X Table Entry Vector Control — MSIXVCTRL (BAR3: 0x000C + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

8.1.8.5 MSIXPBA (BAR3: 0x02000; RO) — MSIXPBA Bit Description

Field	Bit(s)	Init Val.	Description
Pending Bits	2:0	0x0	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	Reserved.

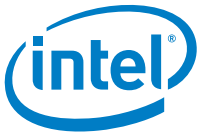
Note: If a page size larger than 8 KB is programmed in the IOV structure, the address of the MSIX PBA table moves to be page aligned.

8.2 Device Registers - PF

8.2.1 BAR3 Registers Summary

Table 8-1 BAR3 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Block	Section
MSI-X Register Summary PF - BAR 3				
0x00000000 + 0x10*n, n=0...255	MSIXTADD[n]	MSI-X Table Entry Lower Address	MSI-X	Section 8.2.2.1.1
0x00000004 + 0x10*n, n=0...255	MSIXTUADD[n]	MSI-X Table Entry Upper Address	MSI-X	Section 8.2.2.1.2
0x00000008 + 0x10*n, n=0...255	MSIXTMSG[n]	MSI-X Table Entry Message	MSI-X	Section 8.2.2.1.3
0x0000000C + 0x10*n, n=0...255	MSIXVCTRL[n]	MSI-X Table Entry Vector Control	MSI-X	Section 8.2.2.1.4
0x00002000 + 0x4*n, n=0...7	MSIXPBA[n]	MSIXPBA Bit Description	MSI-X	Section 8.2.2.1.5



8.2.2 Detailed Register Description - PF BAR3

8.2.2.1 MSI-X Register Summary PF - BAR 3

See Memory and IO Base Address Registers for the MSI-X BAR offset in 32-bit and 64-bit BAR options.

8.2.2.1.1 MSI-X Table Entry Lower Address - MSIXTADD[n] (0x00000000 + 0x10*n, n=0...255)

See MSI-X Table Structure for details of this register.

Field	Bit(s)	Init.	Access Type	Description
MSIXTADD10	1:0	0x0	RW	Message Address. For proper Dword alignment, software must always write zeros to these two bits; otherwise, the result is undefined. The state of these bits after reset must be 0b. These bits are permitted to be read-only or read/write.
MSIXTADD	31:2	0x0	RW	Message Address. System-specified message lower address. For MSI-X messages, the contents of this field from an MSI-X table entry specifies the lower portion of the Dword-aligned address (AD[31:02]) for the memory write transaction. This field is read/write.

8.2.2.1.2 MSI-X Table Entry Upper Address - MSIXTUADD[n] (0x00000004 + 0x10*n, n=0...255)

See MSI-X Table Structure for details of this register.

Field	Bit(s)	Init.	Access 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.

8.2.2.1.3 MSI-X Table Entry Message - MSIXTMSG[n] (0x00000008 + 0x10*n, n=0...255)

See MSI-X Table Structure for details of this register.

Field	Bit(s)	Init.	Access 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.

8.2.2.1.4 MSI-X Table Entry Vector Control - MSIXVCTRL[n] (0x0000000C + 0x10*n, n=0...255)

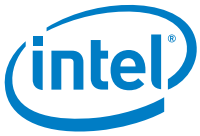
See MSI-X Table Structure for details of this register.

Field	Bit(s)	Init.	Access 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.

8.2.2.1.5 MSIXPBA Bit Description - MSIXPBA[n] (0x00002000 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access 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 to MSI-X vector 32 * 'N' + 'i', 'N' = 0...3.

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.



8.2.3 BAR0 Registers Summary

Table 8-2 BAR0 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Block	Section
General Control Registers				
0x00000000	CTRL	Device Control Register	Target	Section 8.2.4.1.1
0x00000004	CTRL	Device Control Register	Target	Section 8.2.4.1.2
0x00000008	STATUS	Device Status Register	Target	Section 8.2.4.1.3
0x00000018	CTRL_EXT	Extended Device Control Register	Target	Section 8.2.4.1.4
0x00000020	ESDP	Extended SDP Control	Target	Section 8.2.4.1.5
0x00000028	PHY_GPIO	PHY GPIO Register	Target	Section 8.2.4.1.6
0x00000030	MAC_GPIO	MAC GPIO Register	Target	Section 8.2.4.1.7
0x00000100	PHYINT_STATUS 0	PHY Interrupt Status Register 0	Target	Section 8.2.4.1.8
0x00000104	PHYINT_STATUS 1	PHY Interrupt Status Register 1	Target	Section 8.2.4.1.9
0x00000108	PHYINT_STATUS 2	PHY Interrupt Status Register 2	Target	Section 8.2.4.1.10
0x00000200	LEDCTL	LED Control	Target	Section 8.2.4.1.11
0x00005078	EXVET	Extended VLAN Ether Type	RX-Filter	Section 8.2.4.1.12
NVM Registers				
0x00010010	EEC	EEPROM-Mode Control Register	FLEEP	Section 8.2.4.2.1
0x00010014	EERD	EEPROM-Mode Read Register	FLEEP	Section 8.2.4.2.2
0x00010018	EEWR	EEPROM-Mode Write Register	FLEEP	Section 8.2.4.2.3
0x0001001C	FLA	Flash Access Register	FLEEP	Section 8.2.4.2.4
0x00010110	EEMNGCTL	Manageability EEPROM-Mode Control Register	FLEEP	Section 8.2.4.2.5
0x00010114	EEMNGDATA	Manageability EEPROM-Mode Read/ Write Data	FLEEP	Section 8.2.4.2.6
0x00010118	FLMNGCTL	Manageability Flash Control Register	FLEEP	Section 8.2.4.2.7
0x0001011C	FLMNGDATA	Manageability Flash Read/Write Data	FLEEP	Section 8.2.4.2.8

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x0001013C	FLOP	FLASH Opcode Register	FLEEP	Section 8.2.4.2.9
0x00010200	GRC	General Receive Control	FLEEP	Section 8.2.4.2.10
0x00010210	SRAMREL	Shadow RAM Reload	FLEEP	Section 8.2.4.2.11
Flow Control Registers				
0x00003008	PFCTOP	Priority Flow Control Type Opcode	MAC	Section 8.2.4.3.1
0x0000431C	PFCTOP	Priority Flow Control Type Opcode	MAC	Section 8.2.4.3.2
0x00003200 + 0x4*n, n=0...3	FCTTVN[n]	Flow Control Transmit Timer Value n	DBU-RX	Section 8.2.4.3.3
0x00003220 + 0x4*n, n=0...7	FCRTL[n]	Flow Control Receive Threshold Low	DBU-RX	Section 8.2.4.3.4
0x00003260 + 0x4*n, n=0...7	FCRTH[n]	Flow Control Receive Threshold High	DBU-RX	Section 8.2.4.3.5
0x000032A0	FCRTV	Flow Control Refresh Threshold Value	DBU-RX	Section 8.2.4.3.6
0x0000CE00	TFCS	Transmit Flow Control Status	DBU-TX	Section 8.2.4.3.7
0x00003D00	FCCFG	Flow Control Configuration	DBU-RX	Section 8.2.4.3.8
PCIe Registers				
0x00011000	GCR	PCIe Control Register	PCIe	Section 8.2.4.4.1
0x00011004	PCIEPIPEADR	PCIe SerDes Configuration Register	PCIe	Section 8.2.4.4.2
0x00011008	PCIEPIPEDAT	PCIe SerDes Data Register Tab	PCIe	Section 8.2.4.4.3
0x00011010	GSCL_1	PCIe Statistic Control Register #1	PCIe	Section 8.2.4.4.4
0x00011014	GSCL_2	PCIe Statistic Control Registers #2	PCIe	Section 8.2.4.4.5
0x00011030 + 0x4*n, n=0...3	GSCL_5_8[n]	PCIe Statistic Control Register #5...#8	PCIe	Section 8.2.4.4.6
0x00011020 + 0x4*n, n=0...3	GSCN_0_3[n]	PCIe Statistic Counter Registers #0...#3	PCIe	Section 8.2.4.4.7
0x00010150	FACTPS	Function Active and Power State to Manageability	FLEEP	Section 8.2.4.4.8
0x00011040	PCIEPHYADR	PCIe Link/PHY Configuration Register	PCIe	Section 8.2.4.4.9
0x00011044	PCIEPHYDAT	PCIe Link/PHY Data Register Tab	PCIe	Section 8.2.4.4.10
0x00010140	SWSM	Software Semaphore Register	FLEEP	Section 8.2.4.4.11

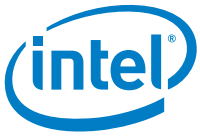


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00010148	FWSM	Firmware Semaphore Register	FLEEP	Section 8.2.4.4.12
0x00010160	SW_FW_SYNC	Software-Firmware Synchronization	FLEEP	Section 8.2.4.4.13
0x00011050	GCR_EXT	PCIe Control Extended Register	PCIe	Section 8.2.4.4.14
0x00011064	MREVID	Mirrored Revision ID	PCIe	Section 8.2.4.4.15
0x000110B0	PICAUSE	PCIe Interrupt Cause	PCIe	Section 8.2.4.4.16
0x000110B8	PIENA	PCIe Interrupt Enable	PCIe	Section 8.2.4.4.17
Interrupt Registers				
0x00000800	EICR	Extended Interrupt Cause Register	Interrupt	Section 8.2.4.5.1
0x00000808	EICS	Extended Interrupt Cause Set Register	Interrupt	Section 8.2.4.5.2
0x00000880	EIMS	Extended Interrupt Mask Set/Read Register	Interrupt	Section 8.2.4.5.3
0x00000888	EIMC	Extended Interrupt Mask Clear Register	Interrupt	Section 8.2.4.5.4
0x00000810	EIAC	Extended Interrupt Auto Clear Register	Interrupt	Section 8.2.4.5.5
0x00000890	EIAM	Extended Interrupt Auto Mask Enable register	Interrupt	Section 8.2.4.5.6
0x00000A90	EICS1	Extended Interrupt Cause Set Registers 1	Interrupt	Section 8.2.4.5.7
0x00000A94	EICS2	Extended Interrupt Cause Set Registers 2	Interrupt	Section 8.2.4.5.8
0x00000AA0	EIMS1	Extended Interrupt Mask Set/Read Registers	Interrupt	Section 8.2.4.5.9
0x00000AA4	EIMS2	Extended Interrupt Mask Set/Read Registers	Interrupt	Section 8.2.4.5.10
0x00000AB0	EIMC1	Extended Interrupt Mask Clear Registers 1	Interrupt	Section 8.2.4.5.11
0x00000AB4	EIMC2	Extended Interrupt Mask Clear Registers 2	Interrupt	Section 8.2.4.5.12
0x00000AD0	EIAM1	Extended Interrupt Auto Mask Enable registers 1	Interrupt	Section 8.2.4.5.13
0x00000AD4	EIAM2	Extended Interrupt Auto Mask Enable registers 2	Interrupt	Section 8.2.4.5.14
0x00000894	EITRSEL	MSIX to EITR Select	Interrupt	Section 8.2.4.5.15

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00000820 + 0x4*n, n=0...23 and 0x00012300 + 0x4*(n-24), n=24...128	EITR[n]	Extended Interrupt Throttle Registers	Interrupt	Section 8.2.4.5.16
0x0000E800 + 0x4*n, n=0...127	L34TIMIR[n]	L3 L4 Tuples Immediate Interrupt Rx	DBU-RX	Section 8.2.4.5.17
0x0000EC90	LLITHRESH	LLI Size Threshold	DBU-RX	Section 8.2.4.5.18
0x0000EC60	IMIRVP	Immediate Interrupt Rx VLAN Priority Register	DBU-RX	Section 8.2.4.5.19
0x00005AC0	IMIRVP	Immediate Interrupt Rx VLAN Priority Register	DBU-RX	Section 8.2.4.5.20
0x00000900 + 0x4*n, n=0...63	IVAR[n]	Interrupt Vector Allocation Registers	Interrupt	Section 8.2.4.5.21
0x00000A00	IVAR_MISC	Miscellaneous Interrupt Vector Allocation	Interrupt	Section 8.2.4.5.22
0x00012000 + 0x4*n, n=0...128	RSCINT[n]	RSC Enable Interrupt	Interrupt	Section 8.2.4.5.23
0x00000898	GPIE	General Purpose Interrupt Enable	Interrupt	Section 8.2.4.5.24
MSI-X Table Registers				
0x00011068	PBACL	MSI-X PBA Clear	PCIe	Section 8.2.4.6.1
0x000110C0 + 0x4*n, n=0...7	PBACL[n]	MSI-X PBA Clear	PCIe	Section 8.2.4.6.2
Receive Registers				
0x00005080	FCTRL	Filter Control Register	RX-Filter	Section 8.2.4.7.1
0x00005088	VLNCTRL	VLAN Control Register	RX-Filter	Section 8.2.4.7.2
0x00005090	MCSTCTRL	Multicast Control Register	RX-Filter	Section 8.2.4.7.3
0x0000EA00 + 0x4*n, n=0...63	PSRTYPE[n]	Packet Split Receive Type Register	DBU-RX	Section 8.2.4.7.4
0x00005480 + 0x4*n, n=0...15	PSRTYPE[n]	Packet Split Receive Type Register	DBU-RX	Section 8.2.4.7.5
0x00005000	RXCSUM	Receive Checksum Control	RX-Filter	Section 8.2.4.7.6
0x00005008	RFCTL	Receive Filter Control Register	RX-Filter	Section 8.2.4.7.7
0x00005200 + 0x4*n, n=0...127	MTA[n]	Multicast Table Array	RX-Filter	Section 8.2.4.7.8

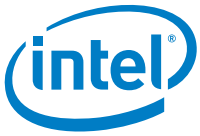


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x0000A200 + 0x8*n, n=0...127	RAL[n]	Receive Address Low	RX-Filter	Section 8.2.4.7.9
0x0000A204 + 0x8*n, n=0...127	RAH[n]	Receive Address High	RX-Filter	Section 8.2.4.7.10
0x0000A600 + 0x4*n, n=0...255	MPSAR[n]	MAC Pool Select Array	RX-Filter	Section 8.2.4.7.11
0x0000A000 + 0x4*n, n=0...127	VFTA[n]	VLAN Filter Table Array	RX-Filter	Section 8.2.4.7.12
0x0000EC80	MRQC	Multiple Receive Queues Command Register	DBU-RX	Section 8.2.4.7.13
0x00005818	MRQC	Multiple Receive Queues Command Register	DBU-RX	Section 8.2.4.7.14
0x0000EC70	RQTC	RSS queues per Traffic class register	DBU-RX	Section 8.2.4.7.15
0x00005C80 + 0x4*n, n=0...9	RSSRK[n]	RSS Random Key Register	DBU-RX	Section 8.2.4.7.16
0x0000EB80 + 0x4*n, n=0...9	RSSRK[n]	RSS Random Key Register	DBU-RX	Section 8.2.4.7.17
0x00005C00 + 0x4*n, n=0...31	RETA[n]	Redirection Table	DBU-RX	Section 8.2.4.7.18
0x0000EB00 + 0x4*n, n=0...31	RETA[n]	Redirection Table	DBU-RX	Section 8.2.4.7.19
0x0000E000 + 0x4*n, n=0...127	SAQF[n]	Source Address Queue Filter	DBU-RX	Section 8.2.4.7.20
0x0000E200 + 0x4*n, n=0...127	DAQF[n]	Destination Address Queue Filter	DBU-RX	Section 8.2.4.7.21
0x0000E400 + 0x4*n, n=0...127	SDPQF[n]	Source Destination Port Queue Filter	DBU-RX	Section 8.2.4.7.22
0x0000E600 + 0x4*n, n=0...127	FTQF[n]	Five tuple Queue Filter	DBU-RX	Section 8.2.4.7.23
0x0000EC30	SYNQF	SYN Packet Queue Filter	DBU-RX	Section 8.2.4.7.24
0x00005128 + 0x4*n, n=0...7	ETQF[n]	EType Queue Filter	RX-Filter	Section 8.2.4.7.25
0x0000EC00 + 0x4*n, n=0...7	ETQS[n]	EType Queue Select	DBU-RX	Section 8.2.4.7.26
Receive DMA Registers				
0x00001000 + 0x40*n, n=0...63 and 0x0000D000 + 0x40*(n-64), n=64...127	RDBAL[n]	Receive Descriptor Base Address Low	DMA-RX	Section 8.2.4.8.1

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00001004 + 0x40*n, n=0...63 and 0x0000D004 + 0x40*(n-64), n=64...127	RDBAH[n]	Receive Descriptor Base Address High	DMA-RX	Section 8.2.4.8.2
0x00001008 + 0x40*n, n=0...63 and 0x0000D008 + 0x40*(n-64), n=64...127	RDLEN[n]	Receive Descriptor Length	DMA-RX	Section 8.2.4.8.3
0x00001010 + 0x40*n, n=0...63 and 0x0000D010 + 0x40*(n-64), n=64...127	RDH[n]	Receive Descriptor Head	DMA-RX	Section 8.2.4.8.4
0x00001018 + 0x40*n, n=0...63 and 0x0000D018 + 0x40*(n-64), n=64...127	RDT[n]	Receive Descriptor Tail	DMA-RX	Section 8.2.4.8.5
0x00001028 + 0x40*n, n=0...63 and 0x0000D028 + 0x40*(n-64), n=64...127	RXDCTL[n]	Receive Descriptor Control	DMA-RX	Section 8.2.4.8.6
0x00001014 + 0x40*n, n=0...63 and 0x0000D014 + 0x40*(n-64), n=64...127	SRRCTL[n]	Split Receive Control Registers	DMA-RX	Section 8.2.4.8.7
0x00002100 + 0x4*n, n=0...15	SRRCTL[n]	Split Receive Control Registers	DMA-RX	Section 8.2.4.8.8
0x00002F00	RDRXCTL	Receive DMA Control Register	DMA-RX	Section 8.2.4.8.9
0x00003C00 + 0x4*n, n=0...7	RXPBSIZE[n]	Receive Packet Buffer Size	DBU-RX	Section 8.2.4.8.10
0x00003000	RXCTRL	Receive Control Register	DBU-RX	Section 8.2.4.8.11
0x00003028	RSCDBU	RSC Data Buffer Control Register	DBU-RX	Section 8.2.4.8.12
0x0000102C + 0x40*n, n=0...63 and 0x0000D02C + 0x40*(n-64), n=64...127	RSCCTL[n]	RSC Control	DMA-RX	Section 8.2.4.8.13
Transmit Registers				
0x00008100	DTXMXSZRQ	DMA Tx TCP Max Allow Size Requests	DMA-TX	Section 8.2.4.9.1
0x00004A80	DMATXCTL	DMA Tx Control	DMA-TX	Section 8.2.4.9.2
0x00004A88	DTXTCPFLGL	DMA Tx TCP Flags Control Low	DMA-TX	Section 8.2.4.9.3
0x00004A8C	DTXTCPFLGH	DMA Tx TCP Flags Control High	DMA-TX	Section 8.2.4.9.4

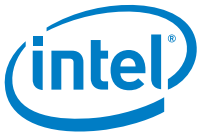


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00006000 + 0x40*n, n=0...127	TDBAL[n]	Transmit Descriptor Base Address Low	DMA-TX	Section 8.2.4.9.5
0x00006004 + 0x40*n, n=0...127	TDBAH[n]	Transmit Descriptor Base Address High	DMA-TX	Section 8.2.4.9.6
0x00006008 + 0x40*n, n=0...127	TDLEN[n]	Transmit Descriptor Length	DMA-TX	Section 8.2.4.9.7
0x00006010 + 0x40*n, n=0...127	TDH[n]	Transmit Descriptor Head	DMA-TX	Section 8.2.4.9.8
0x00006018 + 0x40*n, n=0...127	TDT[n]	Transmit Descriptor Tail	DMA-TX	Section 8.2.4.9.9
0x00006028 + 0x40*n, n=0...127	TXDCTL[n]	Transmit Descriptor Control	DMA-TX	Section 8.2.4.9.10
0x00006038 + 0x40*n, n=0...127	TDWBAL[n]	Tx Descriptor Completion Write Back Address Low	DMA-TX	Section 8.2.4.9.11
0x0000603C + 0x40*n, n=0...127	TDWBAH[n]	Tx Descriptor Completion Write Back Address High	DMA-TX	Section 8.2.4.9.12
0x0000CC00 + 0x4*n, n=0...7	TXPBSIZE[n]	Transmit Packet Buffer Size	DBU-TX	Section 8.2.4.9.13
0x0000CD10	MNGTXMAP	Manageability Transmit TC Mapping	DBU-TX	Section 8.2.4.9.14
0x00008120	MTQC	Multiple Transmit Queues Command Register	DMA-TX	Section 8.2.4.9.15
0x00004950 + 0x4*n, n=0...7	TXPBTHRESH[n]	Tx Packet Buffer Threshold	DMA-TX	Section 8.2.4.9.16
DCB Registers				
0x00002430	RTRPCS	DCB Receive Packet plane Control and Status	DMA-RX	Section 8.2.4.10.1
0x00004900	RTTDCS	DCB Transmit Descriptor Plane Control and Status	DMA-TX	Section 8.2.4.10.2
0x0000CD00	RTTPCS	DCB Transmit Packet plane Control and Status	DBU-TX	Section 8.2.4.10.3
0x00003020	RTRUP2TC	DCB Receive User Priority to Traffic Class	DBU-RX	Section 8.2.4.10.4
0x0000C800	RTTUP2TC	DCB Transmit User Priority to Traffic Class	DBU-TX	Section 8.2.4.10.5
0x00002140 + 0x4*n, n=0...7	RTRPT4C[n]	DCB Receive Packet plane T4 Config	DMA-RX	Section 8.2.4.10.6
0x000082E0 + 0x4*n, n=0...3	TXLLQ[n]	Strict Low Latency Tx Queues	DMA-TX	Section 8.2.4.10.7

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00004910 + 0x4*n, n=0...7	RTTDT2C[n]	DCB Transmit Descriptor Plane T2 Config	DMA-TX	Section 8.2.4.10.8
0x0000CD20 + 0x4*n, n=0...7	RTTPT2C[n]	DCB Transmit Packet plane T2 Config	DBU-TX	Section 8.2.4.10.9
0x00004980	RTTQCNRM	DCB Transmit QCN Rate-Scheduler MMW	DMA-TX	Section 8.2.4.10.10
0x00004904	RTTDQSEL	DCB Transmit Descriptor Plane Queue Select	DMA-TX	Section 8.2.4.10.11
0x00004908	RTTDT1C	DCB Transmit Descriptor Plane T1 Config	DMA-TX	Section 8.2.4.10.12
0x00004984	RTTQCNRC	DCB Transmit QCN Rate-Scheduler Config	DMA-TX	Section 8.2.4.10.13
0x00004988	RTTQCNRS	DCB Transmit QCN Rate-Scheduler Status	DMA-TX	Section 8.2.4.10.14
0x00008B00	RTTQCNCR	DCB Transmit QCN Control Register	SEC-TX	Section 8.2.4.10.15
0x00004A90	RTTQCNTG	DCB Transmit QCN Tagging	DMA-TX	Section 8.2.4.10.16
0x0000498C	RTTQCNRR	DCB Transmit QCN Rate Reset	DMA-TX	Section 8.2.4.10.17
DCA Registers				
0x0000100C + 0x40*n, n=0...63 and 0x0000D00C + 0x40*(n-64), n=64...127	DCA_RXCTRL[n]	Rx DCA Control Register	DMA-RX	Section 8.2.4.11.1
0x00002200 + 0x4*n, n=0...15	DCA_RXCTRL[n]	Rx DCA Control Register	DMA-RX	Section 8.2.4.11.2
0x0000600C + 0x40*n, n=0...127	DCA_TXCTRL[n]	Tx DCA Control Registers	DMA-TX	Section 8.2.4.11.3
0x00011070	DCA_ID	DCA Requester ID Information Register	PCIe	Section 8.2.4.11.4
0x00011074	DCA_CTRL	DCA Control Register	PCIe	Section 8.2.4.11.5
Security Registers				
0x00008800	SECTXCTRL	Security Tx Control	SEC-TX	Section 8.2.4.12.1
0x00008804	SECTXSTAT	Security Tx Status	SEC-TX	Section 8.2.4.12.2
0x0000880C	SECTXBUFFAE	Security Tx Buffer Almost Empty	SEC-TX	Section 8.2.4.12.4
0x00008810	SECTXMINIFG	Security Tx Buffer Minimum IFG	SEC-TX	Section 8.2.4.12.4
0x00008D00	SECRXCTRL	Security Rx Control	SEC-RX	Section 8.2.4.12.5

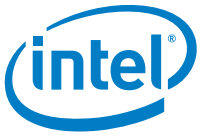


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00008D04	SECRXSTAT	Security Rx Status	SEC-RX	Section 8.2.4.12.6
IPsec Registers				
0x00008900	IPSTXIDX	IPsec Tx Index	SEC-TX	Section 8.2.4.13.1
0x00008908 + 0x4*n, n=0...3	IPSTXKEY[n]	IPsec Tx Key Registers	SEC-TX	Section 8.2.4.13.2
0x00008904	IPSTXSALT	IPsec Tx Salt Register	SEC-TX	Section 8.2.4.13.3
0x00008E00	IPSRXIDX	IPsec Rx Index	SEC-RX	Section 8.2.4.13.4
0x00008E04 + 0x4*n, n=0...3	IPSRXIPADDR[n]	IPsec Rx IP address Register	SEC-RX	Section 8.2.4.13.5
0x00008E14	IPSRXSPI	IPsec Rx SPI Register	SEC-RX	Section 8.2.4.13.6
0x00008E18	IPSRXIPIDX	IPsec Rx SPI Register IP Index	SEC-RX	Section 8.2.4.13.7
0x00008E1C + 0x4*n, n=0...3	IPSRXKEY[n]	IPsec Rx Key Register	SEC-RX	Section 8.2.4.13.8
0x00008E2C	IPSRXSALT	IPsec Rx Salt Register	SEC-RX	Section 8.2.4.13.9
0x00008E30	IPSRXMOD	IPsec Rx Mode Register	SEC-RX	Section 8.2.4.13.10
Timers Registers				
0x0000004C	TCPTIMER	TCP Timer	Target	Section 8.2.4.14.1
0x00000048	FRTIMER	Free Running Timer	Target	Section 8.1.4.14.2
FCoE Registers				
0x00004A98	TSOFF	Tx FC SOF Flags Register	DMA-TX	Section 8.2.4.15.1
0x00004A94	TEOFF	Tx FC EOF Flags Register	DMA-TX	Section 8.2.4.15.2
0x000051F8	RSOFF	Rx FC SOF Flags Register	RX-Filter	Section 8.2.4.15.3
0x00005158	REOFF	Rx FC EOF Flags Register	RX-Filter	Section 8.2.4.15.4
0x00005100	FCRXCTRL	FC Receive control	RX-Filter	Section 8.2.4.15.5
0x0000ED00	FCRECTL	FCoE Redirection Control	DBU-RX	Section 8.2.4.15.6
0x0000ED10 + 0x4*n, n=0...7	FCRETA[n]	FCoE Redirection Table	DBU-RX	Section 8.2.4.15.7
0x00002410	FCPTRL	FC User Descriptor PTR Low	DMA-RX	Section 8.2.4.15.8
0x00002414	FCPTRH	FC User Descriptor PTR High	DMA-RX	Section 8.2.4.15.9

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00002418	FCBUFF	FC Buffer Control	DMA-RX	Section 8.2.4.15.10
0x00002420	FCDMARW	FC Receive DMA RW	DMA-RX	Section 8.2.4.15.11
0x00005108	FCFLT	FC FLT Context	RX-Filter	Section 8.2.4.15.12
0x000051D8	FCPARAM	FC Offset Parameter	RX-Filter	Section 8.2.4.15.13
0x00005110	FCFLTRW	FC Filter RW Control	RX-Filter	Section 8.2.4.15.14
Flow Director Registers				
0x0000EE00	FDIRCTRL	Flow Director Filters control register	DBU-RX	Section 8.2.4.16.1
0x0000EE68	FDIRHKEY	Flow Director Filters Lookup Table HASH Key	DBU-RX	Section 8.2.4.16.2
0x0000EE6C	FDIRSKEY	Flow Director Filters Signature HASH Key	DBU-RX	Section 8.2.4.16.3
0x0000EE3C	FDIRDIP4M	Flow Director Filters DIPv4 Mask	DBU-RX	Section 8.2.4.16.4
0x0000EE40	FDIRSIP4M	Flow Director Filters Source IPv4 Mask	DBU-RX	Section 8.2.4.16.5
0x0000EE44	FDIRTCPM	Flow Director Filters TCP Mask	DBU-RX	Section 8.2.4.16.6
0x0000EE48	FDIRUDPM	Flow Director Filters UDP Mask	DBU-RX	Section 8.2.4.16.7
0x0000EE74	FDIRIP6M	Flow Director Filters IPv6 Mask	DBU-RX	Section 8.2.4.16.8
0x0000EE70	FDIRM	Flow Director Filters Other Mask	DBU-RX	Section 8.2.4.16.9
0x0000EE38	FDIRFREE	Flow Director Filters Free	DBU-RX	Section 8.2.4.16.10
0x0000EE4C	FDIRLEN	Flow Director Filters Length	DBU-RX	Section 8.2.4.16.11
0x0000EE50	FDIRUSTAT	Flow Director Filters Usage Statistics	DBU-RX	Section 8.2.4.16.12
0x0000EE54	FDIRFSTAT	Flow Director Filters Failed Usage Statistics	DBU-RX	Section 8.2.4.16.13
0x0000EE58	FDIRMATCH	Flow Director Filters Match Statistics	DBU-RX	Section 8.2.4.16.14
0x0000EE5C	FDIRMISS	Flow Director Filters Miss Match Statistics	DBU-RX	Section 8.2.4.16.15
0x0000EE0C + 0x4*n, n=0...2	FDIRSIPV6[n]	Flow Director Filters Source IPv6	DBU-RX	Section 8.2.4.16.16
0x0000EE18	FDIRIPSA	Flow Director Filters IP SA	DBU-RX	Section 8.2.4.16.17
0x0000EE1C	FDIRIPDA	Flow Director Filters IP DA	DBU-RX	Section 8.2.4.16.18

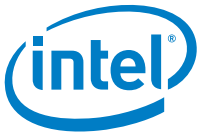


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x0000EE20	FDIRPORT	Flow Director Filters Port	DBU-RX	Section 8.2.4.16.19
0x0000EE24	FDIRVLAN	Flow Director Filters VLAN and FLEX Bytes	DBU-RX	Section 8.2.4.16.20
0x0000EE28	FDIRHASH	Flow Director Filters Hash Signature	DBU-RX	Section 8.2.4.16.21
0x0000EE2C	FDIRCMD	Flow Director Filters Command register	DBU-RX	Section 8.2.4.16.22
MAC Registers				
0x00004240	HLREG0	Core Control 0 Register	MAC	Section 8.1.4.24.1
0x00004244	HLREG1	Core Status 1 Register	MAC	Section 8.1.4.24.2
0x00004248	PAP	Pause and Pace Register	MAC	Section 8.2.4.17.3
0x0000425C	MSCA	MDI Single Command and Address	MAC	Section 8.2.4.17.4
0x00004260	MSRWD	MDI Single Read and Write Data	MAC	Section 8.2.4.17.5
0x00004268	MAXFRS	Max Frame Size	MAC	Section 8.2.4.17.6
0x000042A4	LINKS	Link Status Register	MAC	Section 8.2.4.17.7
0x000042D0	MMNGC	MAC Manageability Control Register	MAC	Section 8.2.4.17.8
0x00004294	MFLCN	MAC Flow Control Register	MAC	Section 8.2.4.17.9
0x00004330	MACC	MAC Control register	MAC	Section 8.2.4.17.10
Statistic Registers				
0x00004000	CRCERRS	CRC Error Count	STAT	Section 8.2.4.18.1
0x00004004	ILLERRC	Illegal Byte Error Count	STAT	Section 8.2.4.18.2
0x00004008	ERRBC	Error Byte Packet Count	STAT	Section 8.2.4.18.3
0x00004034	MLFC	MAC Local Fault Count	STAT	Section 8.2.4.18.4
0x00004038	MRFC	MAC Remote Fault Count	STAT	Section 8.2.4.18.5
0x00004040	RLEC	Receive Length Error Count	STAT	Section 8.2.4.18.6
0x00008780	SSVPC	Switch Security Violation Packet Count	DMA-TX	Section 8.2.4.18.7
0x000041A4	LXONRXCNT	Link XON Received Count	STAT	Section 8.2.4.18.8
0x000041A8	LXOFFRXCNT	Link XOFF Received Count	STAT	Section 8.2.4.18.9

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00004140 + 0x4*n, n=0...7	PXONRXCNT[n]	Priority XON Received Count	STAT	Section 8.2.4.18.10
0x00004160 + 0x4*n, n=0...7	PXOFFRXCNT[n]	Priority XOFF Received Count	STAT	Section 8.2.4.18.11
0x0000405C	PRC64	Packets Received [64 Bytes] Count	STAT	Section 8.2.4.18.12
0x00004060	PRC127	Packets Received [65-127 Bytes] Count	STAT	Section 8.2.4.18.13
0x00004064	PRC255	Packets Received [128-255 Bytes] Count	STAT	Section 8.2.4.18.14
0x00004068	PRC511	Packets Received [256-511 Bytes] Count	STAT	Section 8.2.4.18.15
0x0000406C	PRC1023	Packets Received [512-1023 Bytes] Count	STAT	Section 8.2.4.18.16
0x00004070	PRC1522	Packets Received [1024 to Max Bytes] Count	STAT	Section 8.2.4.18.17
0x00004078	BPRC	Broadcast Packets Received Count	STAT	Section 8.2.4.18.18
0x0000407C	MPRC	Multicast Packets Received Count	STAT	Section 8.2.4.18.19
0x00004074	GPRC	Good Packets Received Count	STAT	Section 8.2.4.18.20
0x00004088	GORCL	Good Octets Received Count Low	STAT	Section 8.2.4.18.21
0x0000408C	GORCH	Good Octets Received Count High	STAT	Section 8.2.4.18.22
0x000041B0	RXNFGPC	Good Rx Non-Filtered Packet Counter	STAT	Section 8.2.4.18.23
0x000041B4	RXNFGBCL	Good Rx Non-Filter Byte Counter Low	STAT	Section 8.2.4.18.24
0x000041B8	RXNFGBCH	Good Rx Non-Filter Byte Counter High	STAT	Section 8.2.4.18.25
0x00002F50	RXDGPC	DMA Good Rx Packet Counter	DMA-RX	Section 8.2.4.18.26
0x00002F54	RXDGBCL	DMA Good Rx Byte Counter Low	DMA-RX	Section 8.2.4.18.27
0x00002F58	RXDGBCH	DMA Good Rx Byte Counter High	DMA-RX	Section 8.2.4.18.28
0x00002F5C	RXDDPC	DMA Duplicated Good Rx Packet Counter	DMA-RX	Section 8.2.4.18.29
0x00002F60	RXDDBCL	DMA Duplicated Good Rx Byte Counter Low	DMA-RX	Section 8.2.4.18.30
0x00002F64	RXDDBCH	DMA Duplicated Good Rx Byte Counter High	DMA-RX	Section 8.2.4.18.31

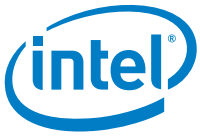


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00002F68	RXLPBKPC	DMA Good Rx LPBK Packet Counter	DMA-RX	Section 8.2.4.18.32
0x00002F6C	RXLPBKBCL	DMA Good Rx LPBK Byte Counter Low	DMA-RX	Section 8.2.4.18.33
0x00002F70	RXLPBKBCH	DMA Good Rx LPBK Byte Counter High	DMA-RX	Section 8.2.4.18.34
0x00002F74	RXDLPBKPC	DMA Duplicated Good Rx LPBK Packet Counter	DMA-RX	Section 8.2.4.18.35
0x00002F78	RXDLPBKBCL	DMA Duplicated Good Rx LPBK Byte Counter Low	DMA-RX	Section 8.2.4.18.36
0x00002F7C	RXDLPBKBCH	DMA Duplicated Good Rx LPBK Byte Counter High	DMA-RX	Section 8.2.4.18.37
0x00004080	GPTC	Good Packets Transmitted Count	STAT	Section 8.2.4.18.38
0x00004090	GOTCL	Good Octets Transmitted Count Low	STAT	Section 8.2.4.18.39
0x00004094	GOTCH	Good Octets Transmitted Count High	STAT	Section 8.2.4.18.40
0x000087A0	TXDGPC	DMA Good Tx Packet Counter	DMA-TX	Section 8.2.4.18.41
0x000087A4	TXDGBCL	DMA Good Tx Byte Counter Low	DMA-TX	Section 8.2.4.18.42
0x000087A8	TXDGBCH	DMA Good Tx Byte Counter High	DMA-TX	Section 8.2.4.18.43
0x000040A4	RUC	Receive Undersize Count	STAT	Section 8.2.4.18.44
0x000040A8	RFC	Receive Fragment Count	STAT	Section 8.2.4.18.45
0x000040AC	ROC	Receive Oversize Count	STAT	Section 8.2.4.18.46
0x000040B0	RJC	Receive Jabber Count	STAT	Section 8.2.4.18.47
0x000040B4	MNGPRC	Management Packets Received Count	STAT	Section 8.2.4.18.48
0x000040B8	MNGPDC	Management Packets Dropped Count	STAT	Section 8.2.4.18.49
0x000040C0	TORL	Total Octets Received Low	STAT	Section 8.2.4.18.50
0x000040C4	TORH	Total Octets Received High	STAT	Section 8.2.4.18.51
0x000040D0	TPR	Total Packets Received	STAT	Section 8.2.4.18.52
0x000040D4	TPT	Total Packets Transmitted	STAT	Section 8.2.4.18.53
0x000040D8	PTC64	Packets Transmitted (64 Bytes) Count	STAT	Section 8.2.4.18.54

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x000040DC	PTC127	Packets Transmitted [65-127 Bytes] Count	STAT	Section 8.2.4.18.55
0x000040E0	PTC255	Packets Transmitted [128-255 Bytes] Count	STAT	Section 8.2.4.18.56
0x000040E4	PTC511	Packets Transmitted [256-511 Bytes] Count	STAT	Section 8.2.4.18.57
0x000040E8	PTC1023	Packets Transmitted [512-1023 Bytes] Count	STAT	Section 8.2.4.18.58
0x000040EC	PTC1522	Packets Transmitted [Greater than 1024 Bytes] Count	STAT	Section 8.2.4.18.59
0x000040F0	MPTC	Multicast Packets Transmitted Count	STAT	Section 8.2.4.18.60
0x000040F4	BPTC	Broadcast Packets Transmitted Count	STAT	Section 8.2.4.18.61
0x00004010	MSPDC	MAC short Packet Discard Count	STAT	Section 8.2.4.18.62
0x00004120	XEC	XSUM Error Count	STAT	Section 8.2.4.18.63
0x00002300 + 0x4*n, n=0...31	RQSMR[n]	Receive Queue Statistic Mapping Registers	DMA-RX	Section 8.2.4.18.64
0x00002F40	RXDSTATCTRL	Rx DMA Statistic Counter Control	DMA-RX	Section 8.2.4.18.65
0x00007300 + 0x4*n, n=0...7	TQSM[n]	Transmit Queue Statistic Mapping Registers	DMA-TX	Section 8.2.4.18.66
0x00008600 + 0x4*n, n=0...31	TQSM[n]	Transmit Queue Statistic Mapping Registers	DMA-TX	Section 8.2.4.18.67
0x00001030 + 0x40*n, n=0...15	QPRC[n]	Queue Packets Received Count	DMA-RX	Section 8.2.4.18.68
0x00001430 + 0x40*n, n=0...15	QPRDC[n]	Queue Packets Received Drop Count	DMA-RX	Section 8.2.4.18.69
0x00001034 + 0x40*n, n=0...15	QBRC_L[n]	Queue Bytes Received Count Low	DMA-RX	Section 8.2.4.18.70
0x00001038 + 0x40*n, n=0...15	QBRC_H[n]	Queue Bytes Received Count High	DMA-RX	Section 8.2.4.18.71
0x00008680 + 0x4*n, n=0...15	QPTC[n]	Queue Packets Transmitted Count	DMA-TX	Section 8.2.4.18.72
0x00008700 + 0x8*n, n=0...15	QBTC_L[n]	Queue Bytes Transmitted Count Low	DMA-TX	Section 8.2.4.18.73
0x00008704 + 0x8*n, n=0...15	QBTC_H[n]	Queue Bytes Transmitted Count High	DMA-TX	Section 8.2.4.18.74
0x00005118	FCCRC	FC CRC Error Count	RX-Filter	Section 8.2.4.18.75

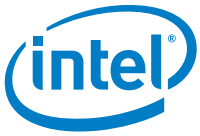


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x0000241C	FCOERPDC	FCoE Rx Packets Dropped Count	DMA-RX	Section 8.2.4.18.76
0x00002424	FCLAST	FC Last Error Count	DMA-RX	Section 8.2.4.18.77
0x00002428	FCOEPRC	FCoE Packets Received Count	DMA-RX	Section 8.2.4.18.78
0x0000242C	FCOEDWRC	FCoE DWord Received Count	DMA-RX	Section 8.2.4.18.79
0x00008784	FCOEPTC	FCoE Packets Transmitted Count	DMA-TX	Section 8.2.4.18.79
0x00008788	FCOEDWTC	FCoE DWord Transmitted Count	DMA-TX	Section 8.2.4.18.80
0x000041C0	B2OSPC	BMC2OS Packets Sent by BMC	STAT	Section 8.2.4.18.81
0x00002F90	B2OGPRC	BMC2OS Packets Received by Host	DMA-RX	Section 8.2.4.18.82
0x000041C4	O2BGPTC	TabOS2BMC packets received by BMC	STAT	Section 8.2.4.18.83
0x000087B0	O2BSPC	OS2BMC packets transmitted by host	DMA-TX	Section 8.2.4.18.84
0x00004180	BUPRC	BMC Total Unicast Packets Received	STAT	Section 8.2.4.18.85
0x00004184	BMPRC	BMC Total Multicast Packets Received	STAT	Section 8.2.4.18.86
0x00004188	BBPRC	BMC Total Broadcast Packets Received	STAT	Section 8.2.4.18.87
0x0000418C	BUPTC	BMC Total Unicast Packets Transmitted	STAT	Section 8.2.4.18.88
0x00004190	BMPTC	BMC Total Multicast Packets Transmitted	STAT	Section 8.2.4.18.89
0x00004194	BBPTC	BMC Total Broadcast Packets Transmitted	STAT	Section 8.2.4.18.90
0x00004198	BCRCERRS	BMC FCS Receive Errors	STAT	Section 8.2.4.18.91
0x0000419C	BXONRXC	BMC Pause XON Frames Received	STAT	Section 8.2.4.18.92
0x000041E0	BXOFFRXC	BMC Pause XOFF Frames Received	STAT	Section 8.2.4.18.93
0x000041E4	BXONTXC	BMC Pause XON Frames Transmitted	STAT	Section 8.2.4.18.94
0x000041E8	BXOFFTXC	BMC Pause XOFF Frames Transmitted	STAT	Section 8.2.4.18.95
Wake-Up Control Registers				
0x00005800	WUC	Wake Up Control Register	RX-Filter	Section 8.2.4.19.1

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00005808	WUFC	Wake Up Filter Control Register	RX-Filter	Section 8.2.4.19.2
0x00005810	WUS	Wake Up Status Register	RX-Filter	Section 8.2.4.19.3
0x00005838	IPAV	IP Address Valid	RX-Filter	Section 8.2.4.19.4
0x00005840 + 0x8*n, n=0...3	IP4AT[n]	IPv4 Address Table	RX-Filter	Section 8.2.4.19.5
0x00005880 + 0x4*n, n=0...3	IP6AT[n]	IPv6 Address Table	RX-Filter	Section 8.2.4.19.6
0x00009000 + 0x10*n, n=0...14	FHFT_FILTER0_DW_L[n]	Filter 0 Dword Lower	RX-Filter	Section 8.2.4.19.7
0x00009004 + 0x10*n, n=0...14	FHFT_FILTER0_DW_U[n]	Filter 0 Dword Upper	RX-Filter	Section 8.2.4.19.8
0x00009008 + 0x10*n, n=0...14	FHFT_FILTER0_MASK[n]	Filter 0 Mask	RX-Filter	Section 8.2.4.19.9
0x0000900C + 0x10*n, n=0...14	FHFT_FILTER0_RESERVED[n]	Filter 0 Reserved	RX-Filter	Section 8.2.4.19.10
0x000090F0	FHFT_FILTER0_DW30	Filter 0 DW30	RX-Filter	Section 8.2.4.19.11
0x000090F4	FHFT_FILTER0_DW31	Filter 0 DW31	RX-Filter	Section 8.2.4.19.12
0x000090F8	FHFT_FILTER0_MASK_120_127	Filter 0 Mask[120:127]	RX-Filter	Section 8.2.4.19.13
0x000090FC	FHFT_FILTER0_LENGTH	Filter 0 Length	RX-Filter	Section 8.2.4.19.14
0x00009100 + 0x10*n, n=0...14	FHFT_FILTER1_DW_L[n]	Filter 1 Dword Lower	RX-Filter	Section 8.2.4.19.15
0x00009104 + 0x10*n, n=0...14	FHFT_FILTER1_DW_U[n]	Filter 1 Dword Upper	RX-Filter	Section 8.2.4.19.16
0x00009108 + 0x10*n, n=0...14	FHFT_FILTER1_MASK[n]	Filter 1 Mask	RX-Filter	Section 8.2.4.19.17
0x0000910C + 0x10*n, n=0...14	FHFT_FILTER1_RESERVED[n]	Filter 1 Reserved	RX-Filter	Section 8.2.4.19.18
0x000091F0	FHFT_FILTER1_DW30	Filter 1 Dword 30	RX-Filter	Section 8.2.4.19.19
0x000091F4	FHFT_FILTER1_DW31	Filter 1 Dword 31	RX-Filter	Section 8.2.4.19.20
0x000091F8	FHFT_FILTER1_MASK_120_127	Filter 1 Mask[120:127]	RX-Filter	Section 8.2.4.19.21

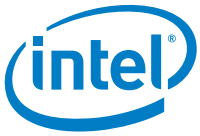


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x000091FC	FHFT_FILTER1_LENGTH	Filter 1 Length	RX-Filter	Section 8.2.4.19.22
0x00009200 + 0x10*n, n=0...14	FHFT_FILTER2_DW_L[n]	Filter 2 Dword Lower	RX-Filter	Section 8.2.4.19.23
0x00009204 + 0x10*n, n=0...14	FHFT_FILTER2_DW_U[n]	Filter 2 Dword Upper	RX-Filter	Section 8.2.4.19.24
0x00009208 + 0x10*n, n=0...14	FHFT_FILTER2_MASK[n]	Filter 2 Mask	RX-Filter	Section 8.2.4.19.25
0x0000920C + 0x10*n, n=0...14	FHFT_FILTER2_RESERVED[n]	Filter 2 Reserved	RX-Filter	Section 8.2.4.19.26
0x000092F0	FHFT_FILTER2_DW30	Filter 2 Dword 30	RX-Filter	Section 8.2.4.19.27
0x000092F4	FHFT_FILTER2_DW31	Filter 2 Dword 31	RX-Filter	Section 8.2.4.19.28
0x000092F8	FHFT_FILTER2_MASK_120_127	Filter 2 Mask[120:127]	RX-Filter	Section 8.2.4.19.29
0x000092FC	FHFT_FILTER2_LENGTH	Filter 2 Length	RX-Filter	Section 8.2.4.19.30
0x00009300 + 0x10*n, n=0...14	FHFT_FILTER3_DW_L[n]	Filter 3 Dword Lower	RX-Filter	Section 8.2.4.19.31
0x00009304 + 0x10*n, n=0...14	FHFT_FILTER3_DW_U[n]	Filter 3 Dword Upper	RX-Filter	Section 8.2.4.19.32
0x00009308 + 0x10*n, n=0...14	FHFT_FILTER3_MASK[n]	Filter 3 Mask	RX-Filter	Section 8.2.4.19.33
0x0000930C + 0x10*n, n=0...14	FHFT_FILTER3_RESERVED[n]	Filter 3 Reserved	RX-Filter	Section 8.2.4.19.34
0x000093F0	FHFT_FILTER3_DW30	Filter 3 Dword 30	RX-Filter	Section 8.2.4.19.35
0x000093F4	FHFT_FILTER3_DW31	Filter 3 Dword 31	RX-Filter	Section 8.2.4.19.36
0x000093F8	FHFT_FILTER3_MASK_120_127	Filter 3 Mask[120:127]	RX-Filter	Section 8.2.4.19.37
0x000093FC	FHFT_FILTER3_LENGTH	Filter 3 Length	RX-Filter	Section 8.2.4.19.38
0x00009800 + 0x10*n, n=0...14	FHFT_FILTER4_DW_L[n]	Filter 4 Dword Lower	RX-Filter	Section 8.2.4.19.39
0x00009804 + 0x10*n, n=0...14	FHFT_FILTER4_DW_U[n]	Filter 4 Dword Upper	RX-Filter	Section 8.2.4.19.40

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00009808 + 0x10*n, n=0...14	FHFT_FILTER4_MASK[n]	Filter 4 Mask	RX-Filter	Section 8.2.4.19.41
0x0000980C + 0x10*n, n=0...14	FHFT_FILTER4_RESERVED[n]	Filter 4 Reserved	RX-Filter	Section 8.2.4.19.42
0x000098F0	FHFT_FILTER4_DW30	Filter 4 DW30	RX-Filter	Section 8.2.4.19.43
0x000098F4	FHFT_FILTER4_DW31	Filter 4 DW31	RX-Filter	Section 8.2.4.19.44
0x000098F8	FHFT_FILTER4_MASK_120_127	Filter 4 Mask[120:127]	RX-Filter	Section 8.2.4.19.45
0x000098FC	FHFT_FILTER4_LENGTH	Filter 4 Length	RX-Filter	Section 8.2.4.19.46
0x00009900 + 0x10*n, n=0...14	FHFT_FILTER5_DW_L[n]	Filter 5 Dword Lower	RX-Filter	Section 8.2.4.19.47
0x00009904 + 0x10*n, n=0...14	FHFT_FILTER5_DW_U[n]	Filter 5 Dword Upper	RX-Filter	Section 8.2.4.19.48
0x00009908 + 0x10*n, n=0...14	FHFT_FILTER5_MASK[n]	Filter 5 Mask	RX-Filter	Section 8.2.4.19.49
0x0000990C + 0x10*n, n=0...14	FHFT_FILTER5_RESERVED[n]	Filter 5 Reserved	RX-Filter	Section 8.2.4.19.50
0x000099F0	FHFT_FILTER5_DW30	Filter 5 DW30	RX-Filter	Section 8.2.4.19.51
0x000099F4	FHFT_FILTER5_DW31	Filter 5 DW31	RX-Filter	Section 8.2.4.19.52
0x000099F8	FHFT_FILTER5_MASK_120_127	Filter 5 Mask[120:127]	RX-Filter	Section 8.2.4.19.53
0x000099FC	FHFT_FILTER5_LENGTH	Filter 5 Length	RX-Filter	Section 8.2.4.19.54
Management Filters Registers				
0x00005010 + 0x4*n, n=0...7	MAVTV[n]	Management VLAN TAG Value	RX-Filter	Section 8.2.4.20.1
0x00005030 + 0x4*n, n=0...7	MFUTP[n]	Management Flex UDP/TCP Ports	RX-Filter	Section 8.2.4.20.2
0x00005190 + 0x4*n, n=0...3	METF[n]	Management Ethernet Type Filters	RX-Filter	Section 8.2.4.20.3
0x00005820	MANC	Management Control Register	RX-Filter	Section 8.2.4.20.4
0x00005824	MFVAL	Manageability Filters Valid	RX-Filter	Section 8.2.4.20.5

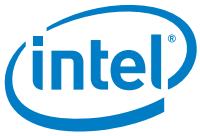


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00005860	MANC2H	Management Control To Host Register	RX-Filter	Section 8.2.4.20.6
0x00005890 + 0x4*n, n=0...7	MDEF[n]	Manageability Decision Filters	RX-Filter	Section 8.2.4.20.7
0x00005160 + 0x4*n, n=0...7	MDEF_EXT[n]	Manageability Decision Filters Ext	RX-Filter	Section 8.2.4.20.8
0x00005050 + 0x4*n, n=0...3	BMCIP[n]	BMC IP address Register	RX-Filter	Section 8.2.4.20.9
0x00005060	BMCIPVAL	BMC IP Valid Register	RX-Filter	Section 8.2.4.20.10
0x000058B0 + 0x4*n + 0x10*m, n=0...3, m=0...3	MIPAF[n,m]	Manageability IP Address Filter	RX-Filter	Section 8.2.4.20.11
0x00005910 + 0x8*n, n=0...3	MMAL[n]	Manageability Ethernet MAC Address Low	RX-Filter	Section 8.2.4.20.12
0x00005914 + 0x8*n, n=0...3	MMAH[n]	Manageability Ethernet MAC Address High	RX-Filter	Section 8.2.4.20.13
0x00009400 + 0x10*n, n=0...14	FTFT_FILTER0_DW_L[n]	FTFT Filter 0 Dword Lower	RX-Filter	Section 8.2.4.20.14
0x00009404 + 0x10*n, n=0...14	FTFT_FILTER0_DW_U[n]	FTFT Filter 0 Dword Upper	RX-Filter	Section 8.2.4.20.15
0x00009408 + 0x10*n, n=0...14	FTFT_FILTER0_MASK[n]	FTFT Filter 0 Mask	RX-Filter	Section 8.2.4.20.16
0x0000940C + 0x10*n, n=0...14	FTFT_FILTER0_RESERVED[n]	FTFT Filter 0 Reserved	RX-Filter	Section 8.2.4.20.17
0x000094F0	FTFT_FILTER0_DW30	FTFT Filter 0 DW30	RX-Filter	Section 8.2.4.20.18
0x000094F4	FTFT_FILTER0_DW31	FTFT Filter 0 DW31	RX-Filter	Section 8.2.4.20.19
0x000094F8	FTFT_FILTER0_MASK_120_127	FTFT Filter 0 Mask[120:127]	RX-Filter	Section 8.2.4.20.20
0x000094FC	FTFT_FILTER0_LENGTH	FTFT Filter 0 Length	RX-Filter	Section 8.2.4.20.21
0x00009500 + 0x10*n, n=0...14	FTFT_FILTER1_DW_L[n]	FTFT Filter 1 Dword Lower	RX-Filter	Section 8.2.4.20.22
0x00009504 + 0x10*n, n=0...14	FTFT_FILTER1_DW_U[n]	FTFT Filter 1 Dword Upper	RX-Filter	Section 8.2.4.20.23
0x00009508 + 0x10*n, n=0...14	FTFT_FILTER1_MASK[n]	FTFT Filter 1 Mask	RX-Filter	Section 8.2.4.20.24
0x0000950C + 0x10*n, n=0...14	FTFT_FILTER1_RESERVED[n]	FTFT Filter 1 Reserved	RX-Filter	Section 8.2.4.20.25

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x000095F0	FTFT_FILTER1_DW30	FTFT Filter 1 DW30	RX-Filter	Section 8.2.4.20.26
0x000095F4	FTFT_FILTER1_DW31	FTFT Filter 1 DW31	RX-Filter	Section 8.2.4.20.27
0x000095F8	FTFT_FILTER1_MASK_120_127	FTFT Filter 1 Mask[120:127]	RX-Filter	Section 8.2.4.20.28
0x000095FC	FTFT_FILTER1_LENGTH	FTFT Filter 1 Length	RX-Filter	Section 8.2.4.20.29
0x00009600 + 0x10*n, n=0...14	FTFT_FILTER2_DW_L[n]	FTFT Filter 2 Dword Lower	RX-Filter	Section 8.2.4.20.30
0x00009604 + 0x10*n, n=0...14	FTFT_FILTER2_DW_U[n]	FTFT Filter 2 Dword Upper	RX-Filter	Section 8.2.4.20.31
0x00009608 + 0x10*n, n=0...14	FTFT_FILTER2_MASK[n]	FTFT Filter 2 Mask	RX-Filter	Section 8.2.4.20.32
0x0000960C + 0x10*n, n=0...14	FTFT_FILTER2_RESERVED[n]	FTFT Filter 2 Reserved	RX-Filter	Section 8.2.4.20.33
0x000096F0	FTFT_FILTER2_DW30	FTFT Filter 2 DW30	RX-Filter	Section 8.2.4.20.34
0x000096F4	FTFT_FILTER2_DW31	FTFT Filter 2 DW31	RX-Filter	Section 8.2.4.20.35
0x000096F8	FTFT_FILTER2_MASK_120_127	FTFT Filter 2 Mask[120:127]	RX-Filter	Section 8.2.4.20.36
0x000096FC	FTFT_FILTER2_LENGTH	FTFT Filter 2 Length	RX-Filter	Section 8.2.4.20.37
0x00009700 + 0x10*n, n=0...14	FTFT_FILTER3_DW_L[n]	FTFT Filter 3 Dword Lower	RX-Filter	Section 8.2.4.20.38
0x00009704 + 0x10*n, n=0...14	FTFT_FILTER3_DW_U[n]	FTFT Filter 3 Dword Upper	RX-Filter	Section 8.2.4.20.39
0x00009708 + 0x10*n, n=0...14	FTFT_FILTER3_MASK[n]	FTFT Filter 3 Mask	RX-Filter	Section 8.2.4.20.40
0x0000970C + 0x10*n, n=0...14	FTFT_FILTER3_RESERVED[n]	FTFT Filter 3 Reserved	RX-Filter	Section 8.2.4.20.41
0x000097F0	FTFT_FILTER3_DW30	FTFT Filter 3 DW30	RX-Filter	Section 8.2.4.20.42
0x000097F4	FTFT_FILTER3_DW31	FTFT Filter 3 DW31	RX-Filter	Section 8.2.4.20.43
0x000097F8	FTFT_FILTER3_MASK_120_127	FTFT Filter 3 Mask[120:127]	RX-Filter	Section 8.2.4.20.44

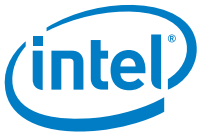


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x000097FC	FTFT_FILTER3_LENGTH	FTFT Filter 3 Length	RX-Filter	Section 8.2.4.20.45
Time Sync (IEEE 1588) Registers				
0x00005188	TSYNCRXCTL	Rx Time Sync Control Register	RX-Filter	Section 8.2.4.21.1
0x000051E8	RXSTMPL	Rx Timestamp Low	RX-Filter	Section 8.2.4.21.2
0x000051A4	RXSTMPH	Rx Timestamp High	RX-Filter	Section 8.2.4.21.3
0x000051A0	RXSATRL	Rx Timestamp Attributes Low	RX-Filter	Section 8.2.4.21.4
0x000051A8	RXSATRH	Rx Timestamp Attributes High	RX-Filter	Section 8.2.4.21.5
0x00005120	RXMTRL	Rx Message Type Register Low	RX-Filter	Section 8.2.4.21.6
0x00008C00	TSYNCTXCTL	Tx Time Sync Control Register	SEC-TX	Section 8.2.4.21.7
0x00008C04	TXSTMPL	Tx Timestamp Value Low	SEC-TX	Section 8.2.4.21.8
0x00008C08	TXSTMPH	Tx Timestamp Value High	SEC-TX	Section 8.2.4.21.9
0x00008C0C	SYSTIMEL	System Time Register Low	SEC-TX	Section 8.2.4.21.10
0x00008C10	SYSTIMEH	System Time Register High	SEC-TX	Section 8.2.4.21.11
0x00008C14	TIMEINCA	Increment Attributes Register	SEC-TX	Section 8.2.4.21.12
0x00008C18	TIMEADJL	Time Adjustment Offset Register Low	SEC-TX	Section 8.2.4.21.13
0x00008C1C	TIMEADJH	Time Adjustment Offset Register High	SEC-TX	Section 8.2.4.21.14
0x00008C20	TSAUXC	TimeSync Auxiliary Control Register	SEC-TX	Section 8.2.4.21.15
0x00008C24	TRGTTIMEL0	Target Time Register 0 Low	SEC-TX	Section 8.2.4.21.16
0x00008C28	TRGTTIMEH0	Target Time Register 0 High	SEC-TX	Section 8.2.4.21.17
0x00008C2C	TRGTTIMEL1	Target Time Register 1 Low	SEC-TX	Section 8.2.4.21.18
0x00008C30	TRGTTIMEH1	Target Time Register 1 High	SEC-TX	Section 8.2.4.21.19
0x00008C34	CLKTIMEL	Clock Phase Time Low Register	SEC-TX	Section 8.2.4.21.20
0x00008C38	CLKTIMEH	Clock Phase Time High Register	SEC-TX	Section 8.2.4.21.21
0x00008C3C	AUXSTMPL0	Auxiliary Time Stamp 0 Register Low	SEC-TX	Section 8.2.4.21.22
0x00008C40	AUXSTMPH0	Auxiliary Time Stamp 0 Register High	SEC-TX	Section 8.2.4.21.23

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00008C44	AUXSTMPL1	Auxiliary Time Stamp 1 Register Low	SEC-TX	Section 8.2.4.21.24
0x00008C48	AUXSTMPH1	Auxiliary Time Stamp 1 Register High	SEC-TX	Section 8.2.4.21.25
Virtualization PF Registers				
0x000051B0	PFVTCTL	PF Virtual Control Register	RX-Filter	Section 8.2.4.22.1
0x00004B00 + 0x4*n, n=0...63	PFMAILBOX[n]	PF Mailbox	Target	Section 8.2.4.22.2
0x00000710 + 0x4*n, n=0...3	PFMBICR[n]	PF Mailbox Interrupt Causes Register	Target	Section 8.2.4.22.3
0x00000720 + 0x4*n, n=0...1	PFMBIMR[n]	PF Mailbox Interrupt Mask Register	Target	Section 8.2.4.22.4
0x00000700 + 0x4*n, n=0...1	PFVFLREC[n]	PF VFLR Events Clear	Target	Section 8.2.4.22.5
0x000051E0 + 0x4*n, n=0...1	PFVFRE[n]	PF VF Receive Enable	RX-Filter	Section 8.2.4.22.6
0x00008110 + 0x4*n, n=0...1	PFVFTE[n]	PF VF Transmit Enable	DMA-TX	Section 8.2.4.22.7
0x00008790	PFVMECM0	PF VM 0:31 Error Count Mask	DMA-TX	Section 8.2.4.22.8
0x00008794	PFVMECM1	PF VM 32:63 Error Count Mask	DMA-TX	Section 8.2.4.22.9
0x00002F04	PFQDE	PF Queue Drop Enable Register	DMA-RX	Section 8.2.4.22.10
0x00005180 + 0x4*n, n=0...1	PFVMTXSW[n]	PF VM Tx Switch Loopback Enable	RX-Filter	Section 8.2.4.22.11
0x00008200 + 0x4*n, n=0...7	PFVFSPOOF[n]	PFVF Anti Spoof Control	DMA-TX	Section 8.2.4.22.12
0x00008220	PFDTXGSWC	PF DMA Tx General Switch Control	DMA-TX	Section 8.2.4.22.13
0x00008000 + 0x4*n, n=0...63	PFVMVIR[n]	PF VM VLAN Insert Register	DMA-TX	Section 8.2.4.22.14
0x0000F000 + 0x4*n, n=0...63	PFVML2FLT[n]	PF VM L2Control Register	RX-Filter	Section 8.2.4.22.15
0x0000F100 + 0x4*n, n=0...63	PFVLVF[n]	PF VM VLAN Pool Filter	RX-Filter	Section 8.2.4.22.16
0x0000F200 + 0x4*n, n=0...127	PFVLVFB[n]	PF VM VLAN Pool Filter Bitmap	RX-Filter	Section 8.2.4.22.17
0x0000F400 + 0x4*n, n=0...127	PFUTA[n]	PF Unicast Table Array	RX-Filter	Section 8.2.4.22.18

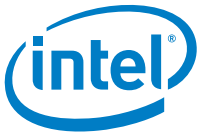


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x0000F600 + 0x4*n, n=0...3	PFMRCTL[n]	PF Mirror Rule Control	RX-Filter	Section 8.2.4.22.19
0x0000F610 + 0x4*n, n=0...7	PFMRVLAN[n]	PF Mirror Rule VLAN	RX-Filter	Section 8.2.4.22.20
0x0000F630 + 0x4*n, n=0...7	PFMRVM[n]	PF Mirror Rule Pool	RX-Filter	Section 8.2.4.22.21
Diagnostic Registers				
0x0000C600	TXBUFCTRL	TX Buffer Access Control	DBU-TX	Section 8.2.4.23.1
0x0000C610 + 0x4*n, n=0...3	TXBUFDATA[n]	TX Buffer Data	DBU-TX	Section 8.2.4.23.2
0x00000740	MEM_CSR	Mailbox Memory CSR	Target	Section 8.2.4.23.3
0x0000CF78	TXRAMCTL	Transmit Memories Control	DBU-TX	Section 8.2.4.23.4
0x00003600	RXBUFCTRL	RX Buffer Access Control	DBU-RX	Section 8.2.4.23.5
0x00003100 + 0x4*n, n=0...7	RXWRPTR[n]	Rx Write Buffer Pointers	DBU-RX	Section 8.2.4.23.6
0x00003120 + 0x4*n, n=0...7	RXUSED[n]	Rx Buffer Used Space	DBU-RX	Section 8.2.4.23.7
0x00003140 + 0x4*n, n=0...7	RXRDPTR[n]	Rx Read Buffer Pointers	DBU-RX	Section 8.2.4.23.8
0x00003160 + 0x4*n, n=0...7	RXRDWRPTR[n]	Rx Read Write Offset Pointers	DBU-RX	Section 8.2.4.23.9
0x00003F80	TXSWERR	Tx Switch Buffer Error	DBU-RX	Section 8.2.4.23.10
0x0000C100 + 0x4*n, n=0...7	TXWRPTR[n]	Tx Write Buffer Pointers	DBU-TX	Section 8.2.4.23.11
0x0000C140 + 0x4*n, n=0...7	TXRDPTR[n]	Tx Read Buffer Pointers	DBU-TX	Section 8.2.4.23.12
0x0000C160 + 0x4*n, n=0...7	TXRDWRPTR[n]	Tx Read Write Offset Pointers	DBU-TX	Section 8.2.4.23.13
0x00003610 + 0x4*n, n=0...7	RXBUFDATA[n]	Rx Buffer DATA	DBU-RX	Section 8.2.4.23.14
0x00003F78	RXRAMCTL	Receive Memories Control	DBU-RX	Section 8.2.4.23.15
0x00002F30	RDMAM	Receive DMA Memory Access Control Register	DMA-RX	Section 8.2.4.23.16
0x00002F34	RDMAD	Receive DMA Memory Data Register	DMA-RX	Section 8.2.4.23.17
0x00002F14	RDRXSTAT	Receive DMA Status Register	DMA-RX	Section 8.2.4.23.18

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00002F8C	DRECCCTL2	Receive DMA ECC Control 2 Register	DMA-RX	Section 8.2.4.23.19
0x000082FC	TDHMPN	Tx Descriptor Handler Memory Page Number	DMA-TX	Section 8.2.4.23.20
0x000082B0	TIC_DW0	Tx Descriptor Handler Memory Read Data 0	DMA-TX	Section 8.2.4.23.21
0x000082B4	TIC_DW1	Tx Descriptor Handler Memory Read Data 1	DMA-TX	Section 8.2.4.23.22
0x000082B8	TIC_DW2	Tx Descriptor Handler Memory Read Data 2	DMA-TX	Section 8.2.4.23.23
0x000082BC	TIC_DW3	Tx Descriptor Handler Memory Read Data 3	DMA-TX	Section 8.2.4.23.24
0x000082CC	TXDESCIC	Tx Descriptor Icache Indirect Access Register	DMA-TX	Section 8.2.4.23.25
0x000082A4	TXIDLE	Tx DMA Performance IDLE Count	DMA-TX	Section 8.2.4.23.26
0x000082A0	TXDESC	Tx DMA Performance Descriptor Fetch Count	DMA-TX	Section 8.2.4.23.27
0x00004970 + 0x4*n, n=0...3	RTDTPBSPC[n]	RT Tx Descriptor PB Space Counter	DMA-TX	Section 8.2.4.23.28
0x00010214	FLEEPECC	FLEEP ECC	FLEEP	Section 8.2.4.23.29
0x00011090	PCIE_DIAG1	PCIe Diagnostic 1	PCIe	Section 8.1.4.32.32
0x00011094	PCIE_DIAG2	PCIe Diagnostic 2	PCIe	Section 8.2.4.23.31
0x00011098	PCIE_DIAG3	PCIe Diagnostic 3	PCIe	Section 8.2.4.23.32
0x0001109C	PCIE_DIAG4	PCIe Diagnostic 4	PCIe	Section 8.2.4.23.33
0x000110A0	PCIE_DIAG5	PCIe Diagnostic 5	PCIe	Section 8.2.4.23.34
0x000110A4	PCIE_DIAG6	PCIe Diagnostic 6	PCIe	Section 8.2.4.23.35
0x000110A8	PCIE_DIAG7	PCIe Diagnostic 7	PCIe	Section 8.2.4.23.36
0x000110AC	PCIE_DIAG8	PCIe Diagnostic 8	PCIe	Section 8.2.4.23.37
0x000110E4	PCIEECCSF	PCIe ECC FIX Status	PCIe	Section 8.2.4.23.38
0x000050A4	RFVAL	Receive Filter Validation Register	RX-Filter	Section 8.2.4.23.39
0x00011100	PCIEECCCTL0	PCIe ECC Control 0 Register	PCIe	Section 8.2.4.23.40
0x00011104	PCIEECCCTL1	PCIe ECC Control 1 Register	PCIe	Section 8.2.4.23.41
0x000110E0	ECC_STATUS	PCIe ECC Status Register	PCIe	Section 8.2.4.23.42

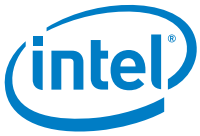


Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x000110F0	PCIEMISC	PCIe Misc. Register	PCIe	Section 8.2.4.23.43
0x00011108	PF_PEND	PCIe PF Pending request Register	PCIe	Section 8.2.4.23.44
0x00011110	VFH_PEND	PCIe VF Pending request Register High	PCIe	Section 8.2.4.23.45
0x0001110C	VFL_PEND	PCIe VF Pending request Register Low	PCIe	Section 8.2.4.23.46
0x0000CF70	TXDBUECC	Tx Memories ECC Register	DBU-TX	Section 8.2.4.23.47
0x00008814	SECTXSAECC	SEC Tx SA ECC Control	SEC-TX	Section 8.2.4.23.48
0x00008818	SECTXBUFFECC	SEC Tx Buffer ECC Control	SEC-TX	Section 8.2.4.23.49
0x0000881C	SECTXAESECC	SEC Tx AES ECC Control	SEC-TX	Section 8.2.4.23.50
0x00008D0C	SECRXSAECC	SEC Rx SA ECC Control	SEC-TX	Section 8.2.4.23.51
0x00008D10	SECRXAESECC	SEC Rx AES ECC Control	SEC-TX	Section 8.2.4.23.52
0x00016000	PROBE_SEL	Probe Select Register	DFTPRD	Section 8.2.4.23.53
0x00016038	PROBE_OUT	Probe Out Register	DFTPRD	Section 8.2.4.23.54
0x00003F70	RXDBUECC	Receive DBU ECC	DBU-RX	Section 8.2.4.23.55
0x00002F0C	DRECCSTAT	DMA Rx ECC Status	DMA-RX	Section 8.2.4.23.56
0x00002F18	RDWBOFST	Receive DMA WB offset	DMA-RX	Section 8.2.4.23.57
0x00002F08	DRECCCTL	DMA Rx ECC Control	DMA-RX	Section 8.2.4.23.58
0x000051B8	RXFECCERR0	Rx Filter ECC Err Insertion 0	RX-Filter	Section 8.2.4.23.59
0x000051C0	RXFECCERR1	Rx Filter ECC Err Insertion 1	RX-Filter	Section 8.2.4.23.60
0x000051C8	RXFECCSTATC	Rx Filter ECC Correctable Status	RX-Filter	Section 8.2.4.23.61
0x000051D0	RXFECCSTATUC	Rx Filter ECC Uncorrectable Status	RX-Filter	Section 8.2.4.23.62
0x00004990	RTTDTECC	Tx DMA QCN VM-ARB ECC	DMA-TX	Section 8.2.4.23.63
0x0000810C	RTTDPECC	Tx DMA descriptor processor ECC	DMA-TX	Section 8.2.4.23.64
0x00004A9C	TXDMAECC	Tx DMA ECC	DMA-TX	Section 8.2.4.23.65
0x000087AC	TXDMASTATECC	Tx DMA Statistics ECC	DMA-TX	Section 8.2.4.23.66
0x00004AA0	TXDMAGENECC	Tx DMA General ECC	DMA-TX	Section 8.2.4.23.67
0x00008280	TDHTP_0	Tx Descriptor Handler Task Priority 0	DMA-TX	Section 8.2.4.23.68

Table 8-2 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00008284	TDHTP_1	Tx Descriptor Handler Task Priority 1	DMA-TX	Section 8.2.4.23.69
0x00008288	TDHTP_2	Tx Descriptor Handler Task Priority 2	DMA-TX	Section 8.2.4.23.70
0x0000828C	TDHTP_3	Tx Descriptor Handler Task Priority 3	DMA-TX	Section 8.2.4.23.71
0x00008290	TDHTE	Tx Descriptor Handler Task Enable	DMA-TX	Section 8.2.4.23.72
0x00008294	TDH_STATES	Tx Descriptor Handler FSM States	DMA-TX	Section 8.2.4.23.73
0x00008108	LVMMC	Last VM Misbehavior Cause	DMA-TX	Section 8.2.4.23.74
0x0000C760	TXMEMWRAP	Tx Packet Buffer Flush Detect	DBU-TX	Section 8.2.4.23.75
0x0000CF74	TXDBUEST	Transmit DBU ECC Counters	DBU-TX	Section 8.2.4.23.76
0x0000C120 + 0x4*n, n=0...7	TXPBUSED[n]	Transmit Packet Buffer Used Space	DBU-TX	Section 8.2.4.23.77
0x00011088	CIAA	Config Indirect Access address	PCIe	Section 8.2.4.23.78
0x0001108C	CIAD	Config Indirect Access Data	PCIe	Section 8.2.4.23.78
0x00016034	TCK_FILTER_RATE	JTAG Clock Filter Register	DFTPRD	Section 8.2.4.23.78

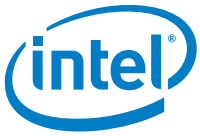
8.2.4 Detailed Register Description - PF BAR0

8.2.4.1 General Control Registers

8.2.4.1.1 Device Control Register - CTRL (0x00000000)

CTRL is also mapped to address 0x00004 to maintain compatibility with predecessors.

Note: Additional address(es): 0x00004 LRST and RST can be used to globally reset the entire X540 10GBase-T Controller. This register is provided primarily as a last-ditch 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 X540 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 X540 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.



Field	Bit(s)	Init.	Access Type	Description
RESERVED	1:0	0x0	RSV	Reserved.Write as 0b for future compatibility.
PCIE_MASTER_DISABLE	2	0b	RW	When set, the X540 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 doing 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.
LRST	3	0b	RW	Link Reset. This bit performs a reset of the MAC, PHY, and the entire X540 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 X540, 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.

8.2.4.1.2 Device Control Register - CTRL (0x00000004; RW)

CTRL is also mapped to address 0x00004 to maintain compatibility with predecessors.

Fields definitions are the same as defined on Section 8.2.4.1.1.

8.2.4.1.3 Device Status Register - STATUS (0x00000008)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	1:0	0x0	RSV	Reserved.
LAN_ID	3:2	0x0	RO	LAN ID. Provides software a mechanism to determine the device LAN identifier for this MAC. Read as: [0,0] LAN 0, [0,1] LAN 1.
PFVFSTAT	6:4	0x0	RO	PFVF Status Indication. Read Write field that can be used by the PF driver to pass any status indication to the VFs.
LINKUP	7	0b	RW	Linkup Status Indication. This bit is useful for IOV mode. The PF software driver sets it according to LINKS register and PHY state. It is reflected in the VFSTATUS register indicating LinkUp to the VF drivers.
RESERVED	9:8	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Access Type	Description
NUM_VFS	17:10	0x0	RO	The Num VFs field, reflects the value of the Num VFs in the IOV capability structure. Note that bit 17 is always 0b.
IOV_ACTIVE	18	0b	RO	The IOV Active bit, reflects the value of the VF Enable (VFE) bit in the IOV control/status register.
PCIE_MASTER_ENABLE_STATUS	19	1b	RO	This is a status bit of the appropriate CTRL.PCIe Master Disable bit.1b = Associated LAN function can issue master requests.0b = Associated LAN function does not issue any master request and all previously issued requests are complete.
RESERVED	31:20	0x0	RSV	Reserved. Reads as 0b.

8.2.4.1.4 Extended Device Control Register - CTRL_EXT (0x00000018)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	13:0	0x00	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 regs is set.
RESERVED	15	0b	RSV	Reserved.
NS_DIS	16	0b	RW	No Snoop Disable. When set to 1b, the X540 does not set the no snoop attribute in any PCIe packet, independent of PCIe configuration and the setting of individual no snoop enable bits. When set to 0b, behavior of no snoop is determined by PCIe configuration and the setting of individual no snoop enable bits.
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 DCA_RXCTRL[n] and DCA_TXCTRL[n] registers.
RESERVED	25:18	0x0	RSV	Reserved.
EXTENDED_VLAN	26	0b	RW	Extended VLAN. When set, all incoming Rx packets are expected to have at least one VLAN with the Ether type as defined in EXVET register. The packets can have an inner- VLAN that should be used for all filtering purposes. All Tx packets are expected to have at least one VLAN added to them by the host. In the case of an additional VLAN request (VLE) the inner-VLAN is added by hardware after the outer-VLAN is added by the host. This bit should only be reset by a PCIe reset and should only be changed while Tx and Rx processes are stopped.Exception to this rule are MAC control packets such as flow control, 802.1x, LACP, etc. that never carry a VLAN tag of any type.
RESERVED	27	0b	RSV	Reserved.

Field	Bit(s)	Init.	Access Type	Description
DRV_LOAD	28	0b	RW	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 driver successfully loaded to it.
RESERVED	31:29	0x0	RSV	Reserved.

8.2.4.1.5 Extended SDP Control - ESDP (0x00000020)

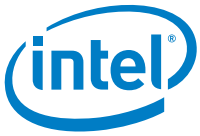
This register is initialized only at LAN Power Good preserving the SDP states across software and PCIe resets. Some specific I/O pins are initialized in other resets in native mode as expected for the specific behavior and described explicitly as follows.

Field	Bit(s)	Init.	Access Type	Description
SDP0_DATA	0	0b	RW	SDP0 Data Value. Used to read (write) a value of the software-controlled I/O pin SDP0. If SDP0 is configured as an output (SDP0_IODIR = 1b), this bit controls the value driven on the pin. If SDP0 is configured as an input, all reads return the current value of the pin. See note 1 and 2 that follows this table.
SDP1_DATA	1	0b	RW	SDP1 Data Value. Used to read (write) a value of the software-controlled I/O pin SDP1. If SDP1 is configured as an output (SDP1_IODIR = 1b), this bit controls the value driven on the pin. If SDP1 is configured as an input, all reads return the current value of the pin. See note 1 and 3 that follows this table.
SDP2_DATA	2	0b	RW	SDP2 Data Value. 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. See note 1 that follows this table.
SDP3_DATA	3	0b	RW	SDP3 Data Value. 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. See note 1 that follows this table.
RESERVED	7:4	0x0	RSV	Reserved.
SDP0_IODIR	8	0b	RW	SDP0 Pin Directionality. Controls whether or not software-controlled pin SDP0 is configured as an input or output. 0b = Input. 1b = Output. See note 1 and 2 that follows this table.
SDP1_IODIR	9	0b	RW	SDP1 Pin Directionality. Controls whether or not software-controlled pin SDP1 is configured as an input or output. 0b = Input. 1b = Output. See note 1 and 3 that follows this table.

Field	Bit(s)	Init.	Access Type	Description
SDP2_IODIR	10	0b	RW	SDP2 Pin Directionality. Controls whether or not software-controlled pin SDP2 is configured as an input or output. 0b = Input. 1b = Output. See note 1 that follows this table.
SDP3_IODIR	11	0b	RW	SDP3 Pin Directionality. Controls whether or not software-controlled pin SDP3 is configured as an input or output. 0b = Input. 1b = Output. See note 1 that follows this table.
RESERVED	15:12	0x0	RSV	Reserved.
SDP0_NATIVE	16	0b	RW	SDP0 Operating Mode. 0b = Generic software controlled I/O by SDP0_DATA and SDP0_IODIR. 1b = Native mode operation (connected to hardware function) is target time 0 / clock out functionality. Refer to Time Sync Related Auxiliary Elements. In this mode SDP0_IODIR should be configured as output. See note 2 that follows this table.
SDP1_NATIVE	17	0b	RW	SDP1 Operating Mode. 0b = Generic software controlled I/O by SDP1_DATA and SDP1_IODIR. 1b = Native mode operation (connected to hardware function) is target time 1 / SDP1_TS functionality according to SDP1_Function bit. In this mode SDP1_IODIR should be configured as output. See note 1 and 3 that follows this table.
SDP2_NATIVE	18	0b	RW	SDP2 Operating Mode. 0b = Generic software controlled I/O by SDP2_DATA and SDP2_IODIR. 1b = Native mode operation (connected to hardware function). Following a transition on this pin the IEEE 1588 time is sampled by the Auxiliary Time Stamp 0 register. In this mode SDP2_IODIR should be configured as input.
SDP3_NATIVE	19	0b	RW	SDP3 Operating Mode. 0b = Generic software controlled I/O by SDP3_DATA and SDP3_IODIR. 1b = Native mode operation (connected to hardware function). Following a transition on this pin the IEEE 1588 time is sampled by the Auxiliary Time Stamp 1 register. In this mode SDP3_IODIR should be configured as input.
RESERVED	24:20	0x0	RSV	Reserved.
SDP1_FUNCTION	25	0b	RW	SDP1 Native mode functionality (SDP1_NATIVE = 1): 0 - Target Time 1 functionality. SDP1_IODIR should be configured as output. See note 3 that follows this table.
RESERVED	31:26	0x0	RSV	Reserved.

Note: 1. Initial values is read from the NVM.

Note: 2. It is assumed that bit 15 of NC-SI Configuration 2 word in NVM is cleared; otherwise, SDP0 pins are used as input pins that encode the NC-SI Package ID of the controller.



Note: 3. It is assumed that bit SDP_FUNC_OFF_EN of PCIe Control 3 word in NVM is cleared; otherwise, SDP1 pins are strapped during PE_RST_N to determine disablement of both PCIe functions.

8.2.4.1.6 PHY GPIO Register - PHY_GPIO (0x00000028)

Field	Bit(s)	Init.	Access 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.

8.2.4.1.7 MAC GPIO Register - MAC_GPIO (0x00000030)

Field	Bit(s)	Init.	Access 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.

8.2.4.1.8 PHY Interrupt Status Register 0 - PHYINT_STATUS0 (0x00000100)

Register contents is valid once corresponding EEMNGCTL.CFG_DONE0/1 bit is asserted by firmware. Bits are set by firmware to notify host of the PHY interrupts that happened. This register is reset by hardware only at power-up events. Host is responsible to clear it when completing PHY interrupt handling routine.

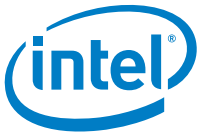
Field	Bit(s)	Init.	Access Type	Description
PMA_RECEIVE_LINK_STATUS	0	0b	RW	Reflects inverse state of PHY register bit 1.1.2 before it was set by firmware read.
PMA_TRANSMIT_FAULT	1	0b	RW	Reflects state of PHY register bit 1.8.B before it was cleared by firmware read.
PMA_RECEIVE_FAULT	2	0b	RW	Reflects state of PHY register bit 1.8.A before it was cleared by firmware read.
PMA_RESERVED	7:3	0x0	RW	Reserved. Read as written.
PCS_RECEIVE_LINK_STATUS	8	0b	RW	Reflects inverse state of PHY register bit 3.1.2 before it was set by firmware read.

Field	Bit(s)	Init.	Access Type	Description
PCS_TRANSMIT_FAULT	9	0b	RW	Reflects state of PHY register bit 3.8.B before it was cleared by firmware read.
PCS_RECEIVE_FAULT	10	0b	RW	Reflects state of PHY register bit 3.8.A before it was cleared by firmware read.
PCS_10GBASE_T_BLOCK_LOCK_LATCHED	11	0b	RW	Reflects inverse state of PHY register bit 3.21.F before it was set by firmware read.
PCS_10GBASE_T_HIGH_BER_LATCHED	12	0b	RW	Reflects state of PHY register bit 3.21.E before it was cleared by firmware read.
PCS_CRC_ERROR	13	0b	RW	Reflects state of PHY register bit 3.EC00.F before it was cleared by firmware read.
PCS_LDPC_DECODE_FAILURE	14	0b	RW	Reflects state of PHY register bit 3.EC00.E before it was cleared by firmware read.
PCS_INVALID_65B_BLOCK	15	0b	RW	Reflects state of PHY register bit 3.EC00.8 before it was cleared by firmware read.
PCS_CHANGE_IN_AUXILIARY_BIT	16	0b	RW	Reflects state of PHY register bit 3.EC00.0 before it was cleared by firmware read.
PCS_RESERVED	23:17	0x0	RW	Reserved. Read as written.
PHY_XS_RESERVED	31:24	0x0	RW	Reserved. Read as written.

8.2.4.1.9 PHY Interrupt Status Register 1 - PHYINT_STATUS1 (0x00000104)

Register contents is valid once corresponding EEMNGCTL.CFG_DONE0/1 bit is asserted by firmware. Bits are set by firmware to notify host of the PHY interrupts that happened. This register is reset by hardware only at power-up events. Host is responsible to clear it when completing PHY interrupt handling routine.

Field	Bit(s)	Init.	Access Type	Description
AUTO_NEGOTIATION_EXTENDED_NEXT_PAGE_RECEIVED	0	0b	RW	Reflects state of PHY register bit 7.1.6 before it was cleared by firmware read.
AUTO_NEGOTIATION_REMOTE_FAULT	1	0b	RW	Reflects state of PHY register bit 7.1.4 before it was cleared by firmware read.



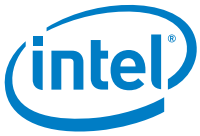
Field	Bit(s)	Init.	Access Type	Description
AUTO_NEGOTIATION_LINK_STATUS	2	0b	RW	Reflects inverse state of PHY register bit 7.1.2 before it was cleared by firmware read.
AUTO_NEGOTIATION_MASTER_SLAVE_CONFIGURATION_FAULT	3	0b	RW	Reflects state of PHY register bit 7.21.F before it was cleared by firmware read.
AUTO_NEGOTIATION_COMPLETED_FOR_NON_SUPPORTED_RATE	4	0b	RW	Reflects state of PHY register bit 7.CC00.3 before it was cleared by firmware read.
AUTO_NEGOTIATION_COMPLETED_FOR_SUPPORTED_RATE	5	0b	RW	Reflects state of PHY register bit 7.CC00.2 before it was cleared by firmware read.
AUTO_NEGOTIATION_AUTOMATIC_DOWNSHIFT	6	0b	RW	Reflects state of PHY register bit 7.CC00.1 before it was cleared by firmware read.
AUTO_NEGOTIATION_CONNECTION_STATE_CHANGE	7	0b	RW	Reflects state of PHY register bit 7.CC00.0 before it was cleared by firmware read.
AUTO_NEGOTIATION_100BASE_TX_DEVICE_DETECT	8	0b	RW	Reflects state of PHY register bit 7.EC01.F before it was cleared by firmware read.
AUTO_NEGOTIATION_ENERGY_ON_LINE_DETECT	9	0b	RW	Reflects state of PHY register bit 7.EC01.E before it was cleared by firmware read.
AUTO_NEGOTIATION_NEXT_PAGE_3RD_RECEIVED	10	0b	RW	Reflects state of PHY register bit 7.EC01.3 before it was cleared by firmware read.
AUTO_NEGOTIATION_NEXT_PAGE_2ND_RECEIVED	11	0b	RW	Reflects state of PHY register bit 7.EC01.2 before it was cleared by firmware read.

Field	Bit(s)	Init.	Access Type	Description
AUTO_NEGOTIATION_NEXT_PAGE_1ST_RECEIVED	12	0b	RW	Reflects state of PHY register bit 7.EC01.1 before it was cleared by firmware read.
AUTO_NEGOTIATION_BASE_PAGE_RECEIVED	13	0b	RW	Reflects state of PHY register bit 7.EC01.0 before it was cleared by firmware read.
AUTO_NEGOTIATION_10BASE_T_DEVICE_DETECT	14	0b	RW	Reflects inverse state of PHY register bit 7.EC02.2 before it was cleared by 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.
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 state of PHY register bit 4.CC02.F:C before it was cleared by firmware read.

8.2.4.1.10 PHY Interrupt Status Register 2 - PHYINT_STATUS2 (0x00000108)

Register contents is valid once corresponding EEMNGCTL.CFG_DONE0/1 bit is asserted by firmware. Bits are set by firmware to notify host of the PHY interrupts that happened. This register is reset by hardware only at power-up events. Host is responsible to clear it when completing PHY interrupt handling routine.

Field	Bit(s)	Init.	Access Type	Description
GLOBAL_MAC_RESET	0	0b	RW	Reflects inverse state of PHY register bit 1E.1200.0 before it was cleared by firmware read.
GLOBAL_MAC_LOW_POWER_LINK_UP_MODE	1	0b	RW	Reflects state of PHY register bit 1E.1204.4 before it was cleared by firmware read.
GLOBAL_MAC_PHY_DISABLE_MODE	2	0b	RW	Reflects inverse state of PHY register bit 1E.1204.1 before it was cleared by firmware read.



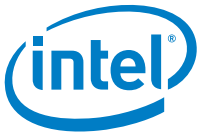
Field	Bit(s)	Init.	Access Type	Description
GLOBAL_MAC_LOW_POWER_MODE	3	0b	RW	Reflects inverse state of PHY register bit 1E.1204.0 before it was cleared by firmware read.
GLOBAL_MAC_SPI_GRANT	4	0b	RW	Reflects state of PHY register bit 1E.1206.2 before it was cleared by firmware read.
GLOBAL_MAC_SPI_CONTROL	5	0b	RW	Reflects inverse state of PHY register bit 1E.1206.1 before it was cleared by firmware read.
GLOBAL_HIGH_TEMPERATURE_FAILURE	6	0b	RW	Reflects state of PHY register bit 1E.CC00.E before it was cleared by firmware read.
GLOBAL_LOW_TEMPERATURE_FAILURE	7	0b	RW	Reflects state of PHY register bit 1E.CC00.D before it was cleared by firmware read.
GLOBAL_HIGH_TEMPERATURE_WARNING	8	0b	RW	Reflects state of PHY register bit 1E.CC00.C before it was cleared by firmware read.
GLOBAL_LOW_TEMPERATURE_WARNING	9	0b	RW	Reflects state of PHY register bit 1E.CC00.B before it was cleared by firmware read.
GLOBAL_RESET_COMPLETED	10	0b	RW	Reflects state of PHY register bit 1E.CC00.6 before it was cleared by firmware read.
GLOBAL_DEVICE_FAULT	11	0b	RW	Reflects state of PHY register bit 1E.CC00.4 before it was cleared by firmware read.
GLOBAL_PAIR_A_CHANGE_OF_STATUS	12	0b	RW	Reflects state of PHY register bit 1E.CC00.3 before it was cleared by firmware read.
GLOBAL_PAIR_B_CHANGE_OF_STATUS	13	0b	RW	Reflects state of PHY register bit 1E.CC00.2 before it was cleared by firmware read.
GLOBAL_PAIR_C_CHANGE_OF_STATUS	14	0b	RW	Reflects state of PHY register bit 1E.CC00.1 before it was cleared by firmware read.
GLOBAL_PAIR_D_CHANGE_OF_STATUS	15	0b	RW	Reflects state of PHY register bit 1E.CC00.0 before it was cleared by firmware read.
GLOBAL_MEDIUM_BER	16	0b	RW	Reflects state of PHY register bit 1E.CC01.E before it was cleared by firmware read.

Field	Bit(s)	Init.	Access Type	Description
GLOBAL_RESERVED	31:17	0x0	RW	Reserved. Read as written.

8.2.4.1.11 LED Control - LEDCTL (0x00000200)

Note: All init bits in this register are read from the NVM.

Field	Bit(s)	Init.	Access 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 field specifies the blink mode of all LEDs.0b = Blink at 200 ms on and 200 ms off.1b = Blink at 83 ms on and 83 ms off.
LED0_IVRT	6	0b	RW	LED0 Invert. This field specifies the polarity/inversion of the LED source prior to output or blink control. By default the output drives the cathode of the LED so when the LED output is 0b the LED is on.0b = LED output is active low.1b = LED output is active high.
LED0_BLINK	7	0b	RW	LED0 Blink. This field specifies whether or not to apply blink logic to the (inverted) LED control source prior to the LED output.0b = Do not blink LED output.1b = Blink LED output.
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	0x0	RSV	Reserved.
LED1_IVRT	14	0b	RW	LED1 Invert. This field specifies the polarity/inversion of the LED source prior to output or blink control. By default the output drives the cathode of the LED so when the LED output is 0b the LED is on. 0b = LED output is active low. 1b = LED output is active high.
LED1_BLINK	15	1b	RW	LED1 Blink. This field specifies whether or not to apply blink logic to the (inverted) LED control source prior to the LED output. 0b = Do not blink LED output. 1b = Blink LED output.
LED2_MODE	19:16	0x4	RW	LED2 Mode. This field specifies the control source for the LED2 output. An initial value of 0100 selects LINK/ACTIVITY indication.
RESERVED	21:20	0x0	RSV	Reserved.
LED2_IVRT	22	0b	RW	LED2 Invert. This field specifies the polarity/inversion of the LED source prior to output or blink control. By default the output drives the cathode of the LED so when the LED output is 0b the LED is on. 0b = LED output is active low. 1b = LED output is active high.
LED2_BLINK	23	0b	RW	LED2 Blink. This field specifies whether or not to apply blink logic to the (inverted) LED control source prior to the LED output. 0b = Do not blink LED output. 1b = Blink LED output.



Field	Bit(s)	Init.	Access Type	Description
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.
RESERVED	29:28	0x0	RSV	Reserved.
LED3_IVRT	30	0b	RW	LED3 Invert. This field specifies the polarity/inversion of the LED source prior to output or blink control. By default the output drives the cathode of the LED so when the LED output is 0b the LED is on. 0b = LED output is active low. 1b = LED output is active high.
LED3_BLINK	31	0b	RW	LED3 Blink. This field specifies whether or not to apply blink logic to the (inverted) LED control source prior to the LED output. 0b = Do not blink LED output. 1b = Blink LED output.

8.2.4.1.12 Extended VLAN Ether Type - EXVET (0x00005078)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	15:0	0x00	RSV	Reserved.
VET_EXT	31:16	0x8100	RW	VLAN Ether Type. The VLAN Tag Protocol Identifier (TPID). **Note:** This field appears in little endian (MS byte first on the wire).

8.2.4.2 NVM Registers

8.2.4.2.1 EEPROM-Mode Control Register - EEC (0x00010010)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	3:0	0x0	RSV	Reserved. Reads as 0b.
FWE	5:4	0x1	RW	Flash Write Enable Control. These two bits control whether or not writes to the Flash are allowed.00b = Flash erase (along with bit 31 in the FLA register).01b = Flash writes disabled.10b = Flash writes enabled.11b = Not allowed.
RESERVED	7:6	0x0	RSV	Reserved. Reads as 0b.
EE_PRES	8	0b	RO	NVM Present. Setting this bit to 1b 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 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 PHY image auto-load process. Use Reset Completed (bit 6) in PHY register 1E.CC00.
RESERVED	10	1b	RSV	Reserved.

Field	Bit(s)	Init.	Access Type	Description
EE_SIZE	14:11	0x5	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 4 KB, in power of 2 Kb units.
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 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.
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.
RESERVED	22	0b	RSV	Reserved. Reads as 0b.
FLUPD	23	0b	RW	Flash Update. Writing 1b to this bit causes the content of the internal 4 KB shadow RAM to be written into one of the first two 4 KB sectors of the Flash device (Sector 0 or Sector 1). The bit is cleared by hardware once the update completes.
RESERVED	24	0b	RSV	Reserved. Reads as 0b.
SEC1VAL	25	0b	RO	Sector 1 Valid. When set to 1b, indicates that the content of the 4 KB Sector 1 (from byte address 0x1000 to 0x1FFF) of the Flash device is valid. When set to 0b, indicates that the content of Sector 0 (from byte address 0x0000 to 0x0FFF) is valid.Meaningful only when EE_PRES bit is read as 1b.

Field	Bit(s)	Init.	Access Type	Description
FLUDONE	26	0b	RO	Flash Update Done. When set to 1b, indicates that the Flash update process that was initiated by setting FLUPD bit has completed and that the Flash is not busy by another transaction initiated via FLMNGCTL or FLA registers.
RESERVED	31:27	0x0	RSV	Reserved. Reads as 0b.

8.2.4.2.2 EEPROM-mode Read Register - EERD (0x00010014)

This register is used by software to read individual words from the internal shadow RAM that reflects the first valid 4 KB sector of the NVM. To read a word, software writes the address to the Read Address field and simultaneously writes a 1b to the Start Read field. The X540 reads the word from the internal shadow RAM and places it in the Read Data field, setting the Read Done field to 1b. Software can poll this register, looking for a 1b in the Read Done field and then using the value in the Read Data field.When this register is used to read a word from the NVM via EEPROM-mode, that word is not written to any of the X540's internal registers even if it is normally a hardware-accessed word.

Field	Bit(s)	Init.	Access Type	Description
START	0	0b	RW	Start Read.Writing a 1b to this bit causes the read of a 16-bit word from the shadow RAM at the address stored in the ADDR field and then stores the result in the DATA field. This bit is self-clearing.
DONE	1	0b	RO	Read Done.Set this bit to 1b when the EEPROM-mode read completes.Set this bit to 0b when the EEPROM-mode read is in progress. Note that writes by software are ignored.
ADDR	12:2	0x0	RW	Read Address.This field is written by software along with Start Read to indicate the address of the word to read.
RESERVED	15:13	0x0	RSV	Reserved. Reads as 0b.
DATA	31:16	0x0	RO	Read Data.Data returned from the EEPROM-Mode read.

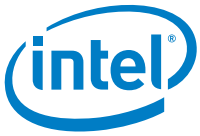
8.2.4.2.3 EEPROM-Mode Write Register - EEWR (0x00010018)

This register is used by software to write individual words in the internal shadow RAM that is about to reflect the first valid 4 KB sector of the NVM. To write a word, software writes the address to the Write Address field, the data to the Write Data field, and simultaneously writes a 1b to the Start Write field. The X540 writes the word into the internal shadow RAM, setting the Write Done field to 1b. Software can poll this register, looking for a 1b in the Write Done field before the next write. The data is effectively copied into the Flash device by use of the EEC.FLUPD command.When this register is used to write a word into the NVM, that word is not written to any of the X540's internal registers even if it is normally a hardware-accessed word.

Field	Bit(s)	Init.	Access Type	Description
START	0	0b	RW	Start Write.Writing a 1b to this bit causes the write of a 16-bit word from the DATA field into the shadow RAM at the address stored in the ADDR field. This bit is self-clearing.
DONE	1	1b	RO	Write Done.Set this bit to 1b when the EEPROM-mode write completes.Set this bit to 0b when the EEPROM-mode write is in progress. Note that writes by software are ignored.
ADDR	12:2	0x0	RW	Write Address.This field is written by software along with Start Write to indicate the address of the word to write.
RESERVED	15:13	0x0	RSV	Reserved. Reads as 0b.
DATA	31:16	0x0	RW	Write Data.Data to be written into the shadow RAM.

8.2.4.2.4 Flash Access Register - FLA (0x0001001C)

Field	Bit(s)	Init.	Access 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	RW	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 that writes to this bit have no effect.
FL_REQ	4	0b	RW	Request Flash Access. Software must write a 1b to this bit to get direct Flash access. It has access when FL_GNT is set to 1b. When software completes the access, it must then write a 0b.
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	16:6	0x0	RSV	Reserved. Reads as 0b.
FL_SIZE	19:17	0x5	RO	Flash Size. Indicates Flash size of 64KB * 2 ** “FL_SIZE”. The Flash size impacts the requested memory space for the Flash and expansion ROM BARs in PCIe configuration space (see CSRSize and FLSize fields in the BARCTRL register).Supported Flash sizes:100b = 1MB.101b = 2MB.
FL_SADDR	28:20	0x0	RW	Flash Sector Erase Address. Determines which 4 KB sector is erased when FL_SER command is used. This address is expressed in sector index units, starting from sector index 0.



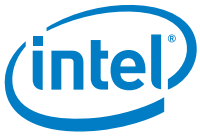
Field	Bit(s)	Init.	Access Type	Description
FL_SER	29	0b	RW	Flash Sector Erase Command. This command is sent to the Flash only if bits 5:4 of register EEC are also set to 00b. This bit is auto-cleared and reads as 0b. The 4 KB sector index to be erased is determined by FL_SADDR field.
FL_BUSY	30	0b	RO	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.
FL_DER	31	0b	RW	Flash Device Erase Command. This command is sent to the Flash only if bits 5:4 of register EEC are also set to 00b. This bit is auto-cleared and reads as 0b. The entire Flash device is erased.

8.2.4.2.5 Manageability EEPROM-Mode Control Register - EEMNGCTL (0x00010110)

This register can be read/write by manageability firmware and is read-only to host software.

The transactions performed through this register are directed to/from the internal shadow RAM. The write data is effectively copied into the Flash device by use of the EEC.FLUPD command.

Field	Bit(s)	Init.	Access Type	Description
ADDR	10:0	0x0	RW	Address. This field is written by manageability along with START bit and the WRITE bit to indicate which NVM word address to read or write. Only the first valid 4KB sector is accessible through this EEPROM-Mode interface dedicated to Manageability.
RESERVED	14:11	0x0	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 with the shadow RAM according to the write bit. This bit is self cleared by hardware.
WRITE	16	0b	RW	Write. This bit signals the device if the current operation is read or write.0b = Read1b = Write
EEBUSY	17	0b	RW	EEPROM-Mode Busy. This bit indicates that the internal shadow RAM is busy and should not be accessed.
CFG_DONE 0	18	0b	RW	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 configuration is complete and 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. Refer to PHY Interrupt Handling Flow.
CFG_DONE 1	19	0b	RW	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 configuration is complete and 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. Refer to PHY Interrupt Handling Flow.
TRANS_AB ORTED	20	0b	RO	When read as 1b, it indicates that the current EEMNGCTL access was aborted due to firmware reset event.The bit is cleared by hardware on next write access to EEMNGCTL register, regardless to the written value.
RESERVED	30:21	0x0	RSV	Reserved.
DONE	31	1b	RW	Transaction Done. This bit is cleared after the START bit and WRITE bit are set by manageability and is set back again when the shadow RAM write or read transaction completes.



8.2.4.2.6 Manageability EEPROM-Mode Read/Write Data - EEMNGDATA (0x00010114)

This register can be read/write by manageability firmware and is read-only to host software.

Field	Bit(s)	Init.	Access 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.

8.2.4.2.7 Manageability Flash Control Register - FLMNGCTL (0x00010118)

This register can be read/write by manageability firmware and is read-only to host software.

Field	Bit(s)	Init.	Access 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.
CMD	25:24	0x0	RW	Command. Indicates which command should be executed. Valid only when the CMDV bit is set.00b = Read command.01b = Write command (single byte).10b = Sector erase. The ADDR field determines the 4KB 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	28	0b	RSV	Reserved.
FW2FL	29	0b	RW	Firmware to Flash Access Firmware has taken ownership over the Flash. Flash is not owned by Firmware.This bit must be set by firmware each time it accesses the Flash via bit-banging. It can also be set for other firmware accesses. It is used by hardware to bypass the Flash protection mechanism as it does not concern firmware accesses. This bit is not used since the Flash protection mechanism has been de-featured.
DONE	30	0b	RW	Read Done. This bit is cleared by firmware when it sets the CMDV bit. It is set by hardware for each DWord read completes. This bit is Read/Clear by hardware enabling the multiple DWord read flow.

Field	Bit(s)	Init.	Access Type	Description
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 are complete. For example, the Flash device finished reading all the requested read or other accesses (write and erase).

8.2.4.2.8 Manageability Flash Read/Write Data - FLMNGDATA (0x0001011C)

This register can be read/write by manageability firmware and is read-only to host software.

Field	Bit(s)	Init.	Access Type	Description
DATA	31:0	0x0	RW	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.

8.2.4.2.9 FLASH Opcode Register - FLOP (0x0001013C)

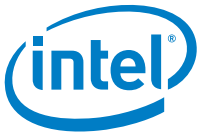
This register enables the host or firmware to define the op-code used in order to erase a sector of the Flash or erase the entire Flash. This register is reset only at power on or during LAN_PWR_GOOD assertion.

Note: The default values are applicable to Atmel* serial Flash memory devices.

Field	Bit(s)	Init.	Access Type	Description
SERASE	7:0	0x20	RW	Flash Sector Erase Instruction.The op-code for the Flash sector erase instruction.
DERASE	15:8	0xC7	RW	Flash Device Erase Instruction.The op-code for the Flash device erase instruction.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.2.10 General Receive Control - GRC (0x00010200)

Field	Bit(s)	Init.	Access Type	Description
MNG_EN	0	1b	RW	Manageability Enable.This read-only bit indicates whether or not manageability functionality is enabled.



Field	Bit(s)	Init.	Access Type	Description
APME	1	0b	RW	Advance Power Management Enable.If set to 1b, manageability wake up is enabled. The X540 sets the PME_Status bit in the Power Management Control/Status Register (PMCSR), asserts GIO_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.
RESERVED	31:2	0x0	RSV	Reserved.

8.2.4.2.11 Shadow RAM Reload - SRAMREL (0x00010210)

This register must be modified only via a read-modify-write procedure.

Field	Bit(s)	Init.	Access Type	Description
MAC2PHY_PTRVLD	5:0	0b	RW	Reserved.
LATCH_IMAGE_VLD	6	0b	RW	Enable to manually latch the Image-Valid signal from FLEEP to PHY0b: Signal is latched1b: Signal is pass-thru from FLEEP to PHY
RESERVED	31:7	0x0	RSV	Reserved.

8.2.4.3 Flow Control Registers

8.2.4.3.1 Priority Flow Control Type Opcode - PFCTOP (0x00003008; RW)

Fields definitions are the same as defined on Section 8.2.4.3.2.

8.2.4.3.2 Priority Flow Control Type Opcode - PFCTOP (0x0000431C)

Field	Bit(s)	Init.	Access Type	Description
FCT	15:0	0x8808	RW	Priority Flow Control EtherType. **Note:** This field appears in little endian (MS byte first on the wire).
FCOP	31:16	0x0101	RW	Priority Flow Control Opcode. **Note:** This field appears in big endian (LS byte first on the wire).

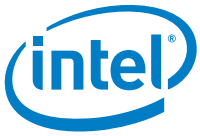
Note: Additional address(es): 0x03008 This register contains the Type and Opcode fields that are matched against a recognized priority flow control packet.

8.2.4.3.3 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) and uses a fixed slot time value of 64-byte times.

Field	Bit(s)	Init.	Access 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 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 RTTUP2TC register.



8.2.4.3.4 Flow Control Receive Threshold Low - FCRTL[n] (0x00003220 + 0x4*n, n=0...7)

Each 32-bit register (n=0... 7) refers to a different receive packet buffer.

Field	Bit(s)	Init.	Access 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 bytes 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.

Note: This register contains the receive threshold used to determine when to send an XON packet and counts in units of bytes. The lower four bits must be programmed to 0x0 (16-byte granularity). Software must set XONE to enable the transmission of XON frames. Each time incoming packets cross the receive high threshold (become more full), and then crosses the receive low threshold, with XONE enabled (1b), hardware transmits an XON frame.

8.2.4.3.5 Flow Control Receive Threshold High - FCRTH[n] (0x00003260 + 0x4*n, n=0...7)

Each 32-bit register (n=0... 7) refers to a different receive packet buffer.

Field	Bit(s)	Init.	Access 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 bytes granularity).
RESERVED	30:19	0x0	RSV	Reserved.
FCEN	31	0b	RW	Transmit Flow control enable for packet buffer n.

Note: This register contains the receive threshold used to determine when to send an XOFF packet and counts in units of bytes. This value must be at least eight bytes less than the maximum number of bytes allocated to the receive packet buffer and the lower five bits must be programmed to 0x0 (32-byte granularity). Each time the receive FIFO reaches the fullness indicated by RTH, hardware transmits a pause frame if the transmission of flow control frames is enabled.

8.2.4.3.6 Flow Control Refresh Threshold Value - FCRTV (0x000032A0)

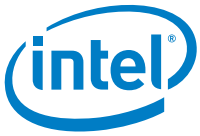
Field	Bit(s)	Init.	Access Type	Description
FC_REFRESH_TH	15:0	0x0	RW	Flow Control Refresh Threshold. This value is used to calculate the actual refresh period for sending the next pause frame if conditions for a pause state are still valid (buffer fullness above low threshold value). The formula for 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 max 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.

8.2.4.3.7 Transmit Flow Control Status - TFCS (0x0000CE00)

Field	Bit(s)	Init.	Access Type	Description
TC_XON	7:0	0xFF	RO	TC is in FC XON state.
RESERVED	31:8	0x0	RSV	Reserved.

8.2.4.3.8 Flow Control Configuration - FCCFG (0x00003D00)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	2:0	0x0	RSV	Reserved.
TFCE	4:3	0x0	RW	Transmit Flow Control Enable. These bits Indicate that the X540 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.



8.2.4.4 PCIe Registers

8.2.4.4.1 PCIe Control Register - GCR (0x00011000)

Note: This register is shared for both LAN ports.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	1:0	0b	RW	Reserved.
RESERVED	2	1b	RW	Reserved.
RESERVED	8:3	X	RSV	Reserved.
COMPLETION_TIMEOUT_RESEND_ENABLE	9	1b	RW	When set, enables a resend request after the completion timeout expires. This field is loaded from the Completion Timeout Resend bit in the NVM (PCIe General Config word 5, bit 15).
RESERVED	10	0b	RSV	Reserved.
NUMBER_OF_RESENDS	12:11	0x3	RW	The number of resends in case of timeout or poisoned.
RESERVED	17:13	0x0	RSV	Reserved.
PCIE_CAPABILITY_VERSION	18	1b	RW	Read only field reporting supported PCIe capability version. 0b = Capability version: 0x1. 1b = Capability version: 0x2.
RESERVED	20:19	0x0	RSV	Reserved.
HDR_LOG_INVERSION	21	0b	RW	If set the header log in error reporting is written as 31:0 to log1, 63:32 in log2, etc. If not, the header is written as 127:96 in log1, 95:64 in log 2, etc.
RESERVED	31:28	0x0	RSV	Reserved.

8.2.4.4.2 PCIe SerDes Configuration Register - PCIEPIPEADR (0x00011004)

Note: This register is shared by both LAN ports.

Field	Bit(s)	Init.	Access Type	Description
ADDRESS	11:0	0x0	RW	The indirect access address form SerDes register space.
QUAD_0_SELECT	12	0b	RW	The indirect access Select bit for SerDes Quad 0 (Lanes 0 to 3). Must be set for read and write accesses.
QUAD_1_SELECT	13	0b	RW	The indirect access Select bit for SerDes Quad 1 (Lanes 4 to 7). Must be set for read and write accesses.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	27:14	0x0	RSV	Reserved.
READ_ENABLE	28	0b	RW	The indirect access is a read transaction.
WRITE_ENABLE	29	0b	RW	The indirect access is a write transaction.
ERROR_INDICATION	30	0b	RW	The indirect access error indication. Valid only when Done bit is set. When set, indicates that either read or write access were not completed successfully.
DONE_INDICATION	31	0b	RW	Acknowledge for the indirect access to the CSR.

8.2.4.4.3 PCIe SerDes Data Register Tab - PCIEPIPEDAT (0x00011008)

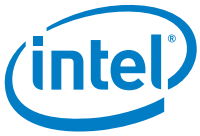
Note: This register is shared by both LAN ports.

Field	Bit(s)	Init.	Access Type	Description
QUAD_0_DATA	7:0	0x0	RW	Quad 0 (Lanes 0 to 3) data to write in the indirect access, or the returned data of the indirect read.
QUAD_1_DATA	15:8	0x0	RW	Quad 1 (Lanes 4 to7) data to write in the indirect access, or the returned data of the indirect read.
RESERVED	31:16	0x0	RSV	Reserved

8.2.4.4.4 PCIe Statistic Control Register #1 - GSCL_1 (0x00011010)

Note: This register is shared for both LAN ports.

Field	Bit(s)	Init.	Access Type	Description
GIO_COUNT_EN_0	0	0b	RW	Enables PCIe statistic counter number 0.
GIO_COUNT_EN_1	1	0b	RW	Enables PCIe statistic counter number 1.
GIO_COUNT_EN_2	2	0b	RW	Enables PCIe statistic counter number 2.
GIO_COUNT_EN_3	3	0b	RW	Enables PCIe statistic counter number 3.



Field	Bit(s)	Init.	Access Type	Description
LBC_ENABLE_0	4	0b	RW	When set, statistics counter 0 operates in leaky bucket mode. In this mode there is an internal counter that is incremented by one for each event and is decremented by one each time the LBC timer n (n=0) expires. When the internal counter reaches the value of LBC threshold n (n=0) the internal counter is cleared and the visible associated statistic counter GSCN_0_3[0] is incremented by one. When cleared, leaky bucket mode is disabled and the counter is incremented by one for each event.
LBC_ENABLE_1	5	0b	RW	When set, statistics counter 1 operates in leaky bucket mode. See detailed description for LBC Enable 0.
LBC_ENABLE_2	6	0b	RW	When set, statistics counter 2 operates in leaky bucket mode. See detailed description for LBC Enable 0.
LBC_ENABLE_3	7	0b	RW	When set, statistics counter 3 operates in leaky bucket mode. See detailed description for LBC Enable 0.
RESERVED	26:8	0x0	RSV	Reserved.
GIO_COUNT_TEST	27	0b	RW	Test Bit.Forward counters for testability.
GIO_64_BIT_EN	28	0b	RW	Enables two 64-bit counters instead of four 32-bit counters.
GIO_COUNT_RESET	29	0b	RW	Reset indication of PCIe statistic counters.
GIO_COUNT_STOP	30	0b	RW	Stop indication of PCIe statistic counters.
GIO_COUNT_START	31	0b	RW	Start indication of PCIe statistic counters.

8.2.4.4.5 PCIe Statistic Control Registers #2 - GSCL_2 (0x00011014)

- This register is shared by both LAN ports.
- PCIe Statistic Events Encoding
- Bad TLP From LL - 0x00: Each cycle, the counter increases by one, if bad TLP is received (bad crc, error reported by AL, misplaced special char, reset in thI of received tlp).
- Requests That Reached Timeout - 0x10: Number of requests that reached time out.
- NACK DLLP Received - 0x20: For each cycle, the counter increases by one, if a message was transmitted.
- Replay Happened in Retry Buffer - 0x21: Occurs when a replay happened due to timeout (not asserted when replay initiated due to NACK.
- Receive Error - 0x22: Set when one of the following occurs:
  - Decoder error occurred during training in the PHY. It is reported only when training ends.

— Decoder error occurred during link-up or until the end of the current packet (in case the link failed). This error is masked when entering/exiting EI.

- Replay Roll Over - 0x23: Occurs when replay was initiated for more than three times threshold is configurable by the PHY CSRs.
- Re-Sending Packets - 0x24: Occurs when TLP is resend in case of completion timeout.
- Surprise Link Down - 0x25: Occurs when link is unpredictably down (not because of reset or DFT).
- LTSSM in L0s in Both Rx and Tx - 0x30: Occurs when LTSSM enters L0s state in both Tx and Rx.
- LTSSM in L0s in Rx - 0x31: Occurs when LTSSM enters L0s state in Rx.
- LTSSM in L0s in Tx - 0x32: Occurs when LTSSM enters L0s state in Tx.
- LTSSM in L1 Active - 0x33: Occurs when LTSSM enters L1-active state (requested from host side).
- LTSSM in L1 software- 0x34: Occurs when LTSSM enters L1-switch (requested from switch side).

LTSSM in Recovery - 0x35: Occurs when LTSSM enters recovery state.

Note: In case of RECOVERY, entries not due to an L1 exit, if the host NAKs the L1 request, there are false L1 entry counts.

Field	Bit(s)	Init.	Access Type	Description
GIO_EVENT_NUM_0	7:0	0x0	RW	Event number that counter 0 counts (GSCN_0).
GIO_EVENT_NUM_1	15:8	0x0	RW	Event number that counter 1 counts (GSCN_1).
GIO_EVENT_NUM_2	23:16	0x0	RW	Event number that counter 2 counts (GSCN_2).
GIO_EVENT_NUM_3	31:24	0x0	RW	Event number that counter 3 counts (GSCN_3).

Note: See the PCIe Statistic Events Encoding table for details.

8.2.4.4.6 PCIe Statistic Control Register #5...#8 - GSCL_5_8[n] (0x00011030 + 0x4*n, n=0...3)

These registers control the operation of the leaky bucket counter n. While it is GSCL_5 for n=0. GSCL_6 for n=1, GSCL_7 for n=2 and GSCL_8 for n=3. Note that there are no GSCL_3 and GSCL_4 registers.

Note: These registers are shared by both LAN ports.

Field	Bit(s)	Init.	Access Type	Description
LBC_THRESHOLD_N	15:0	0x00	RW	Threshold for the leaky bucket counter n.
LBC_TIMER_N	31:16	0x00	RW	Time period in ms between decrementing the value in leaky bucket counter n.

8.2.4.4.7 PCIe Statistic Counter Registers #0...#3 - GSCN_0_3[n] (0x00011020 + 0x4*n, n=0...3)

Note: This register is shared by both LAN ports.

- GSCN_0 — n=0.
- GSCN_1 — n=1
- GSCN_2 — n=2
- GSCN_3 — n=3.

Field	Bit(s)	Init.	Access Type	Description
EVENT_COUNTER	31:0	0x0	RO	Event counter as defined in GSCL_2.GIO_EVENT_NUM fields. These registers are stuck at their maximum value of 0xFF...F and cleared on read.

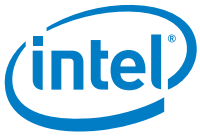
8.2.4.4.8 Function Active and Power State to Manageability - FACTPS (0x00010150)

Register for use by the device firmware for configuration.

In regular mode (port swap disabled), the power state indication of port 0 is mapped to bit FUNC0_POWER_STATE and the power state indication of port 1 is mapped to bit FUNC1_POWER_STATE. Vice versa when port swap mode is enabled.

In regular mode (port swap disabled), the auxiliary mode enable indication of port 0 is mapped to bit FUNC0_AUX_EN and the auxiliary mode enable indication of port 1 is mapped to bit FUNC1_AUX_EN. Vice versa when port swap mode is enabled.

Field	Bit(s)	Init.	Access Type	Description
FUNC0_POWER_STATE	1:0	0x0	RO	Power state indication of function 0: 00b = DR. 01b = D0u. 10b = D0a. 11b = D3.
LAN0_VALID	2	0b	RO	LAN 0 Enable. When this bit is 0b, it indicates that the LAN 0 function is disabled. When the function is enabled, the bit is 1b.This bit reflects if the function is disabled through the external pad.
FUNC0_AUX_EN	3	0b	RO	Function 0 Auxiliary (AUX) Power PM Enable bit shadow from the configuration space.
RESERVED	5:4	0x0	RSV	Reserved.
FUNC1_POWER_STATE	7:6	0x0	RO	Power state indication of function 1: 00b = DR. 01b = D0u. 10b = D0a. 11b = D3.
LAN1_VALID	8	0b	RO	LAN 1 Enable. When this bit is 0b, it indicates that the LAN 1 function is disabled. When the function is enabled, the bit is 1b. This bit reflects if the function is disabled through the external pad.
FUNC1_AUX_EN	9	0b	RO	Function 1 Auxiliary (AUX) Power PM Enable bit shadow from the configuration space.
RESERVED	28:10	0x0	RSV	Reserved.
MNGCG	29	0b	RO	Manageability Clock Gated. When set, indicates that the manageability clock is gated.
LAN_FUNCTION_SEL	30	0b	RO	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 the LAN Function Select bit in the NVM word PCIe Control 2 – Offset 0x05.



Field	Bit(s)	Init.	Access Type	Description
PM_STATE_ CHANGED	31	0b	RO	Indication that one or more of the functions power states had changed. This bit is also a signal to the manageability unit to create an interrupt.This bit is cleared on read and is not set for at least eight cycles after it was cleared.

8.2.4.4.9 PCIe Link/PHY Configuration Register - PCIEPHYADR (0x00011040)

Note: This register is shared by both LAN ports.

Field	Bit(s)	Init.	Access Type	Description
ADDRESS	11:0	0x0	RW	The indirect access address.
RESERVED	24:12	0x0	RSV	Reserved.
BYTE_ENAB LE	28:25	0x0	RW	The indirect access byte enable (4 bit).
READ_ENAB LE	29	0b	RW	The indirect access is read transaction.
WRITE_ENA BLE	30	0b	RW	The indirect access is write transaction.
DONE_INDI CATION	31	0b	RW	Acknowledge for the indirect access to the CSR.

8.2.4.4.10 PCIe Link/PHY Data Register Tab - PCIEPHYDAT (0x00011044)

Note: This register is shared by both LAN ports.

Field	Bit(s)	Init.	Access Type	Description
DATA	31:0	0x0	RW	The data to write in the indirect access or the returned data of the indirect read.

8.2.4.4.11 Software Semaphore Register - SWSM (0x00010140)

Note: This register is shared by both LAN ports.

Field	Bit(s)	Init.	Access 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. This bit is cleared on PCIe reset.
RESERVED	31:1	0x0	RSV	Reserved.

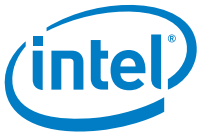
8.2.4.4.12 Firmware Semaphore Register - FWSM (0x00010148)

Note: 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.

This register is shared by both LAN ports.

Field	Bit(s)	Init.	Access Type	Description
FW_MODE	3:1	0x0	RW	Firmware mode Indicates the firmware mode as follows: 0x0 = None (manageability off). 0x1 = Reserved. 0x2 = PT mode. 0x3 = Reserved. 0x4 = Host interface enable only. Else, reserved.
RESERVED	5:4	0x0	RSV	Reserved.
EEP_RELOAD__IND	6	0b	RW	NVM reloaded indication Set to 1b after firmware reloaded NVM.Cleared by firmware once the Clear Bit host command is received from host software.
RESERVED	14:7	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 mode (bits 1-5) is invalid. Firmware should set it to 1b when it is ready (end of boot sequence).
RESET_CNT	18:16	0x0	RW	Reset counter. Firmware increments this counter upon every reset. Saturates at 7.



Field	Bit(s)	Init.	Access 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: 0x00 = No error. 0x01 = Invalid NVM checksum. 0x02 = Unlocked secured NVM. 0x03 = Clock Off host command. 0x04 = Reserved. 0x05 = C0 checksum failed. 0x06 = C1 checksum failed. 0x07 = C2 checksum failed. 0x08 = C3 checksum failed. 0x09 = TLB table exceeded. 0x0A = DMA load failed. 0x0B = Bad hardware version in patch load. 0x0C = Reserved. 0x0D = Unspecified error. 0x0E:0x3E = Reserved. 0x3F = Reserved (max error value).
PCIE_CONFIG__ERR_IND	25	0b	RW	PCIe Configuration Error Indication. Set to 1b by firmware when it fails to configure PCIe interface.Cleared by firmware upon successful configuration of PCIe interface.
PHY_SERDES0_CONFIG__ERR_IND	26	0b	RW	PHY/SERDES0 Configuration Error Indication. Set to 1b by firmware when it fails to configure PHY/SERDES of LAN0.Cleared by firmware upon successful configuration of PHY/SERDES of LAN0.
PHY_SERDES1_CONFIG__ERR_IND	27	0b	RW	PHY/SERDES1 Configuration Error indication. Set to 1b by firmware when it fails to configure PHY/SERDES of LAN1. Cleared by firmware upon successful configuration of PHY/SERDES of LAN1
RESERVED	30:28	0x0	RSV	Reserved.
FACTORY_MAC_ADDRESS_RESTORED	31	0b	RO	When set, it indicates to software that the factory MAC address was successfully restored after the last power-up event.This bit is cleared by the X540 at power up.

8.2.4.4.13 Software-Firmware Synchronization - SW_FW_SYNC (0x00010160)

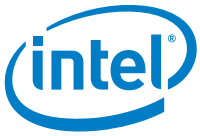
Note: See Software and Firmware Synchronization for more details.

This register is shared by both LAN ports.

Field	Bit(s)	Init.	Access Type	Description
SMBITS	10:0	0x0	RW	Semaphore Bits. Each bit represent different software semaphore agreed between software and firmware as follows. Bits 4:0 are owned by software while bits 9:5 are owned by firmware. Note that hardware does not lock access to these bits. 0 = SW_NVM_SM at 1b, NVM access is owned by software. 1 = SW_PHY_SM0 at 1b, PHY 0 access is owned by software. 2 = SW_PHY_SM1 at 1b, PHY 1 access is owned by software. 3 = SW_MAC_CSR_SM at 1b, software owns access to shared CSRs. 4 = HW_NVM_SM at 1b, NVM access is owned by hardware. 5 = FW_NVM_SM at 1b, NVM access is owned by firmware. 6 = FW_PHY_SM0 at 1b, PHY 0 access is owned by firmware. 7 = FW_PHY_SM1 at 1b, PHY 1 access is owned by firmware. 8 = FW_MAC_CSR_SM at 1b, firmware owns access to shared CSRs. 9 = NVM_Update_Started at 1b, NVM write accesses are not allowed to others. 10 = SW_MNG_SM at 1b, manageability host interface is owned by software.
RESERVED	30:11	0x0	RSV	Reserved. for future use.
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 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.

8.2.4.4.14 PCIe Control Extended Register - GCR_EXT (0x00011050)

Field	Bit(s)	Init.	Access Type	Description
VT_MODE	1:0	0x0	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 = no VT - Reserved. for the case that STSTUS. IOV Ena is not set. 01b = VT16 - Resources are allocated to 16 VFs. 10b = VT31 - Resources are allocated to 32 VFs. 11b = VT64 - Resources are allocated to 64 VFs.
RESERVED	3:2	0x0	RSV	Reserved.
APBACD	4	0b	RW	Auto PBA Clear Disable. When set to 1b, software can clear the PBA only by direct write to clear access to the PBA bit. When set to 0b, any active PBA entry is cleared on the falling edge of the appropriate interrupt request to the PCIe block. The appropriate interrupt request is cleared when software sets the associated interrupt mask bit in the EIMS (re-enabling the interrupt) or by direct write to clear to the PBA.



Field	Bit(s)	Init.	Access Type	Description
RESERVED	29:5	0x00	RSV	Reserved
BUFFERS_CLEAR_FUNC	30	0b	RW	Initiate a cleaning flow for the buffers in the transaction layer for both read and write flows.
RESERVED	31	0b	RW	Reserved.

8.2.4.4.15 Mirrored Revision ID - MREVID (0x00011064)

Note: This register is shared by both LAN ports.

Field	Bit(s)	Init.	Access Type	Description
NVM_REVID	7:0	0x00	RO	Mirroring of Rev ID loaded from NVM.
DEFAULT_REVID	15:8	0x00	RO	Mirroring of Default Rev ID, before NVM load (0 for the X540 A0).
RESERVED	31:16	0x00	RSV	Reserved.

8.2.4.4.16 PCIe Interrupt Cause - PICAUSE (0x000110B0)

Field	Bit(s)	Init.	Access Type	Description
CA	0	0b	RW1C	PCI completion abort exception.
UA	1	0b	RW1C	Unsupported IO address exception.
BE	2	0b	RW1C	Wrong byte-enable exception in the FUNC unit.
TO	3	0b	RW1C	PCI timeout exception in the FUNC unit.
BMEF	4	0b	RW1C	Asserted when bus-master-enable of the PF or one of the VFs is de-asserted.
ECCERR	5	0b	RW1C	ECC error interrupt.
ECCFIX	6	0b	RW1C	ECC Error Fix Occurred
RESERVED	31:7	0x0	RSV	Reserved.

8.2.4.4.17 PCIe Interrupt Enable - PIENA (0x000110B8)

Field	Bit(s)	Init.	Access Type	Description
CA	0	0b	RW	When set to 1b, the PCI completion abort interrupt is enabled.

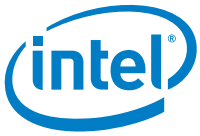
Field	Bit(s)	Init.	Access Type	Description
UA	1	0b	RW	When set to 1b, the unsupported I/O address interrupt is enabled.
BE	2	0b	RW	When set to 1b, the wrong byte-enable interrupt is enabled.
TO	3	0b	RW	When set to 1b, the PCI timeout interrupt is enabled.
BMEF	4	0b	RW	When set to 1b, the bus master enable interrupt is enabled.
ECCERR	5	0b	RW	When set to 1b, the ECC error interrupt is enabled
ECCFIX	6	0b	RW	When set to 1 the ECC Error Fix interrupt is enabled
RESERVED	31:7	0x0	RSV	Reserved.

8.2.4.5 Interrupt Registers

8.2.4.5.1 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.

Field	Bit(s)	Init.	Access 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 exception is activated by one of the following events: 1. Filter removal failed (there was no matched filter to be removed). 2. The number of remaining free filters in the flexible filter table exceeds (goes below) the FDIRCTRL.Full-Thresh.
RX_MISS	17	0b	RW1C	Missed packet interrupt is activated for each received packet that overflows the Rx packet buffer (overrun). Note that the packet is dropped and also increments the associated RXMPC[n] counter.
PCI_EXCEPTION	18	0b	RW1C	The PCI timeout exception is activated by one of the following events while the specific PCI event is reported in the INTRPT_CSR register: 1. I/O completion abort (Write to Flash when Flash is write-disabled). 2. Unsupported IO request (Wrong address). 3. Timeout occurred in the FUNC block.
MAILBOX	19	0b	RW1C	VF to PF mailbox interrupt. Caused by a VF write access to the PF mailbox.
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).



Field	Bit(s)	Init.	Access Type	Description
RESERVED	21	0b	RW1C	Reserved.
MNG	22	0b	RW1C	Manageability Event Detected. Indicates that a manageability event happened. When the device is in power-down mode, the BMC might generate a PME for the same events that would cause an interrupt when the device is at the D0 state. This interrupt bit is also used to alert the host when the BMC IP address was changed (see when EEMNGCTL.CFG_DONE0/1 bit is set by firmware (see PHY Interrupt Handling Flow).
TIMESYNC	24	0b	RW1C	TimeSync Interrupt. Indicates that a target time event occurred.
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.
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. • 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 Interrupt (Non-fatal).
TCP_TIMER	31:30	0b	RW1C	TCP Timer Expired. This bit is set when the timer expires.

8.2.4.5.2 Extended Interrupt Cause Set Register - EICS (0x00000808)

Field	Bit(s)	Init.	Access Type	Description
INTERRUPT_CAUSE_SET	30:0	0x0	WO	Setting any bit in this field, sets its corresponding bit in the EICR and generates an interrupt if enabled by EIMS register.
RESERVED	31	0b	RSV	Reserved.

8.2.4.5.3 Extended Interrupt Mask Set/Read Register - EIMS (0x00000880)

Field	Bit(s)	Init.	Access Type	Description
INTERRUPT_ENABLE	30:0	0x0	RWS	Each bit that is set to 1b enables its corresponding interrupt in the EICR. Writing a 1b to any bit sets it. Writing 0b has no impact. Reading this register provides a map of those interrupts that are enabled.
RESERVED	31	0b	RSV	Reserved.

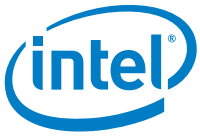
8.2.4.5.4 Extended Interrupt Mask Clear Register - EIMC (0x00000888)

Field	Bit(s)	Init.	Access Type	Description
INTERRUPT_MASK	30:0	0x0	WO	Writing a 1b to any bit clears its corresponding bit in the EIMS disabling the corresponding interrupt in the EICR. Writing 0b has no impact. Reading this register provides no meaningful data.
RESERVED	31	0b	RSV	Reserved.

8.2.4.5.5 Extended Interrupt Auto Clear Register - EIAC (0x00000810)

Note: Bits 29:20 should never be set to auto clear since they share the same MSI-X vector.

Field	Bit(s)	Init.	Access Type	Description
RTXQ_AUTO_CLEAR	15:0	0x0	RW	At 1b each bit enables auto clear of the corresponding RTxQ bits in the EICR following interrupt assertion. At 0b the corresponding bits in the EICR are not auto cleared.
RESERVED	29:16	0x0	RSV	Reserved.
TCP_TIMER_AUTO_CLEAR	30	0b	RW	At 1b, this bit enables auto clear of the TCP timer interrupt cause in the EICR following interrupt assertion. At 0b auto clear is not enabled.
RESERVED	31	0b	RSV	Reserved.



8.2.4.5.6 Extended Interrupt Auto Mask Enable register - EIAM (0x00000890)

Field	Bit(s)	Init.	Access Type	Description
AUTO_MASK	30:0	0x0	RW	At 1b each bit enables auto set and clear of its corresponding bits in the EIMS. Note that in MSIX mode, if any of the Auto Mask enable bits is set, the GPIE.EIAME bit must be set as well.
RESERVED	31	0b	RSV	Reserved.

8.2.4.5.7 Extended Interrupt Cause Set Registers 1 - EICS1 (0x00000A90)

Field	Bit(s)	Init.	Access Type	Description
INTERRUPT_CAUSE_SET	31:0	0x0	WO	Setting any bit in these registers, sets its corresponding bit in the EICR[n] and generates an interrupt if enabled by EIMS[n] register.Reading this register provides no meaningful data.

8.2.4.5.8 Extended Interrupt Cause Set Registers 2 - EICS2 (0x00000A94)

Field	Bit(s)	Init.	Access Type	Description
INTERRUPT_CAUSE_SET	31:0	0x0	WO	Setting any bit in these registers, sets its corresponding bit in the EICR[n] and generates an interrupt if enabled by EIMS[n] register.Reading this register provides no meaningful data.

8.2.4.5.9 Extended Interrupt Mask Set/Read Registers - EIMS1 (0x00000AA0)

Field	Bit(s)	Init.	Access Type	Description
INTERRUPT_ENABLE	31:0	0x0	RWS	Each bit at 1b enables its corresponding interrupt in the EICR[n]. Writing 1b to any bit sets it. Writing 0b has no impact. Reading this register provides a map of those interrupts that are enabled.Bits 15:0 of EIMS1 are mirrored in EIMS bits 15:0.

8.2.4.5.10 Extended Interrupt Mask Set/Read Registers - EIMS2 (0x00000AA4)

Field	Bit(s)	Init.	Access Type	Description
INTERRUPT_ENABLE	31:0	0x0	RWS	Each bit at 1b enables its corresponding interrupt in the EICR[n]. Writing 1b to any bit sets it. Writing 0b has no impact. Reading this register provides a map of those interrupts that are enabled.Bits 15:0 of EIMS1 are mirrored in EIMS bits 15:0.

8.2.4.5.11 Extended Interrupt Mask Clear Registers 1 - EIMC1 (0x00000AB0)

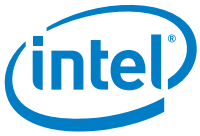
Field	Bit(s)	Init.	Access Type	Description
INTERRUPT_MASK	31:0	0x0	WO	Writing 1b to any bit clears its corresponding bit in the EIMS[n] disabling the corresponding interrupt in the EICR[n]. Writing 0b has no impact. Reading this register provides no meaningful data.

8.2.4.5.12 Extended Interrupt Mask Clear Registers 2 - EIMC2 (0x00000AB4)

Field	Bit(s)	Init.	Access Type	Description
INTERRUPT_MASK	31:0	0x0	WO	Writing 1b to any bit clears its corresponding bit in the EIMS[n] disabling the corresponding interrupt in the EICR[n]. Writing 0b has no impact. Reading this register provides no meaningful data.

8.2.4.5.13 Extended Interrupt Auto Mask Enable registers 1 - EIAM1 (0x00000AD0)

Field	Bit(s)	Init.	Access Type	Description
AUTO_MASK	31:0	0x0	RW	At 1b each bit enables auto set and clear of its corresponding bits in the EIMS[n]. Bits 15:0 of EIAM1 are mirrored in EIAM bits 15:0. Note that if any of the Auto Mask enable bits is set, the GPIE.EIAME bit must be set as well.



8.2.4.5.14 Extended Interrupt Auto Mask Enable registers 2 - EIAM2 (0x00000AD4)

Field	Bit(s)	Init.	Access Type	Description
AUTO_MASK	31:0	0x0	RW	At 1b each bit enables auto set and clear of its corresponding bits in the EIMS[n]. Bits 15:0 of EIAM1 are mirrored in EIAM bits 15:0. Note that if any of the Auto Mask enable bits is set, the GPIE.EIAME bit must be set as well.

8.2.4.5.15 MSIX to EITR Select - EITRSEL (0x00000894)

Field	Bit(s)	Init.	Access Type	Description
VFSELECT	31:0	0x0	RW	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 MSI-X Vectors Mapping to EITR). At 0x0 it selects the PF and at 0x1 it selects the VF.

8.2.4.5.16 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 MSI-X Vectors Mapping to EITR.

Note: Additional address(es): 0x012300 + 4*(n-24), n=24...128

Field	Bit(s)	Init.	Access Type	Description
RESERVED	2:0	0x0	RSV	Reserved.
ITR_INTERVAL	11:3	0x0	RW	Minimum inter-interrupt interval specified in 2.048 us units at 1 GbE and 10 GbE link. At 100 Mb/s link speed the interval is specified in 20.48 µs units. At 0x0 interrupt throttling is disabled while any event causes an immediate interrupt.
RESERVED	14:12	0x0	RSV	Reserved.
LLI_MODERATION	15	0b	RW	When set, LLI moderation is enabled. Otherwise, any LLI packet generates an immediate interrupt.LLI moderation can be set only if interrupt throttling is enabled by the ITR Interval field in this register and LLI moderation is enabled by the LL Interval field in the GPIE register.
LLI_CREDIT	20:16	0x0	RW	Reflects the current credits for associated interrupt. When CNT_WDIS is not set on write cycle, this field must be set to zero.

Field	Bit(s)	Init.	Access Type	Description
ITR_COUNTER	27:21	0x0	RW	This field represents the 7 MSbits (out of 9 bits) of the ITR counter. It is a down counter that is loaded with ITR Interval value each time the associated interrupt is asserted. When the ITR counter reaches zero it stops counting and triggers an interrupt.On a write cycle, software must set this field to 0 if CNT_WDIS in this register is cleared (write enable to the ITR counter).
RESERVED	30:28	0x0	RSV	Reserved.
CNT_WDIS	31	0b	RW	Write disable to the LLI credit and ITR counter. When set, the LLI credit and ITR counter are not overwritten by the write access. When cleared software must set the LLI credit and ITR counter to zero, which enables an immediate interrupt on packet reception. This bit is write only and always read as zero.

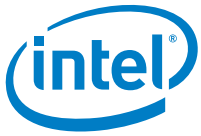
8.2.4.5.17 L3 L4 Tuples Immediate Interrupt Rx - L34TIMIR[n] (0x0000E800 + 0x4*n, n=0...127)

This register must be initialized by software.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	11:0	X	RSV	Reserved.
SIZE_BP	12	0b	RW	Size Bypass. When 1b, the size check is bypassed. When 0b, the size of the received packet is checked for LLI. See also bit 20.
RESERVED	14:13	X	RSV	Reserved.
PSH_BIT	15	0b	RW	When the PSH_Bit bit is set to 1b, the PSH flag is checked in the received packet for LLI. See also bit 20 in this register.
RESERVED	19:16	X	RSV	Reserved.
LLI_EN	20	0b	RW	Enables issuing a Low Latency Interrupt when the following conditions are met:- The received packet matches the 5-tuple filter associated with this register and...- If the Size_BP bit is cleared the packet length is smaller than the length defined by LLITHRESH.SizeThresh and...- If the PSH_Bit bit is set the PSH flag in the received packet is set
RX_QUEUE	27:21	X	RW	Identifies the Rx queue associated with this 5-tuple filter.
RESERVED	31:28	X	RSV	Reserved.

8.2.4.5.18 LLI Size Threshold - LLITHRESH (0x0000EC90)

Field	Bit(s)	Init.	Access Type	Description
SIZETHRESH	11:0	0x000	RW	Size Threshold. A packet with length below this threshold that matches one of the 5- tuple filters with active Low Latency Interrupt flag in the L34TIMIR[n] registers might trigger an LLI.



Field	Bit(s)	Init.	Access Type	Description
RESERVED	25:12	0x0	RSV	Reserved.
RESERVED	31:26	0x5	RSV	Implemented reserved bits.

8.2.4.5.19 Immediate Interrupt Rx VLAN Priority Register - IMIRVP (0x0000EC60)

Note: Additional address(es): 0x05AC0.

Field	Bit(s)	Init.	Access Type	Description
VLAN_PRI	2:0	0x0	RW	VLAN Priority. This field includes the VLAN priority threshold. When Vlan_pri_en is set to 1b, then an incoming packet with VLAN tag with a priority equal or higher to VlanPri must trigger a Low Latency Interrupt, regardless of the ITR moderation.
VLAN_PRI_EN	3	0b	RW	VLAN Priority Enable. When 1b, an incoming packet with VLAN tag with a priority equal or higher to Vlan_Pri must trigger a Low Latency Interrupt, regardless of the ITR moderation. When 0b, the interrupt is moderated by ITR.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.5.20 Immediate Interrupt Rx VLAN Priority Register - IMIRVP (0x00005AC0; RW)

Fields definitions are the same as defined in Section 8.2.4.5.19.

8.2.4.5.21 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 Mapping of Interrupt Causes for the mapping and use of these registers.

Field	Bit(s)	Init.	Access Type	Description
INT_ALLOC_0	5:0	X	RW	The interrupt allocation for Rx queue (2 x N for IVAR register N).
RESERVED	6	0b	RSV	Reserved.
INT_ALLOC_VAL_0	7	0b	RW	Interrupt Allocation Valid indication for INT_Alloc[0].
INT_ALLOC_1	13:8	X	RW	The interrupt allocation for Tx queue (2 x N for IVAR register N).
RESERVED	14	0b	RSV	Reserved.

Field	Bit(s)	Init.	Access Type	Description
INT_ALLOC_VAL_1	15	0b	RW	Interrupt Allocation Valid indication for INT_Alloc[1].
INT_ALLOC_2	21:16	X	RW	The interrupt allocation for Rx queue (2 x N+1 for IVAR register N).
RESERVED	22	0b	RSV	Reserved.
INT_ALLOC_VAL_2	23	0b	RW	Interrupt Allocation Valid indication for INT_Alloc[2].
INT_ALLOC_3	29:24	X	RW	The interrupt allocation for Tx queue (2 x N+1 for IVAR register N).
RESERVED	30	0b	RSV	Reserved.
INT_ALLOC_VAL_3	31	0b	RW	Interrupt Allocation Valid indication for INT_Alloc[3].

8.2.4.5.22 Miscellaneous Interrupt Vector Allocation - IVAR_MISC (0x00000A00)

These register maps interrupt causes into MSI-X vectors (MSI-X modes). See Mapping of Interrupt Causes for details on the use of these registers.

Note: The INT_ALLOC_VAL[1] bit default value is one to enable legacy driver functionality.

Field	Bit(s)	Init.	Access Type	Description
INT_ALLOC_0	6:0	X	RW	Defines the MSI-X vector assigned to the TCP timer interrupt cause.
INT_ALLOC_VAL_0	7	0b	RW	Valid bit for INT_Alloc[0].
INT_ALLOC_1	14:8	X	RW	Defines the MSI-X vector assigned to the Other interrupt cause.
INT_ALLOC_VAL_1	15	1b	RW	Valid bit for INT_Alloc[1].
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.5.23 RSC Enable Interrupt - RSCINT[n] (0x00012000 + 0x4*n, n=0...128)

Field	Bit(s)	Init.	Access Type	Description
RSCEN	0	1b	RW	RSC Enable. This bit enables RSC on the receive queues associated with interrupt vector n.

Field	Bit(s)	Init.	Access Type	Description
AUTORSC	1	0b	RW	Autonomous RSC Setting by interrupt event. When this bit is set, autonomous setting by hardware of the RSCEN bit in this register is enabled: The RSCEN bit is cleared before the active RSC's in the receive queues associated with this interrupt are closed. The RSCEN is set back to 1b when the software re-enables this interrupt.
RESERVED	31:2	0x0	RSV	Reserved.

8.2.4.5.24 General Purpose Interrupt Enable - GPIE (0x00000898)

Note: The X540 allows for up to three externally controlled interrupts. The three software-definable pins.

SDP[3:1] might be mapped for use as GPI interrupt bits. These mappings are enabled by theSDPx_GPIEN bits only when these signals are also configured as inputs via SDPx_IODIR. When configured to function as external interrupt pins, a GPI interrupt is generated when the corresponding pin is sampled in an active-high state.

The bit mappings are shown in GPI to SDP Bit Mappings table for clarity.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	0	0b	RSV	Reserved.
SDP0_GPIEN	1	0b	RW	General Purpose Interrupt Detection Enable for SDP0. If software-controllable IO pin SDP0 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection.
SDP1_GPIEN	2	0b	RW	General Purpose Interrupt Detection Enable for SDP1. If software-controllable IO pin SDP1 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection.
SDP2_GPIEN	3	0b	RW	General Purpose Interrupt Detection Enable for SDP2. If software-controllable IO pin SDP2 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection.
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. When set indicates that only bits 16...29 of the EICR are cleared on read. When cleared, the whole EICR is cleared on read.
EIMEN	6	0b	RW	EICS immediate interrupt enable. When set, setting bit in the EICS causes a Low Latency Interrupt. If not set, the EICS interrupt waits for EITR expiration.
LL_INTERVALTAB	10:7	0x00	RW	Low latency credits increment rate. The interval is specified in 4 µs increments at 1 GbE and 10 GbE link. It is defined in 40 µus at 100 Mb/s link. A value of 0x00 disables moderation of LLI for all interrupt vectors. When LLI is disabled by the LL Interval, the LLI Moderation bit in all EITR registers must not be set.

Field	Bit(s)	Init.	Access Type	Description
RSC_DELAY	13:11	0x0	RW	Delay from RSC completion triggered by ITR and interrupt assertion. The delay equals = (RSC Delay + 1) x 4 µs = 4, 8, 12... 32 µs.
VT_MODE	15:14	0x0	RW	Specify the number of active Virtual functions. Software must set this field the same as GCR_Ext.VT_Mode. 00b = Non-IOV mode. 10b = 32 VF mode. 01b = 16 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 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 X540 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.

8.2.4.6 MSI-X Table Registers

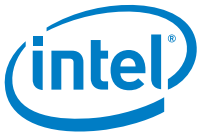
The MSI-X capability is described in MSI-X Capability. The MSI-X table is described in MSI-X Table Structure and the Pending Bit Array (PBA) is described in MSI-X PBA Structure. These registers are located in the MSI-X BAR.

8.2.4.6.1 MSI-X PBA Clear - PBACL (0x00011068; RW)

Fields definitions are the same as defined on Section 8.2.4.6.2.

8.2.4.6.2 MSI-X PBA Clear - PBACL[n] (0x000110C0 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access 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.



8.2.4.7 Receive Registers

8.2.4.7.1 Filter Control Register - FCTRL (0x00005080) RX-Filter

Note: Additional address(es): 0x11068 [n=0].

Before receive filters are updated/modified the RXCTRL.RXEN bit should be set to zero. After the proper filters have been set the RXCTRL.RXEN bit can be set to 1 to re-enable the receiver. In FCoE mode, DDp contexts should be invalidated before clearing RXCTRL.RXEN bit.

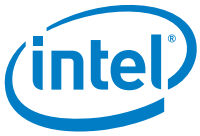
Field	Bit(s)	Init.	Access Type	Description
RESERVED	0	0b	RSV	Reserved.
SBP	1	0b	RW	Store bad packets (SBP). 0b = Do not store. 1b = Store. Note that CRC errors before the SFD are ignored. Any packet must have a valid SFD (RX_DV with no RX_ER in the XGMII/GMII i/f) in order to be recognized by the device (even bad packets). **Note:** Packets with errors are not routed to manageability even if this bit is set. **Note:** Erroneous packets might be routed to the default queue rather than the originally intended queue. **Note:** In packets shorted than 64 bytes the checksum errors might be hidden while MAC errors are reported. **Note:** 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	7:2	0x0	RSV	Reserved.
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.

8.2.4.7.2 VLAN Control Register - VLNCTRL (0x00005088)

Field	Bit(s)	Init.	Access Type	Description
VET	15:0	0x8100	RW	VLAN Ether Type (the VLAN Tag Protocol Identifier - TPID). This register contains the type field hardware matches against to recognize an 802.1Q (VLAN) Ethernet packet. For proper operation, software must not change the default setting of this field (802.3ac standard defines it as 0x8100). This field must be set to the same value as the VT field in the DMATXCTL register. **Note:** This field appears in Little Endian order (the upper byte is first on the wire (VLNCTRL.VET[15:8]).
RESERVED	27:16	0x0	RSV	Reserved.
CFI	28	0b	RW	Canonical Form Indicator bit value.If CFIEN is set to 1b, then.1q packets with CFI equal to this field are accepted; otherwise, the.1q packet is discarded.
CFIEN	29	0b	RW	Canonical Form Indicator Enable. 0b = Disabled (CFI bit not compared to decide packet acceptance); 1b = Enabled (CFI from packet must match next CFI field to accept.1q packet)
VFE	30	0b	RW	VLAN Filter Enable. 0b = Disabled (filter table does not decide packet acceptance); 1b = Enabled (filter table decides packet acceptance for.1q packets).
RESERVED	31	0b	RSV	Reserved.

8.2.4.7.3 Multicast Control Register - MCSTCTRL (0x00005090)

Field	Bit(s)	Init.	Access Type	Description
MO	1:0	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.



8.2.4.7.4 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, then an IPv4 packet that is not TCP is split after the IPv4 header.

This bit mask table enables or disables each type of header to be split. A value of 1b 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.	Access Type	Description
PSR_TYPE0	0	0b	RW	Reserved.
PSR_TYPE1	1	0b	RW	Split received NFS packets after NFS header.
PSR_TYPE2	2	0b	RW	Reserved.
PSR_TYPE3	3	0b	RW	Reserved.
PSR_TYPE4	4	0b	RW	Split received TCP packets after TCP header. PSR_type4 bit should be set to enable RSC, regardless to the header split mode.
PSR_TYPE5	5	0b	RW	Split received UDP packets after UDP header.
PSR_TYPE6	6	0b	RW	Reserved.
PSR_TYPE7	7	0b	RW	Reserved.
PSR_TYPE8	8	0b	RW	Split received IPv4 packets after IPv4 header.
PSR_TYPE9	9	0b	RW	Split received IPv6 packets after IPv6 header.
PSR_TYPE10	10	0b	RW	Reserved.
PSR_TYPE11	11	0b	RW	Reserved.
PSR_TYPE12	12	0b	RW	Split received L2 packets after L2 header.
PSR_TYPE13	13	0b	RW	Reserved.
PSR_TYPE14	14	0b	RW	Reserved.

Field	Bit(s)	Init.	Access Type	Description
PSR_TYPE15	15	0b	RW	Reserved.
PSR_TYPE16	16	0b	RW	Reserved.
PSR_TYPE17	17	0b	RW	Reserved.
PSR_TYPE18	18	0b	RW	Reserved.
RESERVED	28:19	X	RSV	Reserved.
RQPL	31:29	X	RW	Defines the number of bits to use for RSS redirection of packets dedicated to this pool.Valid values are zero, 0001b and 0010b. The default value should be 0010b, meaning that up to 4 queues can be enabled for this pool. A value of 0001b means that up to 2 queues can be enabled for this pool.A value of zero means that all the traffic of the pool is sent to queue 0 of the pool. This field is used only if MRQC.MRQE equals 1010b or 1011b.

8.2.4.7.5 Packet Split Receive Type Register - PSRTYPE[n] (0x00005480 + 0x4*n, n=0...15; RW)

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, then an IPv4 packet that is not TCP is split after the IPv4 header.

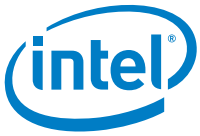
This bit mask table enables or disables each type of header to be split. A value of 1b enables an entry.

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.

Fields definitions are the same as defined on Section 8.2.4.7.4.

8.2.4.7.6 Receive Checksum Control - RXCSUM (0x00005000)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	11:0	0x00	RSV	Reserved.
IPPCSE	12	0b	RW	IP Payload Checksum Enable.
PCSD	13	0b	RW	RSS/Fragment Checksum status selection. When set to 1b, the extended descriptor write-back has the RSS field. When set to 0b, it contains the fragment checksum.
RESERVED	31:14	0x0	RSV	Reserved.



8.2.4.7.7 Receive Filter Control Register - RFCTL (0x00005008) RX-Filter

The Receive Checksum Control register controls the receive checksum off loading features of the X540.

The X540 supports the off loading of three receive checksum calculations: the Fragment Checksum, the IP Header Checksum, and the TCP/UDP Checksum.

PCSD: 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. The RXCSUM.PCSD affect is shown in the Checksum Enable/Disable table of Receive Functionality.

IPPCSE: The IPPCSE controls the Fragment Checksum calculation. As noted above, the fragment checksum shares the same location as the RSS field. The Fragment checksum is reported in the Receive descriptor when the RXCSUM.PCSD bit is cleared.

If RXCSUM.IPPCSE is cleared (the default value), the checksum calculation is not done and the value that is reported in the Rx Fragment checksum field is 0b.

If the RXCSUM.IPPCSE is set, the Fragment checksum is aimed to accelerate checksum calculation of fragmented UDP packets. Please refer to Receive UDP Fragmentation Checksum for detailed explanation.

This register should only be initialized (written) when the receiver is not enabled (for example, only write this register when RXCTRL.RXEN = 0b).

Field	Bit(s)	Init.	Access Type	Description
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	0x0	RW	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	Should not be set to 1b.
IP6XSUM_DIS	11	0b	RW	Should not be set to 1b.
RESERVED	13:12	0x0	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. Should not be set to 1b.
RESERVED	15	0b	RSV	Reserved.

Field	Bit(s)	Init.	Access Type	Description
IPV6_EXDIS	16	0b	RW	IPv6 Extension Header Disable. Disables 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 = Do not recognize previous extension headers. Should not be set to 1b.
RESERVED	17	0b	RSV	Reserved.
RESERVED	31:18	0x0	RSV	Reserved. Should be written with 0b to ensure future compatibility.

8.2.4.7.8 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.	Access Type	Description
BIT_VECTOR	31:0	X	RW	Word wide bit vector specifying 32 bits in the multicast address filter table. The X540 provides Multicast filtering for 4096 Multicast Addresses by providing single bit entry per Multicast Address. Those 4096 address locations organized in Multicast Table Array – 128 registers of 32 bits each one. Only 12 bits out of 48 bit Destination Address are considered as Multicast Address. Those 12 bits can be selected by MO field of MCSTCTRL register. The 7 MS bits of the Ethernet MAC Address (out of the 12 bits) selects the register index while the 5 LS bits (out of the 12 bits) selects the bit within a register.

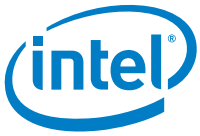
8.2.4.7.9 Receive Address Low - RAL[n] (0x0000A200 + 0x8*n, n=0...127)

While "n" is the exact unicast/Multicast address entry and it is equals to 0,1,...127.

Notes: These registers contain the lower 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.

RAL[n] (n=0...15) are also mapped to address (0x5400 + 0x8*n) to maintain back compatibility.

Field	Bit(s)	Init.	Access Type	Description
RAL	31:0	X	RW	Receive Address Low. The lower 32 bits of the 48-bit Ethernet MAC Address. **Note:** Field is defined in Big Endian (LS byte of RAL is first on the wire).



8.2.4.7.10 Receive Address High - RAH[n] (0x0000A204 + 0x8*n, n=0...127)

While "n" is the exact unicast/Multicast address entry and it is equals to 0,1,...127.

These registers contain the upper 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. AV is cleared by a master reset.

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.

RAH[n] (n=0...15) are also mapped to address (0x5404 + 0x8*n) to maintain back compatibility.

Field	Bit(s)	Init.	Access Type	Description
RAH	15:0	X	RW	Receive Address High. The upper 16 bits of the 48-bit Ethernet MAC Address. **Note:** Field is defined in Big Endian (MS byte of RAH is Last on the wire).
RESERVED	21:16	0x0	RSV	Reserved.
RESERVED	30:22	0x0	RSV	Reserved. Reads as 0x0. Ignored on write.
AV	31	0b	RW	Address valid. All Receive Addresses are not initialized by hardware and software should init them before receive is enabled. If the NVM is present, the Receive Address[0] is loaded from the NVM and its Address Valid field is set to 1b after a software or PCI reset or NVM read.

8.2.4.7.11 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.	Access 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 below: Bit 'i' in register '2*n' is associated with POOL 'i'. Bit 'i' in register '2*n+1' is associated with POOL '32+i'.

8.2.4.7.12 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.	Access Type	Description
VLAN_FLT	31:0	X	RW	VLAN Filter. Each bit 'i' in register 'n' affects packets with VLAN tags equal to 32*n+i. 128 VLAN Filter registers compose a table of 4096 bits that cover all possible VLAN tags. Each bit when set, allows packets with the associated VLAN tags to pass. Each bit when cleared, blocks packets with this VLAN tag.

8.2.4.7.13 Multiple Receive Queues Command Register - MRQC (0x0000EC80)

Note: Additional address(es): 0x05818.

Field	Bit(s)	Init.	Access 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 16 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 16 RSS queues. 0110b = Reserved. 0111b = Reserved. 1000b = Virtualization only - 64 pools, no RSS, each pool allocated 2 queues. 1001b = Reserved. 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 = Reserved. 1111b = Reserved.
RESERVED	13:4	0x0	RSV	Reserved.
RSC_DIS_LP	14	0b	RW	RSC Disable LLI and Push Coalescing. If set, RSC does not coalesce packets with LLI or PSH.
RESERVED	15	0b	RW	Reserved.

Field	Bit(s)	Init.	Access Type	Description
RSS_FIELD_ENABLE	31:16	0x0	RW	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 that on Tunnel packets IPv4-IPv6 only the IPv4 header might be used for the RSS filtering.

8.2.4.7.14 Multiple Receive Queues Command Register - MRQC (0x00005818; RW)

Fields definitions are the same as defined on Section 8.2.4.7.13.

8.2.4.7.15 RSS Queues Per Traffic Class Register - RQTC (0x0000EC70)

Field	Bit(s)	Init.	Access Type	Description
RQTC0	2:0	0x4	RW	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 0. A value of zero means that all the traffic of TC0 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b.
RESERVED	3	0b	RSV	Reserved.
RQTC1	6:4	0x4	RW	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 1. A value of zero means that all the traffic of TC1 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b.
RESERVED	7	0b	RSV	Reserved.
RQTC2	10:8	0x4	RW	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 2. A value of zero means that all the traffic of TC2 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b.
RESERVED	11	0b	RSV	Reserved.

Field	Bit(s)	Init.	Access Type	Description
RQTC3	14:12	0x4	RW	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 3. A value of zero means that all the traffic of TC3 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b.
RESERVED	15	0b	RSV	Reserved.
RQTC4	18:16	0x4	RW	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 4. A value of zero means that all the traffic of TC4 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b.
RESERVED	19	0b	RSV	Reserved.
RQTC5	22:20	0x4	RW	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 5. A value of zero means that all the traffic of TC5 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b.
RESERVED	23	0b	RSV	Reserved.
RQTC6	26:24	0x4	RW	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 6. A value of zero means that all the traffic of TC6 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b.
RESERVED	27	0b	RSV	Reserved.
RQTC7	30:28	0x4	RW	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 7. A value of zero means that all the traffic of TC7 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b.
RESERVED	31	0b	RSV	Reserved.

8.2.4.7.16 RSS Random Key Register - RSSRK[n] (0x00005C80 + 0x4*n, n=0...9; RW)

The RSS Random Key is 40 bytes wide (see RSS Hash Function).

Fields definitions are the same as defined on Section 8.2.4.7.17.

8.2.4.7.17 RSS Random Key Register - RSSRK[n] (0x0000EB80 + 0x4*n, n=0...9)

The RSS Random Key is 40 bytes wide (see RSS Hash Function).

Note: Additional address(es): 0x05C80 + 4*n, n=0...9.

Field	Bit(s)	Init.	Access Type	Description
K0	7:0	0x0	RW	RSS Key Byte '4*n+0' of the RSS random key, for each register 'n'.
K1	15:8	0x0	RW	RSS Key Byte '4*n+1' of the RSS random key, for each register 'n'.
K2	23:16	0x0	RW	RSS Key Byte '4*n+2' of the RSS random key, for each register 'n'.
K3	31:24	0x0	RW	RSS Key Byte '4*n+3' of the RSS random key, for each register 'n'.

8.2.4.7.18 Redirection Table - RETA[n] (0x00005C00 + 0x4*n, n=0...31; RW)

The redirection table has 128-entries in 32 registers.

Fields definitions are the same as defined on Section 8.2.4.7.19.

8.2.4.7.19 Redirection Table - RETA[n] (0x0000EB00 + 0x4*n, n=0...31)

The redirection table has 128-entries in 32 registers.

Additional address(es): 0x05C00 + 4*n, n=0...31.

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.	Access Type	Description
ENTRY0	3:0	X	RW	Entry0 defines the RSS output index for hash value '4 4*n+0'. While 'n' is the register index, equals to 0...31.
RESERVED	7:4	0x0	RSV	Reserved.
ENTRY1	11:8	X	RW	Entry1 defines the RSS output index for hash value '4*n+1'. While 'n' is the register index, equals to 0...31.
RESERVED	15:12	0x0	RSV	Reserved.
ENTRY2	19:16	X	RW	Entry2 defines the RSS output index for hash value '4*n+2'. While 'n' is the register index, equals to 0...31.
RESERVED	23:20	0x0	RSV	Reserved.
ENTRY3	27:24	X	RW	Entry3 defines the RSS output index for hash value '4*n+3'. While 'n' is the register index, equals to 0...31.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:28	0x0	RSV	Reserved.

8.2.4.7.20 Source Address Queue Filter - SAQF[n] (0x0000E000 + 0x4*n, n=0...127)

This register must be initialized by software.

Field	Bit(s)	Init.	Access Type	Description
SOURCE_ADDRESS	31:0	X	RW	IP source address, part of the 5-tuple queue filters. **Note:** Field is defined in Big Endian (LS byte is first on the wire).

8.2.4.7.21 Destination Address Queue Filter - DAQF[n] (0x0000E200 + 0x4*n, n=0...127)

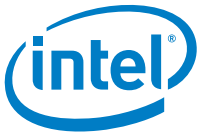
This register must be initialized by software.

Field	Bit(s)	Init.	Access Type	Description
DESTINATION_ADDRESS	31:0	X	RW	IP destination address, part of the 5-tuple queue filters. **Note:** Field is defined in Big Endian (LS byte is first on the wire).

8.2.4.7.22 Source Destination Port Queue Filter - SDPQF[n] (0x0000E400 + 0x4*n, n=0...127)

This register must be initialized by software.

Field	Bit(s)	Init.	Access Type	Description
SOURCE_PORT	15:0	X	RW	TCP/UDP source port, part of the 5-tuple queue filters. **Note:** Field is defined in Big Endian (LS byte is first on the wire).
DESTINATION_PORT	31:16	X	RW	TCP/UDP destination port, part of the 5-tuple queue filters.



8.2.4.7.23 Five tuple Queue Filter - FTQF[n] (0x0000E600 + 0x4*n, n=0...127)

Field	Bit(s)	Init.	Access Type	Description
PROTOCOL	1:0	X	RW	IP L4 protocol, part of the 5-tuple queue filters. 00b = TCP. 01b = UDP. 10b = SCTP. 11b = Other. **Note:** Encoding of the Protocol type for the 128 x 5 tuple filters is defined differently than the L4TYPE encoding for the Flow Director filters.
PRIORITY	4:2	X	RW	Priority value in case more than one 5-tuple filter matches. 000b = Reserved. 001b = Lowest priority. ... 111b = Highest priority.
RESERVED	7:5	X	RSV	Reserved.
POOL	13:8	X	RW	The Pool Index of the pool associated with this filter.
RESERVED	24:14	X	RSV	Reserved. for extension of the Pool field.
MASK	29:25	X	RW	Mask bits for the 5-tuple fields (1b - don't compare). The corresponding field participates in the match if the bit below is cleared: Bit 25 = Mask source address comparison. Bit 26 = Mask destination address comparison. Bit 27 = Mask source port comparison. Bit 28 = Mask destination port comparison. Bit 29 = Mask protocol comparison.
POOL_MASK	30	0b	RW	Mask bit for the Pool field. When set to 1b, the Pool field is not compared as part of the 5-tuple filter. Software might clear (activate) the Pool Mask bit only when operating in IOV mode.
QUEUE_ENABLE	31	0b	RW	When set, enables filtering of Rx packets by the 5-tuple defined in this filter to the queue indicated in register L34TIMIR.

8.2.4.7.24 SYN Packet Queue Filter - SYNQF (0x0000EC30)

Field	Bit(s)	Init.	Access Type	Description
QUEUE_ENABLE	0	0b	RW	When set, enables routing of Rx packets to the queue indicated in this register.
RX_QUEUE	7:1	0x00	RW	Identifies an Rx queue associated with SYN packets.
RESERVED	9:8	0b	RSV	Reserved. for extension of the Rx Queue field.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	30:10	0x0	RSV	Reserved.
SYNQFP	31	0b	RW	Defines the priority between SYNQF and 5-tuples filter. 0b = 5-tuple filter priority. 1b = SYN filter priority.

8.2.4.7.25 EType Queue Filter - ETQF[n] (0x00005128 + 0x4*n, n=0...7)

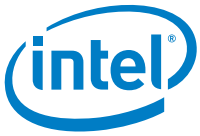
See L2 Ethertype Filters for more details on the use of this register.

Field	Bit(s)	Init.	Access Type	Description
ETYPE	15:0	0x0000	RW	Identifies the protocol running on top of IEEE 802. Used to route Rx packets containing this EtherType to a specific Rx queue. **Note:** Field is defined in Little Endian (MS byte is first on the wire).
UP	18:16	0x0	RW	User Priority. A 802.1Q UP value to be compared against the User Priority field in the Rx packet. Enabled by the UP Enable bit.
UP_ENABLE	19	0b	RW	User Priority Enable. Enables comparison of the User Priority field in the received packet.
POOL	25:20	0x00	RW	In virtualization modes, determines the target pool for the packet.
POOL_ENABLE	26	0b	RW	In virtualization modes, enables the Pool field.
FCOE	27	0b	RW	When set, packets that match this filter are identified as FCoE packets.
RESERVED	29	0b	RSV	Reserved.
IEEE_1588_TIME_STAMP	30	0b	RW	When set, packets with this EType are time stamped according to the IEEE 1588 specification.
FILTER_ENABLE	31	0b	RW	0b = The filter is disabled for any functionality. 1b = The filter is enabled. Exact actions are determined by separate bits.

8.2.4.7.26 EType Queue Select - ETQS[n] (0x0000EC00 + 0x4*n, n=0...7)

See L2 Ethertype Filters for more details on the use of this register.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	15:0	0x00	RSV	Reserved.



Field	Bit(s)	Init.	Access Type	Description
RX_QUEUE	22:16	0x00	RW	Identifies the Rx queue associated with this EType.
RESERVED	24:23	0x00	RSV	Reserved. for future extension of the Rx Queue field.
RESERVED	28:25	0x00	RSV	Reserved.
LOW_LATENCY_INTERRUPT	29	0b	RW	When set, packets that match this filter generate a Low Latency Interrupt.
RESERVED	30	0b	RSV	Reserved.
QUEUE_ENABLE	31	0b	RW	When set, enables filtering of Rx packets by the EType defined in this register to the queue indicated in this register.

8.2.4.8 Receive DMA Registers

8.2.4.8.1 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 bits of the 64 bit descriptor base address. The lower 7 bits are always ignored. The Receive Descriptor Base Address must point to a 128B aligned block of data.

Additional address(es): 0x0D000 + 0x40*(n-64), n=64...127.

Field	Bit(s)	Init.	Access Type	Description
ZERO	6:0	0x0	RW	Ignored on writes. Returns 0x0 on reads.
RDBAL	31:7	X	RW	Receive Descriptor Base Address Low.

8.2.4.8.2 Receive Descriptor Base Address High - RDBAH[n] (0x00001004 + 0x40*n, n=0...63 and 0x0000D004 + 0x40*(n-64), n=64...127)

Field	Bit(s)	Init.	Access Type	Description
RDBAH	31:0	X	RW	Receive Descriptor Base Address [63:32]

Note: Additional address(es): 0x0D004 + 0x40*(n-64), n=64...127 This register contains the upper 32 bits of the 64 bit Descriptor base address.

8.2.4.8.3 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.	Access 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 128B aligned (7 LS bit must be set to zero). **Note:** validated Lengths up to 128K (8K descriptors).
RESERVED	31:20	0x0	RSV	Reads as 0. Should be written to 0 for future compatibility.

8.2.4.8.4 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.

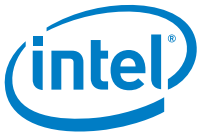
Field	Bit(s)	Init.	Access Type	Description
RDH	15:0	0x0	RO	Receive Descriptor Head. This register holds the head pointer for the receive descriptor buffer in descriptor units (16-Byte datum). The RDH is controlled by hardware.
RESERVED	31:16	0x0	RSV	Reserved. Should be written with 0x0.

8.2.4.8.5 Receive Descriptor Tail - RDT[n] (0x00001018 + 0x40*n, n=0...63 and 0x0000D018 + 0x40*(n-64), n=64...127)

Field	Bit(s)	Init.	Access Type	Description
RDT	15:0	0x0	RW	Receive Descriptor Tail
RESERVED	31:16	0x0	RSV	Reads as 0x0. Should be written as 0x0 for future compatibility.

8.2.4.8.6 Receive Descriptor Control - RXDCTL[n] (0x00001028 + 0x40*n, n=0...63 and 0x0000D028 + 0x40*(n-64), n=64...127) DMA-RX

This register contains the tail pointer for the receive descriptor buffer. The register points to a 16B datum. Software writes the tail register to add receive descriptors to the hardware free list for the ring.



Software should write an even number to the tail register, if the X540 uses the Packet Split feature. The tail pointer should be set to one descriptor beyond the last empty descriptor in Host Descriptor Ring.

Note: Additional address(es): 0x0D028 + 0x40*(n-64), n=64...127.

Additional address(es): 0x0D018 + 0x40*(n-64), n=64...127

Field	Bit(s)	Init.	Access Type	Description
RLPML	13:0	0x0	RW	Long packet size filter defined in bytes. Packets larger than the RLMPL 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. **Note:** The device closes active RSC flows when an LLI packet was dropped due to RLPML exceeded.
RESERVED	14	0b	RSV	Reserved. (software might Read and Write maintaining backward compatibility.)
RLPML_EN	15	0b	RW	Enable Long packet size filter (1b = enable).
RESERVED	24:16	0x000	RSV	Reserved. (software might Read and Write maintaining backward compatibility.)
ENABLE	25	0b	RW	Receive Queue enable. When set, the ENABLE bit enables the operation of the specific receive queue. Upon read, gets the actual status of the queue (internal indication that the queue is actually enabled/disabled).
RESERVED	26	0b	RSV	Reserved. (software might Read and Write maintaining backward compatibility.)
RESERVED	29:27	0x00	RSV	Reserved.
VME	30	0b	RW	VLAN Mode Enable1 = Strip VLAN tag from received 802.1Q packets destined to this queue.0 = Do not strip VLAN tag.
RESERVED	31	0b	RSV	Reserved.

8.2.4.8.7 Split Receive Control Registers - SRRCTL[n] (0x00001014 + 0x40*n, n=0...63 and 0x0000D014 + 0x40*(n-64), n=64...127)

Additional address(es): 0x0D014 + 0x40*(n-64), n=64...127 / 0x02100 + 4*n, [n=0...15].

BSIZEHEADER must be bigger than 0 if DESCTYPE is equal to 010, 011 100 or 101.

Field	Bit(s)	Init.	Access 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 0x0. This field should be greater or equal to 2 in queues which RSC is enabled.

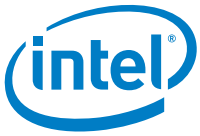
Field	Bit(s)	Init.	Access Type	Description
RESERVED	7:5	0x0	RSV	Reserved. Should be written with 0x0 to ensure future compatibility.
BSIZEHEADER	13:8	0x4	RW	Receive Buffer Size for Header Buffer. The value is in 64 bytes resolution. Value can be from 64 bytes to 1024 bytes. Note that the max supported header size is limited to 1023. Default buffer size is 256 bytes. This field must be greater than 0 if the value of DESCTYPE is greater or equal to 2.
RESERVED	21:14	0x0	RSV	Reserved.
RDMTS	24:22	0x0	RW	Receive Descriptor Minimum Threshold Size. A low latency interrupt (LLI) associated with this queue is asserted whenever the number of free descriptors is decreased to RDMTS * 64 (this event is considered as Rx ring buffer almost empty).
DESCTYPE	27:25	0x0	RW	Define the descriptor type in Rx000. 001b = Legacy. 010b = Advanced descriptor one buffer. 011b = Advanced descriptor header splitting 100b = Reserved. 101b = Reserved. 110b - 111b = Reserved. Note that advanced descriptor header splitting always uses a header buffer.
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	0x0	RSV	Reserved. Should be written as 0x0 to ensure future compatibility.

8.2.4.8.8 Split Receive Control Registers - SRRCTL[n] (0x00002100 + 0x4*n, n=0...15; RW)

Fields definitions are the same as defined on Section 8.2.4.8.7.

8.2.4.8.9 Receive DMA Control Register - RDRXCTL (0x00002F00)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	0	0b	RW	Reserved.
CRCSTRIP	1	1b	RW	Rx CRC Strip indication to the Rx DMA unit. This bit must be set the same as HLREG0.RXCRCSTRP. 1b = Strip CRC by hardware (default). 0b = No CRC Strip by hardware.
RESERVED	2	0b	RSV	Reserved.
DMAIDONE	3	0b	RW	DMA Init Done. When read as 1b indicates that the DMA init cycle completed (RO).



Field	Bit(s)	Init.	Access Type	Description
RESERVED	4	0b	RSV	Reserved.
RSC_PUSH_DIS	5	0b	RW	Disable coalescing of packets with PUSH flag asserted.
RSC_LLI_DIS	6	0b	RW	Disable coalescing of packets with LLI flag asserted.
RESERVED	8:7	0x0	RSV	Reserved.
RESERVED	12:9	0x4	RW	Reserved.
RESERVED	15:13	0x4	RW	Reserved.
RESERVED	16	0b	RW	Reserved.
Reserved	21:17	0x8	RW	Reserved.
RSCITRDIS	24:22	0x0	RW	Reserved
RSCACKC	25	0b	RW	Reserved, software should this bit to 1b.
RESERVED	26	0b	RW	Reserved, software should this bit to 1b.
RESERVED	31:27	0x0	RSV	Reserved.

8.2.4.8.10 Receive Packet Buffer Size - RXPBSIZE[n] (0x00003C00 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	9:0	0x0	RSV	Reserved.
SIZE	19:10	0x180	RW	Receive Packet Buffer Size for traffic class 'n' while 'n' is the register index. The size is defined in KB units and the default size of PB[0] is 384 KB. The default size of PB[1-7] is also 384 KB but it is meaningless in non-DCB mode. When DCB mode is enabled the size of PB[1-7] must be set to meaningful values. The total meaningful allocated PB sizes plus the size allocated to the flow director filters should be equal to 384 KB. Possible PB allocation in DCB mode for 8 x TCs is 0x30 (48 KB) for all PBs. Other possible setting of 4 x TCs is 0x60 (96 KB) for all PB[0-3] and 0x0 for PB[4-7]. **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.

8.2.4.8.11 Receive Control Register - RXCTRL (0x00003000)

Field	Bit(s)	Init.	Access Type	Description
RXEN	0	0b	RW	Receive Enable. When set to 0b filter inputs to the packet buffer are ignored.
RESERVED	31:1	0x0	RSV	Reserved.

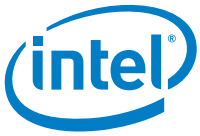
8.2.4.8.12 RSC Data Buffer Control Register - RSCDBU (0x00003028)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	6:0	0x20	RW	Reserved.
RSCACKDIS	7	0b	RW	Disable RSC for ACK packets. Disables coalescing of TCP packets without TCP payload. This bit should be set if performance problems are found.
RESERVED	31:8	0x0	RSV	Reserved.

8.2.4.8.13 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.	Access Type	Description
RSCEN	0	0b	RW	RSC Enable. When the RSCEN bit is set, RSC is enabled on this queue.
RESERVED	1	0b	RW	Reserved.
MAXDESC	3:2	0x0	RW	Maximum descriptors per Large receive as follow: 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.BSIZEPKT must not exceed 64 KB minus 1 which is the maximum total length in the IP header. It also must be larger than the expected received MSS.
RESERVED	31:4	0x00	RSV	Reserved.



8.2.4.9 Transmit Registers

8.2.4.9.1 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.	Access 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.

8.2.4.9.2 DMA Tx Control - DMATXCTL (0x00004A80)

Field	Bit(s)	Init.	Access Type	Description
TE	0	0b	RW	Transmit Enable.When set, this bit enables the transmit operation of the DMA-Tx.
NOSNOOP_LSO_HDR_BUF	1	0b	RW	NoSnoop Header buffer of TSO packets. In TSO packets, the header is fetched multiple times, once for each transmitted segment. When this bit is set and CTRL_EXT.NS_DIS is cleared, the header buffer fetching for all segments apart from the first one, is set in no-snoop mode. The first segment always uses the attribute as in the DCA registers.
RESERVED	2	1b	RW	Reserved.
GDV	3	0b	RW	Global Double VLAN mode.When set, this bit enables the Double VLAN mode.
RESERVED	4	1b	RW	Reserved.
RESERVED	15:5	0x0	RSV	Reserved.
VT	31:16	0x8100	RW	VLAN Ether-Type (the VLAN Tag Protocol Identifier - TPID). For proper operation, software must not change the default setting of this field (802.3ac standard defines it as 0x8100). This field must be set to the same value as the VET field in the VLNCTRL register.

8.2.4.9.3 DMA Tx TCP Flags Control Low - DTXTCPFLGL (0x00004A88)

This register holds the mask bits for the TCP flags in Tx segmentation (described in TCP/IP/UDP Header for the First Frame and TCP/IP Header for the Subsequent Frames).

Field	Bit(s)	Init.	Access 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	0x00	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	0x00	RSV	Reserved.

8.2.4.9.4 DMA Tx TCP Flags Control High - DTXTCPFLGH (0x00004A8C)

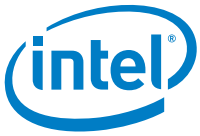
This register holds the mask bits for the TCP flags in Tx segmentation (described in TCP/IP Header for the Last Frame).

Field	Bit(s)	Init.	Access 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	0x00	RSV	Reserved.

8.2.4.9.5 Transmit Descriptor Base Address Low - TDBAL[n] (0x00006000 + 0x40*n, n=0...127)

Note: This register contains the lower bits of the 64-bit descriptor base address. The lower 7 bits are ignored. The Transmit Descriptor Base Address must point to a 128-byte aligned block of data.

Field	Bit(s)	Init.	Access Type	Description
ZERO	6:0	0x0	RW	Ignored on writes. Returns 0x0 on reads.
TDBAL	31:7	X	RW	Transmit Descriptor Base Address Low.



8.2.4.9.6 Transmit Descriptor Base Address High - TDBAH[n] (0x00006004 + 0x40*n, n=0...127)

Note: This register contains the upper 32 bits of the 64-bit Descriptor base address.

Field	Bit(s)	Init.	Access Type	Description
TDBAH	31:0	X	RW	Transmit Descriptor Base Address [63:32].

8.2.4.9.7 Transmit Descriptor Length - TDLEN[n] (0x00006008 + 0x40*n, n=0...127) DMA-TX

Field	Bit(s)	Init.	Access 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 128 KB (8 KB descriptors).
RESERVED	31:20	0x0	RSV	Reads as 0x0. Should be written as 0x0.

8.2.4.9.8 Transmit Descriptor Head - TDH[n] (0x00006010 + 0x40*n, n=0...127)

Notes: This register contains the head pointer for the transmit descriptor ring. It points to a 16-byte datum. Hardware controls this pointer.

The values in these registers might point to descriptors that are still not in the host memory. As a result, the host cannot rely on these values in order to determine which descriptor to release.

The only time that software should write to this register is after a reset (hardware reset or CTRL.RST) and before enabling the transmit function (TXDCTL.ENABLE). If software were to write to this register while the transmit function was enabled, the on-chip descriptor buffers might be invalidated and hardware could become confused.

Field	Bit(s)	Init.	Access Type	Description
TDH	15:0	0x0	RO	Transmit Descriptor Head.
RESERVED	31:16	0x0	RSV	Reserved. Should be written as 0x0.

8.2.4.9.9 Transmit Descriptor Tail - TDT[n] (0x00006018 + 0x40*n, n=0...127)

Note: This register contains the tail pointer for the transmit descriptor ring. It points to a 16-byte datum. Software writes the tail pointer to add more descriptors to the transmit ready queue. Hardware attempts to transmit all packets referenced by descriptors between head and tail.

Field	Bit(s)	Init.	Access Type	Description
TDT	15:0	0x0	RW	Transmit Descriptor Tail.
RESERVED	31:16	0x0	RSV	Reads as 0x0. Should be written as 0x0 for future compatibility.

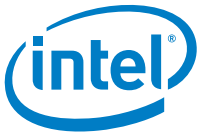
8.2.4.9.10 Transmit Descriptor Control - TXDCTL[n] (0x00006028 + 0x40*n, n=0...127)

Notes: This register controls the fetching and write-back of transmit descriptors. The three threshold values are used to determine when descriptors is read from and written to host memory.

When WTHRESH = 0b only descriptors with the RS bit set are written back.

For PTHRESH and HTHRESH recommended setting, see Transmit Descriptor Fetching.

Field	Bit(s)	Init.	Access Type	Description
PTHRESH	6:0	0x00	RW	Pre-fetch Threshold. Controls when a pre-fetch of descriptors is considered. This threshold refers to the number of valid, unprocessed transmit descriptors the X540 has in its 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	0x00	RW	Host Threshold.
RESERVED	15	0b	RSV	Reserved.



Field	Bit(s)	Init.	Access Type	Description
WTHRESH	22:16	0x00	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. **Note:** Since the default value for 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 of the X540. **Note:** When WTHRESH is set to non Zero value the software driver should not set the RS bit in the Tx descriptors. When WTHRESH is set to Zero the software driver should set the RS bit in the last Tx descriptors of every packet (in the case of TSO it is the last descriptor of the whole large send). **Note:** When Head Write-back is enabled (TDWBAL[n].Head_WB_En = 1b), the WTHRESH must be set to zero.
RESERVED	24:23	0x00	RSV	Reserved.
ENABLE	25	0b	RW	Transmit Queue enable. When set, this bit enables the operation of a specific transmit queue: Default value for all queues is 0b. Setting this bit initializes all the internal registers of a specific queue. When disabling a queue, this bit is cleared only after all activity at the queue stopped. **Note:** Following a write cycle by software, this bit is set by hardware only when the queue is actually enabled. Upon read - get the actual status of the queue (internal indication that the queue is actually enabled/disable) **Note:** When setting the global Tx enable DMATXCTL.TE the ENABLE bit of Tx queue zero is enabled as well.
SWFLSH	26	0b	RW	Transmit Software Flush This bit enables software to trigger descriptor write-back flushing, independently of other conditions. This bit is self cleared by hardware.
RESERVED	31:27	0x00	RSV	Reserved.

8.2.4.9.11 Tx Descriptor Completion Write Back Address Low - TDWBAL[n] (0x00006038 + 0x40*n, n=0...127)

Field	Bit(s)	Init.	Access Type	Description
HEAD_WB_EN	0	0b	RW	Head Write-back Enable. At 1b, Head write-back is enabled. At 0b, Head write-back is disabled. When head_WB_en is set, the X540 does not write-back Tx descriptors.
RESERVED	1	0b	RW	Reserved.

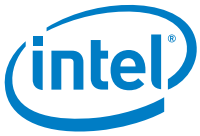
Field	Bit(s)	Init.	Access Type	Description
HEADWB_LOW	31:2	0x0	RW	Lowest 32 bits of the head write-back memory location (DWord aligned). Last 2 bits of this field are ignored and are always read as 0.0, meaning that the actual address is QWord aligned.

8.2.4.9.12 Tx Descriptor Completion Write Back Address High - TDWBAH[n] (0x0000603C + 0x40*n, n=0...127)

Field	Bit(s)	Init.	Access Type	Description
HEADWB_HIGH	31:0	0x0	RW	Highest 32 bits of head write-back memory location (for 64-bit addressing).

8.2.4.9.13 Transmit Packet Buffer Size - TXPBSIZE[n] (0x0000CC00 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	9:0	0x0	RSV	Reserved.
SIZE	19:10	0xA0	RW	Transmit Packet buffer size of TCn. At default setting (no DCB) only packet buffer 0 is enabled and TXPBSIZE values for TC 1-7 are meaningless. Other than the default configuration of the X540 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. In order to enable wire speed transmission it is recommended to set the transmit packet buffers to: 1. 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). 2. At least 3 times MSS plus PCIe latency when IPSec AH is enabled (security block operates in store and forward mode). Default value is 0xA0 (160 KB).
RESERVED	31:20	0x0	RSV	Reserved.



8.2.4.9.14 Manageability Transmit TC Mapping - MNGTXMAP (0x0000CD10)

Field	Bit(s)	Init.	Access Type	Description
MAP	2:0	0x0	RW	MAP value indicates the TC that the transmit manageability traffic is routed to.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.4.9.15 Multiple Transmit Queues Command Register - MTQC (0x00008120)

This register can be modified only as part of the initialization phase.

Note: Permitted value and functionality of: DCB_ena; VT_Ena; NUM_TC_OR_Q. For Tx queue assignment in DCB and VT modes. See Tx Queuing Schemes for details.

Field	Bit(s)	Init.	Access Type	Description
DCB_ENA	0	0b	RW	DCB Enabled Mode. See functionality in the table that follows.
VT_ENA	1	0b	RW	Virtualization Enabled Mode. When set, the X540 supports either 16, 32, or 64 pools. See functionality in the table that follows.This bit should be set the same as PFVTCTL.VT_Ena.
NUM_TC_OR_Q	3:2	0x0	RW	Number of TCs or Number of Tx Queues per Pools. See functionality in the table that follows.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.9.16 Tx Packet Buffer Threshold - TXPBTHRESH[n] (0x00004950 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
THRESH	9:0	0x96/0x0	RW	THRESHold used for checking room place in Tx packet buffer of TCn. Threshold in KB units, when the packet buffer is filled up with payload over that threshold, no more data read request is sent. Default values: 0x96 (150 KB) for TXPBSIZE0. 0x0 (0 KB) for TXPBSIZE1-7. It should be set to: (Packet Buffer Size) - MSS. For instance, if packet buffer size is set to 20 KB in corresponding TXPBSIZE.SIZE, if MSS of 9.5 KB Jumbo frames is supported for TCn, it is set to: 0xA (10 KB).

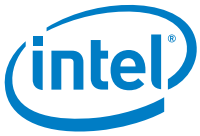
Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:10	0x0	RSV	Reserved.

8.2.4.10 DCB Registers

DCB registers are owned by the PF in an IOV mode.

8.2.4.10.1 DCB Receive Packet plane Control and Status - RTRPCS (0x00002430)

Field	Bit(s)	Init.	Access 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	0x0	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	15:7	0x0	RSV	Reserved.
RESERVED	18:16	0x0	RW	Reserved.
RESERVED	26:19	0x0	RSV	Reserved.
BWGC	27	0b	RW	This bit affects the Rx WSP arbiter. When cleared (default operation) the step T5[BWG].Credits > 0- or no GSP in the BWG- is taking affect. When set to 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.
RESERVED	31:28	0x6	RW	Reserved.



8.2.4.10.2 DCB Transmit Descriptor Plane Control and Status - RTTDCS (0x00004900)

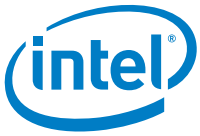
Field	Bit(s)	Init.	Access Type	Description
TDPAC	0	0b	RW	TC Transmit Descriptor Plane Arbitration Control. 0b = RR, Round Robin. 1b = WSP, Weighted Strict Priority.
VMPAC	1	0b	RW	VM transmit descriptor Plane Arbitration Control. 0b = RR, Round Robin. 1b = WRR, Weighted Round Robin.
RESERVED	3:2	0x0	RSV	Reserved.
TDRM	4	0b	RW	TC Transmit Descriptor plane Recycle Mode defines the recycle mode within a BWG. 0b = No recycle. 1b = Recycle within the BWG. (only supported mode).
RESERVED	5	0b	RSV	Reserved.
ARBDIS	6	0b	RW	DCB Arbiters Disable. When set to 1b this bit pauses the Tx descriptor plane arbitration state machine. Therefore, during normal operation this bit should be set to 0b.
RESERVED	16:7	0x0	RSV	Reserved.
LTTDESC	19:17	0x0	RO	Last Transmitted TC (RO). This field indicates the last transmitted TC in XMIT Descriptor Arbiter DMA.
RESERVED	21:20	0x0	RSV	Reserved.
BDPM	22	1b	RW	Bypass Data_Pipe monitor. in order to enable bypassing the above limit. In DCB mode, this bit must be cleared.
BPBFSM	23	1b	RW	Bypass PB free space monitor in order to enable bypassing the PB free space monitor (not checking if there is enough free space in the packet buffer before requesting the data). This bit must be cleared in DCB mode or SR-IOV mode.
RESERVED	30:24	0x0	RSV	Reserved.
SPEED_CHG	31	0b	RW	Link speed has changed. Read and clear flag. Set by hardware to indicate that the link speed has changed. Cleared by software at the end of the link speed change procedure. Refer to Link Speed Change Procedure.

8.2.4.10.3 DCB Transmit Packet plane Control and Status - RTTPCS (0x0000CD00)

Field	Bit(s)	Init.	Access 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 normal operation this bit should be set to 0b.
RESERVED	7	0b	RSV	Reserved.
TPRM	8	0b	RW	Transmit Packet plane Recycle Mode defines the recycle mode within a BWG. 0 - No recycle 1 - Recycle within the BWG
RESERVED	21:9	0x0	RSV	Reserved.
ARBD	31:22	0x224	RW	ARB_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-DCB mode. In DCB mode, should be set to 0x004.

8.2.4.10.4 DCB Receive User Priority to Traffic Class - RTRUP2TC (0x00003020)

Field	Bit(s)	Init.	Access Type	Description
UP0MAP	2:0	0x0	RW	Receive User Priority 0 to TC Mapping. When set to n, user priority 0 is bound to TC n. Used for two purposes: • Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 0x0 is routed. • Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 0* field and *Class Enable Vector* bit 0 set is sent.
UP1MAP	5:3	0x0	RW	Receive User Priority 1 to TC Mapping. When set to n, user priority 1 is bound to TC n. Used for two purposes: • Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 0x0 is routed. • Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 1* field and *Class Enable Vector* bit 1 set is sent.



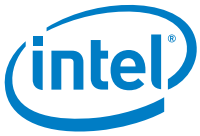
Field	Bit(s)	Init.	Access Type	Description
UP2MAP	8:6	0x0	RW	Receive User Priority 2 to TC Mapping. When set to n, user priority 2 is bound to TC n. Used for two purposes: • Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 0x0 is routed. • Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 2* field and *Class Enable Vector* bit 2 set is sent.
UP3MAP	11:9	0x0	RW	Receive User Priority 3 to TC Mapping. When set to n, user priority 3 is bound to TC n. Used for two purposes: • Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 0x0 is routed. Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 3* field and *Class Enable Vector* bit 3 set is sent.
UP4MAP	14:12	0x0	RW	Receive User Priority 4 to TC Mapping. When set to n, user priority 4 is bound to TC n. Used for two purposes: • Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 0x0 is routed. • Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 4* field and *Class Enable Vector* bit 4 set is sent.
UP5MAP	17:15	0x0	RW	Receive User Priority 5 to TC Mapping. When set to n, user priority 5 is bound to TC n. Used for two purposes: • Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 0x0 is routed. • Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 5* field and *Class Enable Vector* bit 5 set is sent.
UP6MAP	20:18	0x0	RW	Receive User Priority 6 to TC Mapping. When set to n, user priority 6 is bound to TC n. Used for two purposes: • Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 0x0 is routed. • Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 6* field and *Class Enable Vector* bit 6 set is sent.
UP7MAP	23:21	0x0	RW	Receive User Priority 7 to TC Mapping. When set to n, user priority 7 is bound to TC n. Used for two purposes: • Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 0x0 is routed. • Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 7* field and *Class Enable Vector* bit 7 set is sent.
RESERVED	31:24	0x0	RSV	Reserved.

8.2.4.10.5 DCB Transmit User Priority to Traffic Class - RTTUP2TC (0x0000C800)

Field	Bit(s)	Init.	Access Type	Description
UP0MAP	2:0	0x0	RW	Transmit User Priority 0 to TC Mapping. When set to n, User Priority 0 is bound to TC n. Used when receiving a Priority Flow Control frame with the *Timer 0* field and *Class Enable Vector* bit 0 set, to determine which TC must be paused.
UP1MAP	5:3	0x0	RW	Transmit User Priority 1 to TC Mapping. When set to n, User Priority 1 is bound to TC n. Used when receiving a Priority Flow Control frame with the *Timer 1* field and *Class Enable Vector* bit 1 set, to determine which TC must be paused.
UP2MAP	8:6	0x0	RW	Transmit User Priority 2 to TC Mapping. When set to n, User Priority 2 is bound to TC n. Used when receiving a Priority Flow Control frame with the *Timer 2* field and *Class Enable Vector* bit 2 set, to determine which TC must be paused.
UP3MAP	11:9	0x0	RW	Transmit User Priority 3 to TC Mapping. When set to n, User Priority 3 is bound to TC n. Used when receiving a Priority Flow Control frame with the *Timer 3* field and *Class Enable Vector* bit 3 set, to determine which TC must be paused.
UP4MAP	14:12	0x0	RW	Transmit User Priority 4 to TC Mapping. When set to n, User Priority 4 is bound to TC n. Used when receiving a Priority Flow Control frame with the *Timer 4* field and *Class Enable Vector* bit 4 set, to determine which TC must be paused.
UP5MAP	17:15	0x0	RW	Transmit User Priority 5 to TC Mapping. When set to n, User Priority 5 is bound to TC n. Used when receiving a Priority Flow Control frame with the *Timer 5* field and *Class Enable Vector* bit 5 set, to determine which TC must be paused.
UP6MAP	20:18	0x0	RW	Transmit User Priority 6 to TC Mapping. When set to n, User Priority 6 is bound to TC n. Used when receiving a Priority Flow Control frame with the *Timer 6* field and *Class Enable Vector* bit 6 set, to determine which TC must be paused.
UP7MAP	23:21	0x0	RW	Transmit User Priority 7 to TC Mapping. When set to n, User Priority 7 is bound to TC n. Used when receiving a Priority Flow Control frame with the *Timer 7* field and *Class Enable Vector* bit 7 set, to determine which TC must be paused.
RESERVED	31:24	0x0	RSV	Reserved.

8.2.4.10.6 DCB Receive Packet plane T4 Config - RTRPT4C[n] (0x00002140 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
CRQ	8:0	0x0	RW	Credit Refill Quantum. Amount of credits to refill in 64-byte granularity. Possible values 0x000:0x1FF (0 = 32,704 bytes).
BWG	11:9	0x0	RW	Bandwidth Group Index. Assignment of this PB to a bandwidth group.



Field	Bit(s)	Init.	Access Type	Description
MCL	23:12	0x0	RW	Max Credit Limit. Max amount of credits for a configured PB in 64-byte granularity. Possible values 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.

8.2.4.10.7 Strict Low Latency Tx Queues - TXLLQ[n] (0x000082E0 + 0x4*n, n=0...3)

Field	Bit(s)	Init.	Access 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 belonging to an LSP TC must be set as strict low latency queues. Bit 'm' in register 'n' correspond to Tx queue 32 x 'n' + 'm'.

8.2.4.10.8 DCB Transmit Descriptor Plane T2 Config - RTTDT2C[n] (0x00004910 + 0x4*n, n=0...7)

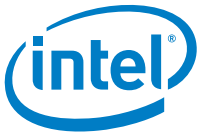
Field	Bit(s)	Init.	Access Type	Description
CRQ	8:0	0x0	RW	Credit Refill Quantum. Amount of credits to refill the TC in 64-byte granularity. Possible values 0x000 - 0x1FF (0 - 32,704 bytes).
BWG	11:9	0x0	RW	Bandwidth Group Index. Assignment of this TC to a bandwidth group.
MCL	23:12	0x0	RW	Max Credit Limit. Max amount of credits for a configured TC in 64-byte granularity. Possible values 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.

8.2.4.10.9 DCB Transmit Packet plane T2 Config - RTTPT2C[n] (0x0000CD20 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
CRQ	8:0	0x0	RW	Credit Refill Quantum. Amount of credits to refill the TC in 64-byte granularity. Possible values 0x000:0x1FF (0 = 32,704 bytes).
BWG	11:9	0x0	RW	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 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.

8.2.4.10.10 DCB Transmit QCN Rate-Scheduler MMW - RTTQCNRM (0x00004980)

Field	Bit(s)	Init.	Access Type	Description
MMW_SIZE	10:0	0x0	RW	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.



8.2.4.10.11 DCB Transmit Descriptor Plane Queue Select - RTTDQSEL (0x00004904)

Field	Bit(s)	Init.	Access 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, RTTDT1S, RTTQCNRC, RTTQCNRS
RESERVED	31:7	0x0	RSV	Reserved.

8.2.4.10.12 DCB Transmit Descriptor Plane T1 Config - RTTDT1C (0x00004908)

128 internal registers indirectly addressed via RTTDQSEL.TXDQ_IDX. When DCB is disabled, configure the pool index with the credits allocated to the entire pool.

Field	Bit(s)	Init.	Access Type	Description
CRQ	13:0	X	RW	Credit Refill Quantum. Amount of credits to refill the VM in 64-byte granularity. Possible values 0x000 - 0x3FFF (0 - 1,048,512 bytes).
RESERVED	30:14	0x0	RSV	Reserved.
RESERVED	31	0b	RSV	Reserved.

8.2.4.10.13 DCB Transmit QCN Rate-Scheduler Config - RTTQCNRC (0x00004984)

128 internal registers indirectly addressed via RTTDQSEL.TXDQ_IDX.

Field	Bit(s)	Init.	Access Type	Description
RF_DEC	13:0	X	RW	Tx QCN rate-scheduler rate factor hex part for the Tx queue indexed by TXDQ_IDX field in RTTDQSEL register. Rate factor bits that come after the hex point. Meaningful only if RS_ENA bit is set.

Field	Bit(s)	Init.	Access Type	Description
RF_INT	23:14	X	RW	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 hex point. Rate factor is defined as the ratio between the nominal link rate (like 1 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. For example, 10 Mb/s for the X540 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.
RS_ENA	31	0b	SC	Tx QCN rate-scheduler enable for the Tx queue indexed by TXDQ_IDX field in RTTDQSEL register. When set, the QCN rate programmed in this register is enforced. For example, the queue is rate controlled for QCN. At the time it is set, the current timer value is loaded into the time stamp stored for that entry. When cleared, 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. 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.

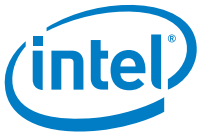
8.2.4.10.14 DCB Transmit QCN Rate-Scheduler Status - RTTQCNRS (0x00004988)

128 internal registers indirectly addressed via RTTDQSEL.TXDQ_IDX.

Field	Bit(s)	Init.	Access Type	Description
MIFS	31:0	0x0	RW	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 time stamp stored for that queue. A strict positive value means a switch in off state. It is expressed in 2's complement format.

8.2.4.10.15 DCB Transmit QCN Control Register - RTTQCNCR (0x00008B00)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	15:0	0x0	RSV	Reserved.
CNT_ETH	31:16	0x0	RW	ETHerType of CN-tagged frames. EtherType field value to be used by hardware when inserting CN-tags in every transmitted frame. **Note:** Field is defined in Little Endian (LS byte is last on the wire). Meaningful only when CN-tag insertion is enabled.



8.2.4.10.16 DCB Transmit QCN Tagging - RTTQCNTG (0x00004A90)

Field	Bit(s)	Init.	Access 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, as specified in CN-tagging. It is assumed that QCN is enabled for the User Priorities mapped to that TC.
RESERVED	31:8	0x0	RSV	Reserved.

8.2.4.10.17 DCB Transmit QCN Rate Reset - RTTQCNRR (0x0000498C)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	0	0b	RSV	Reserved.
QCN_CLEAR_ALL	1	0b	RW	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 each time the link speed has changed. This bit is self cleared by hardware.
RESERVED	31:2	0x0	RSV	Reserved.

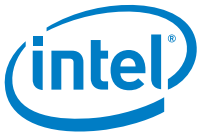
8.2.4.11 DCA Registers

8.2.4.11.1 Rx DCA Control Register - DCA_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 / 0x02200 + 4*n, [n=0...15].

In most applications non snoop should not be enabled. The software device driver might use the non-snoop option only if it can guarantee coherency between the cache and host memory when it matters. For example, read descriptor no-snoop could be enabled if the software write these descriptors using the CPU write-through instruction. Rx data write no-snoop is activated when the NSE bit is set in the receive descriptor.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
RX_DESCRIPTOR_DCA_EN	5	0b	RW	Descriptor DCA EN. When set, hardware enables DCA for all Rx descriptors written back into memory. When cleared, hardware does not enable DCA for descriptor write-backs. See the note that follows.
RX_HEADER_DCA_EN	6	0b	RW	Rx Header DCA EN. When set, hardware enables DCA for all received header buffers. When cleared, hardware does not enable DCA for Rx headers. See the note that follows.
RX_PAYLOAD_DCA_EN	7	0b	RW	Payload DCA EN. When set, hardware enables DCA for all Ethernet payloads written into memory. When cleared, hardware does not enable DCA for Ethernet payloads. Default cleared. **Note:** This bit is probably not used except for engineering purposes.
RXDESCREADNSEN	8	0b	RW	Rx descriptor Read No Snoop Enable. See the note that follows.
RXDESCREADROEN	9	1b	RW	Rx Descriptor Read Relax Order Enable.
RXDESCWBNSEN	10	0b	RW	Rx Descriptor Write Back No Snoop Enable. See the note that follows.
RXDESCWBROEN	11	0b	RW	Rx Descriptor Write Back Relax Order Enable. This bit must be 0b to allow correct functionality of the descriptors write-back.
RXDATAWRITENSEN	12	1b	RW	Rx data Write No Snoop Enable. When 0b, the No-snoop enabling (NSE) bit in the receive descriptor should be set to 0b. When 1b, the No-snoop enabling (NSE) bit in the receive descriptor is enabled.
RXDATAWRITEROEN	13	1b	RW	Rx data Write Relax Order Enable.
RXREPHEADERNSEN	14	0b	RW	Rx Split Header No Snoop Enable. See the note that follows.



Field	Bit(s)	Init.	Access Type	Description
RXREPHEADERROEN	15	1b	RW	Rx Split Header Relax Order Enable.
RESERVED	23:16	0x0	RSV	Reserved.
CPUID	31:24	0x00	RW	Physical ID (see complete description in TAG ID Allocation for Write Transactions). Legacy DCA capable platforms - the device driver, upon discovery of the physical CPU ID and CPU Bus ID, programs the CPUID field with the Physical CPU and Bus ID associated with this Rx queue. DCA 1.0 capable platforms - the device driver programs a value, based on the relevant APIC ID, associated with this Rx queue.

8.2.4.11.2 Rx DCA Control Register - DCA_RXCTRL[n] (0x00002200 + 0x4*n, n=0...15; RW)

Fields definitions are the same as defined on Section 8.2.4.11.1.

8.2.4.11.3 Tx DCA Control Registers - DCA_TXCTRL[n] (0x0000600C + 0x40*n, n=0...127)

Note: In most applications non snoop should not be enabled. The software device driver might use the non-snoop option only if it can guarantee coherency between the cache and host memory when it matters. For example, read descriptor no-snoop could be enabled if the software write these descriptors using the CPU write-through instruction.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
TX_DESCRIPTOR_DCA_EN	5	0b	RW	Descriptor DCA Enable. When set, hardware enables DCA for all Tx descriptors written back into memory. When cleared, hardware does not enable DCA for descriptor write backs. This bit is cleared as a default and also applies to head write back when enabled.
RESERVED	7:6	0x0	RSV	Reserved.
TXDESCRDNSEN	8	0b	RW	Tx Descriptor Read No Snoop Enable.
TXDESCRDROEN	9	1b	RW	Tx Descriptor Read Relax Order Enable.
TXDESCWBNSEN	10	0b	RW	Tx Descriptor Write Back No Snoop Enable.
TXDESCWBROEN	11	1b	RW	Relax Order Enable of Tx Descriptor as well as Head Pointer Write Back (when set).
TXDATAREADNSEN	12	0b	RW	Tx Data Read No Snoop Enable.

Field	Bit(s)	Init.	Access Type	Description
TXDATAREADROEN	13	1b	RW	Tx Data Read Relax Order Enable.
RESERVED	23:14	0x0	RSV	Reserved.
CPUID	31:24	0x0	RW	Physical ID (see TAG ID Allocation for Write Transactions) Legacy DCA capable platforms — the device driver, upon discovery of the physical CPU ID and CPU Bus ID, programs the CPUID field with the Physical CPU and Bus ID associated with this Tx queue. DCA 1.0 capable platforms — the device driver programs a value, based on the relevant APIC ID, associated with this Tx queue.

8.2.4.11.4 DCA Requester ID Information Register - DCA_ID (0x00011070)

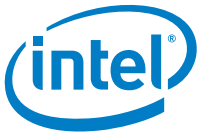
To ease software implementation, a DCA requester ID field, composed of device ID, bus # and function # is set up in MMIO space for software to program the chipset DCA Requester ID Authentication register.

Field	Bit(s)	Init.	Access Type	Description
FUNCTION_NUMBER	2:0	0x0	RO	Function Number. Function number assigned to the function based on BIOS/ operating system enumeration.
DEVICE_NUMBER	7:3	0x0	RO	Device Number. Device number assigned to the function based on BIOS/operating system enumeration.
BUS_NUMBER	15:8	0x0	RO	Bus Number. Bus number assigned to the function based on BIOS/operating system enumeration.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.11.5 DCA Control Register - DCA_CTRL (0x00011074)

Note: This register is shared by both LAN functions.

Field	Bit(s)	Init.	Access Type	Description
DCA_DIS	0	1b	RW	DCA Disable. When 0b, DCA tagging is enabled for this device. When 1b, DCA tagging is disabled for this device.



Field	Bit(s)	Init.	Access Type	Description
DCA_MODE	4:1	0x0	RW	DCA Mode. 0000b = Legacy DCA is supported. The TAG field in the TLP header is based on the following coding: bit 0 is DCA enable; bits 3:1 are CPU ID). 0001b = DCA 1.0 is supported. When DCA is disabled for a given message, the TAG field is 0000,0000b. If DCA is enabled, the TAG is set per queue as programmed in the relevant DCA Control register. All other values are undefined.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.4.12 Security Registers

Security registers are mainly concerned with the internal settings of the AES crypto engine shared by IPsec. They are owned by the PF in an IOV mode.

Refer to rate limiters for the way to modify these registers prior to enabling or disabling a security offload.

Security offload can be disabled via internal fuses as exposed in the UFUSE register. In that case, the following security related fields are not writable:

- SECTXCTRL.SECTX_DIS is read as 1b.
- SECRXCTRL.SECRX_DIS is read as 1b.
- IPSTXIDX.IPS_TX_EN is read as 0b.
- IPSRXIDX.IPS_RX_EN is read as 0b.

8.2.4.12.1 Security Tx Control - SECTXCTRL (0x00008800)

Field	Bit(s)	Init.	Access Type	Description
SECTX_DIS	0	1b	RW	Tx Security Offload Disable Bit. When set, the AES crypto engine used in Tx by IPsec offloads is disabled. This mode must be used to save the X540's power consumption when no security offload is enabled. When cleared, the AES crypto engine used in Tx by IPsec offload is enabled. Normal operating mode when a security offload is enabled. This bit is RW /RO if fused off.
TX_DIS	1	0b	RW	Disable Sec Tx Path. When set, 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. SECTXSTAT.SECTX_RDY bit is de-asserted until the path is emptied by hardware. When cleared, Tx data path is enabled. Normal operating mode.

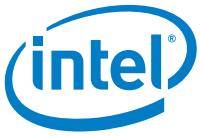
Field	Bit(s)	Init.	Access Type	Description
STORE_FORWARD	2	0b	RW	Tx Sec Buffer Mode. When set, a complete frame is stored in the internal security Tx buffer prior to being forwarded to the MAC. Operating mode when IPsec offload is enabled (as requested to overwrite ICV field in AH frames). Note that it increases the Tx internal latencies (for all TCs). When cleared, Tx sec buffer is operated in pass-through mode. Operating mode when no security offload is enabled.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.4.12.2 Security Tx Status - SECTXSTAT (0x00008804)

Field	Bit(s)	Init.	Access Type	Description
SECTX_RDY	0	0b	RO	Tx Security Block Ready For Mode Change. When set, it indicates that the internal data path from the Tx packet buffers to the Tx security block has been emptied. As a result, the security mode can be changed by software. When cleared, it indicates that the internal data path from the Tx packet buffers to the Tx security block is not empty. As such, software cannot change the security mode. This bit is polled by software once SECTXCTRL.TX_DIS bit was set.
SECTX_OFF_DIS	1	0b	RO	Tx Security Offload Disabled. When set, it indicates that the Tx security offload feature is disabled by fuse or strapping pin.
ECC_TXERR	2	0b	RO	Unrecoverable ECC Error in the Tx SA Table or SEC Tx FIFO occurred. When set, it 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. When cleared, no ECC error occurred on the Tx SA table from the last time device has reset.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.4.12.3 Security Tx Buffer Almost Full - SECTXBUFFAF (0x00008808)

Field	Bit(s)	Init.	Access Type	Description
FULLTHRESH	9:0	0x250	RW	Tx Security Buffer Almost Full Threshold (relatively to full capacity). The size of the security buffer is 0x274 lines of 16 bytes. 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 an entire jumbo packet.
RESERVED	31:10	0x0	RSV	Reserved.



8.2.4.12.4 Security Tx Buffer Minimum IFG - SECTXMINIFG (0x00008810)

Field	Bit(s)	Init.	Access Type	Description
MINSECIFG	3:0	0x1	RW	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.

8.2.4.12.5 Security Rx Control - SECRXCTRL (0x00008D00)

Field	Bit(s)	Init.	Access Type	Description
SECRX_DIS	0	1b	RW	Rx Security offload Disable bit. When set, the AES crypto engine used in Rx when IPsec offload is disabled. This mode must be used to save the X540's power consumption when no security offload is enabled. When cleared, the AES crypto engine used in Rx when IPsec offload is enabled. Normal operating mode when a security offload is enabled.
RX_DIS	1	0b	RW	Disable Sec Rx Path. When set, 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. SECRXSTAT.SECRX_RDY bit is de-asserted until the path is emptied by hardware. When cleared, Rx data path is enabled. Normal operating mode. This bit is RW / RO if fused off.
RESERVED	31:2	0x0	RSV	Reserved.

8.2.4.12.6 Security Rx Status - SECRXSTAT (0x00008D04)

Field	Bit(s)	Init.	Access Type	Description
SECRX_RDY	0	0b	RO	Rx Security Block Ready For Mode Change. When set, it indicates that the internal data path from the Rx MAC to the Rx security block has been emptied. As a result, the security mode can be changed by software. When cleared, it indicates that the internal data path from the Rx MAC to the Rx security block is not empty. As such, software cannot change the security mode. This bit is polled by software once SECRXCTRL.RX_DIS bit was set.
SECRX_OFF_DIS	1	0b	RO	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	Unrecoverable ECC Error in an Rx SA Table Occurred. When set, it 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. When cleared, no ECC error occurred on the Rx SA table from the last time the device was reset.
RESERVED	31:3	0x0	RSV	Reserved.

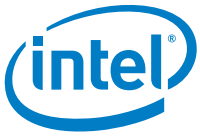
8.2.4.13 IPsec Registers

IPsec registers are owned by the PF in an IOV mode.

8.2.4.13.1 IPsec Tx Index - IPSTXIDX (0x00008900)

Note: 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.	Access Type	Description
IPS_TX_EN	0	0b	RW	IPsec Tx Offload Enable Bit. When set, IPsec offload ability is enabled for Tx path. When cleared, IPsec offload ability is disabled for Tx path, regardless of the contents of the Tx SA table.
RESERVED	2:1	0x0	RSV	Reserved.
SA_IDX	12:3	0x0	RW	SA index for indirect access into the Tx SA table.
RESERVED	29:13	0x0	RSV	Reserved.



Field	Bit(s)	Init.	Access Type	Description
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.

8.2.4.13.2 IPsec Tx Key Registers - IPSTXKEY[n] (0x00008908 + 0x4*n, n=0...3)

Field	Bit(s)	Init.	Access Type	Description
AES_128_KEY	31:0	0x0	RW	4 bytes of 16 bytes key that has been read/written from/into 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.

8.2.4.13.3 IPsec Tx Salt Register - IPSTXSALT (0x00008904)

Field	Bit(s)	Init.	Access Type	Description
AES_128_SALT	31:0	0x0	RW	4-byte salt that has been read/written from/into the Tx SA entry pointed to by SA_IDX.

8.2.4.13.4 IPsec Rx Index - IPSRXIDX (0x00008E00)

Note: 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.

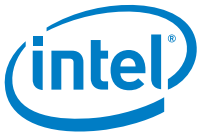
Software is not allowed to write/read access registers that belong to different Rx SA tables without writing the IPSRXIDX register in between for setting the WRITE/READ bit. Refer to Rx SA tables access rules described in Access to Rx SA Tables. Software should not make changes in the Rx SA tables while changing the IPSEC_EN bit.

Field	Bit(s)	Init.	Access Type	Description
IPS_RX_EN	0	0b	RW	IPsec Rx Offload Enable Bit. When set, IPsec offload ability is enabled for Rx path. When cleared, IPsec offload ability is disabled for Rx path, regardless of the contents of Rx SA tables.
TABLE	2:1	0x0	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 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=10 and TB_IDX=0x9, then 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=01.
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=10 and TB_IDX=0x9, then the value written in IPSRXSPI 0...3 registers is loaded into the SPI table entry 9.

8.2.4.13.5 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.	Access Type	Description
IPADDR	31:0	0x0	RW	4 bytes of 16 bytes 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:** Field is defined in Big Endian (LS byte is first on the wire).



8.2.4.13.6 IPsec Rx SPI Register - IPSRXSPI (0x00008E14)

This register is related to the Rx SPI table.

Field	Bit(s)	Init.	Access Type	Description
SPI	31:0	0x0	RW	SPI field for the SPI entry. **Note:** Field is defined in Big Endian (LS byte is first on the wire).

8.2.4.13.7 IPsec Rx SPI Register IP Index - IPSRXIPIDX (0x00008E18)

This register is related to the Rx SPI table.

Field	Bit(s)	Init.	Access 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.

8.2.4.13.8 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.	Access Type	Description
AES_128_KEY	31:0	0x0	RW	4 bytes of 16 bytes key of the key entry. n=0 contains the LSB of the key. n=3 contains the MSB of the key.

8.2.4.13.9 IPsec Rx Salt Register - IPSRXSALT (0x00008E2C)

This register is related to the Rx key table.

Field	Bit(s)	Init.	Access Type	Description
AES_128_SALT	31:0	0x0	RW	4-byte salt associated to the key entry.

8.2.4.13.10 IPsec Rx Mode Register - IPSRXMOD (0x00008E30)

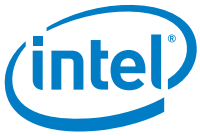
This register is related to the Rx key table.

Field	Bit(s)	Init.	Access Type	Description
VALID	0	0b	RW	Valid Bit. When set, the key entry is valid. When cleared, the key entry is not valid.
RESERVED	1	0b	RSV	Reserved.
PROTO	2	0b	RW	IPsec Protocol select. When set, the key entry off loads ESP packets. When cleared, the key entry off loads AH packets.
DECRYPT	3	0b	RW	Decryption Bit When set, hardware performs decryption offload for this key entry. Meaningful only if the PROTO bit is set (for example, ESP mode).
IPV6	4	0b	RW	IPv6 type. When set, only matched IPv6 packet are offloaded for that key entry. When cleared, only matched IPv4 packets are offloaded for that key entry.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.4.14 Timers Registers

8.2.4.14.1 TCP Timer - TCPTIMER (0x0000004C)

Field	Bit(s)	Init.	Access Type	Description
DURATION	7:0	0x0	RW	Duration. Duration of the TCP interrupt interval, in ms.
KICKSTART	8	0b	RW	Counter Kick-start. Writing 1b to this bit kick starts the counter down count from the initial value defined in *Duration* field. Writing 0b has no effect (WS).
TCPCOUNTEN	9	0b	RW	TCP Count Enable When set to 1b, the TCP timer counting is enabled. When set to 0b, it is disabled. Upon enabling, the TCP counter must count from its internal state. If the internal state is equal to zero, the down count does not restart until *KickStart* is activated. If the internal state is not 0b, the down count continues from the internal state.



Field	Bit(s)	Init.	Access Type	Description
TCPCOUNTFINISH	10	0b	RW	TCP Count Finish. This bit enables software to trigger a TCP timer interrupt, regardless of the internal state. Writing 1b to this bit triggers an interrupt and resets the internal counter to its initial value. Down count does not restart until either *KickStart* is activated or *Loop* is set. Writing 0b to this bit has no effect (WS).
LOOP	11	0b	RW	TCP Loop. When set to 1b, the TCP counter must reload duration each time it reaches zero and must down count from this point without kick starting. When set to 0b, the TCP counter must stop at a zero value and must not re-start until *KickStart* is activated.
RESERVED	31:12	0x0	RSV	Reserved.

8.2.4.15 FCoE Registers

FCoE Tx registers

8.2.4.15.1 Tx FC SOF Flags Register - TSOFF (0x00004A98)

Field	Bit(s)	Init.	Access Type	Description
SOF0	7:0	0x2D	RW	Start Of Frame 0. Class 2 Start of Frame used in the first packet of FC sequence. Default setting of SOFi2.
SOF1	15:8	0x2E	RW	Start Of Frame 1. Class 3 Start of Frame used in the first packet of FC sequence. Default setting of SOFi3.
SOF2	23:16	0x35	RW	Start Of Frame 2. Class 2 Start of Frame used in all packets but the first one of FC sequence. Default setting of SOFn2.
SOF3	31:24	0x36	RW	Start Of Frame 3. Class 3 Start of Frame used in all packets but the first one of FC sequence. Default setting of SOFn3.

8.2.4.15.2 Tx FC EOF Flags Register - TEOFF (0x00004A94)

Field	Bit(s)	Init.	Access Type	Description
EOF0	7:0	0x41	RW	End Of Frame 0. By default it is set to EOFn code used in all packets but the last one on a sequence.
EOF1	15:8	0x42	RW	End Of Frame 1. By default it is set to EOFt code used to close a sequence.
EOF2	23:16	0x49	RW	End Of Frame 2. By default it is set to EOFni code.
EOF3	31:24	0x50	RW	End Of Frame 3. By default it is set to EOFa code.

Notes: FCoE Rx registers

8.2.4.15.3 Rx FC SOF Flags Register - RSOFF (0x000051F8)

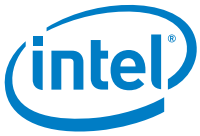
Field	Bit(s)	Init.	Access Type	Description
SOF0	7:0	0x2D	RW	Start Of Frame 0. Class 2 Start of Frame used in the first packet of FC sequence. Default setting of SOFi2.
SOF1	15:8	0x2E	RW	Start Of Frame 1. Class 3 Start of Frame used in the first packet of FC sequence. Default setting of SOFi3.
SOF2	23:16	0x35	RW	Start Of Frame 2. Class 2 Start of Frame used in all packets but the first one of FC sequence. Default setting of SOFn2.
SOF3	31:24	0x36	RW	Start Of Frame 3. Class 3 Start of Frame used in all packets but the first one of FC sequence. Default setting of SOFn3.

8.2.4.15.4 Rx FC EOF Flags Register - REOFF (0x00005158)

Field	Bit(s)	Init.	Access Type	Description
EOF0	7:0	0x41	RW	End Of Frame 0. By default it is set to EOFn code used in all packets but the last one on a sequence.
EOF1	15:8	0x42	RW	End Of Frame 1. By default it is set to EOFt code used to close a sequence.
EOF2	23:16	0x49	RW	End Of Frame 2. By default it is set to EOFni code.
EOF3	31:24	0x50	RW	End Of Frame 3. By default it is set to EOFa code.

8.2.4.15.5 FC Receive Control - FCRXCTRL (0x00005100)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	6:0	0b	RW	Reserved.
RESERVED	7	1b	RW	Reserved.
RESERVED	11:8	0x0	RW	Reserved.
RESERVED	31:12	0x0	RSV	Reserved.



8.2.4.15.6 FCoE Redirection Control - FCRECTL (0x0000ED00)

Field	Bit(s)	Init.	Access Type	Description
ENA	0	0b	RW	FC Redirection Enable. When cleared, the redirection table is not active. When set to 1b the FC redirection is enabled. Software **Note:** When FC redirection is enabled, the Pool Enable and the Queue Enable in the ETQF and ETQS registers must be cleared for FCoE data packets.
FCRETASEL	1	0b	RW	When the FCRETASEL is cleared, received FCoE packets are directed to receive queues by the FCoE Redirection Table according to the OX_ID. It is the same rule for packets received by the originator as well as the responder. When the FCRETASEL is set (restricted setting option), the packets received by the responder are directed to the receive queues by the RX_ID.
RESERVED	31:2	0x0	RSV	Reserved.

8.2.4.15.7 FCoE Redirection Table - FCRETA[n] (0x0000ED10 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access 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 3 LS bits of the FC exchange ID.
RESERVED	31:7	0x0	RSV	Reserved.

8.2.4.15.8 FC User Descriptor PTR Low - FCPTRL (0x00002410)

Field	Bit(s)	Init.	Access Type	Description
PTR_LOW	31:0	X	RW	User Descriptor PTR Low. 4 LSBs 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.

8.2.4.15.9 FC User Descriptor PTR High - FCPTRH (0x00002414)

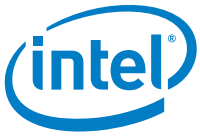
Field	Bit(s)	Init.	Access Type	Description
PTR_HI	31:0	X	RW	User Descriptor PTR High. Four MSBs of the physical pointer to the User Descriptor list.

8.2.4.15.10 FC Buffer Control - FCBUFF (0x00002418)

Field	Bit(s)	Init.	Access Type	Description
VALID	0	0b	RW	DMA Context Valid. When set to 1b indicates that the context is valid. If software clears the Context Valid bit, the software should poll it till 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	0x0	RW	Buffer Size. This field defines the User buffer size used in this context as follows: 00b = 4 KB. 10b = 16 KB. 01b = 8 KB. 11b = 64 KB.
RESERVED	6:5	0x0	RSV	Reserved.
WRCONTX	7	0b	RW	Write DDP Context. This bit should be set to 1b for write exchange context aimed for target (responder) usage. This bit should be set to 0b for read exchange context aimed for initiator (originator) usage.
BUFFCNT	15:8	0x0	RW	Buffer Count. Defines the number of the User Buffer while 0x00 equals 256. 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.

8.2.4.15.11 FC Receive DMA RW - FCDMARW (0x00002420)

Field	Bit(s)	Init.	Access Type	Description
FCOESEL	8:0	0x0	RW	FCoE Context Select. This field defines the FCoE Rx context index (equals the OX_ID for that context).
RESERVED	13:9	0x0	RSV	Reserved.
WE	14	0b	RW	Write Enable. When this bit is set, the content of FCPTRL; FCPTRH and FCBUFF registers are programmed to the FCoE DMA context of index FCoESEL.
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.



Field	Bit(s)	Init.	Access Type	Description
LASTSIZE	31:16	0x0	RW	Last User Buffer Size. Defined the size in bytes of the last user buffer.

8.2.4.15.12 FC FLT Context - FCFLT (0x00005108)

Field	Bit(s)	Init.	Access 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. Init to zero by the driver at context programming.
SEQ_CNT	31:16	X	RW	Sequence Count. The sequence Count of the expected received frame. Init to zero by the driver at context programming.

8.2.4.15.13 FC Offset Parameter - FCPARAM (0x000051D8)

Field	Bit(s)	Init.	Access Type	Description
PARAM	31:0	0x0	RW	FC Parameter. This field contains the expected FC Parameter in the next received frame. Init to 0 by the driver at context programming. **Note:** Field is defined in Big Endian (LS byte is first on the wire).

8.2.4.15.14 FC Filter RW Control - FCFLTRW (0x00005110)

Field	Bit(s)	Init.	Access Type	Description
FCOESEL	8:0	0x0	WO	FCoE context Select. This field defines the FCoE Rx context index (equals the OX_ID for that context).
RESERVED	12:9	0x0	RSV	Reserved.
RE_VALIDATE	13	0b	WO	Fast Re-Validation of the filter context. Setting this bit together with the WE bit in this register validate the selected filter context. Hardware sets the Valid bit and clears the First bit (described in the FCFLT register) while keeping all other filter parameters intact.

Field	Bit(s)	Init.	Access Type	Description
WE	14	0b	WO	Write Enable. When this bit is set, the content of the FCFLT register is 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 register. This bit should never be set together with the WE bit in this register.
RESERVED	31:16	0x0	RSV	Reserved.

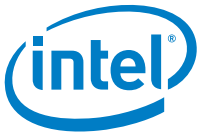
8.2.4.16 Flow Director Registers

Global Settings Registers

8.2.4.16.1 Flow Director Filters Control Register - FDIRCTRL (0x0000EE00)

Note: This register should be configured ONLY as part of the Flow Director init flow or Clearing the Flow Director table. Programming of this register with non-zero value PBALLOC init the Flow Director table.

Field	Bit(s)	Init.	Access Type	Description
PBALLOC	1:0	0x0	RW	Memory allocation for the Flow Director Filters. 00b = No memory allocation - Flow Director Filters are disabled 01b = 64 KB (8 KB minus 1 signature filters or 2 KB minus 1 perfect match filters) 10b = 128 KB (16 KB minus 1 signature filters or 4 KB minus 1 perfect match filters) 11b = 256 KB (32 KB minus 1 signature filters or 8 KB minus 1 perfect match filters)
RESERVED	2	0b	RSV	Reserved.
INIT_DONE	3	0b	RW	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	Flow Director Filters Mode of Operation When set to 1b, hardware supports perfect match filters according to PBALLOC. When cleared to 0b, hardware supports signature filters according to PBALLOC.
REPORT_STATUS	5	0b	RW	Report Flow Director Filter's status in the RSS field of the Rx descriptor for packets that matches a Flow Director filter. Enabling the Flow Director filter's status, the RXCSUM.PCSD bit should be set as well (disabling the fragment checksum). Note that 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.



Field	Bit(s)	Init.	Access Type	Description
REPORT_STATUS_ALWAYS	7	0b	RW	Report Flow Director Status in the RSS field of the Rx descriptor on any packet that can be candidates for the Flow Director filters. This bit can be set to 1 only when both the RXCSUM.PCSD bit and the Report-Status bit in this register are set.
DROP_QUEUE	14:8	0x0	RW	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.
RESERVED	15	0b	RSV	Reserved.
FLEX_OFFSET	20:16	0x0	RW	Offset within the first 64 bytes of the packet of a flexible 2 byte tuple. The offset is defined in word units counted from the first byte of the destination Ethernet MAC Address.
RESERVED	23:21	0x0	RSV	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 the 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 of it reflects the addition of two more filters. **Note:** Software should set this field to a value that indicates exceptional long buckets. Supporting 32 KB 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 is a recommended minimum number of flows that should remain unused (defined in units of 16 filters). When software exceeds this threshold (too low number of unused flows), hardware generates the Flow Director Full interrupt. Software should avoid additional programming following this interrupt. Note that when the Flow Director Filters are used completely, hardware discards silently further filters programming.

8.2.4.16.2 Flow Director Filters Lookup Table HASH Key - FDIRHKEY (0x0000EE68)

Field	Bit(s)	Init.	Access Type	Description
KEY	31:0	0x80000001	RW	Programmable Hash Lookup Ttable Key.

8.2.4.16.3 Flow Director Filters Signature HASH Key - FDIRSKEY (0x0000EE6C)

Field	Bit(s)	Init.	Access Type	Description
KEY	31:0	0x80800101	RW	Programmable Signature Key.

8.2.4.16.4 Flow Director Filters DIPv4 Mask - FDIRDIP4M (0x0000EE3C)

Field	Bit(s)	Init.	Access 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.

8.2.4.16.5 Flow Director Filters Source IPv4 Mask - FDIRSIP4M (0x0000EE40)

Field	Bit(s)	Init.	Access 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.

8.2.4.16.6 Flow Director Filters TCP Mask - FDIRTCPM (0x0000EE44)

Field	Bit(s)	Init.	Access 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). Note that this register is swizzle as follow: bit 0 in the mask affects bit 15 of the source port as defined in FDIRPORT.Source. bit 1 in the mask affects bit 14 in FDIRPORT.Source and so on while bit 15 in the mask affects bit 0 in FDIRPORT.Source.
DPORTM	31:16	0x0	RW	Mask TCP Destination Port. Each cleared bit means that the associated bit of the TCP destination Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the TCP destination Port is ignored (masked out). Note that this register is swizzle the same as the FDIRTCPM.SPortM.

8.2.4.16.7 Flow Director Filters UDP Mask - FDIRUDPM (0x0000EE48)

Field	Bit(s)	Init.	Access 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). Note that this register is swizzle the same as the FDIRTCPM.SPortM.
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). Note that this register is swizzle the same as the FDIRTCPM.SPortM.

8.2.4.16.8 Flow Director Filters IPv6 Mask - FDIRIP6M (0x0000EE74)

Field	Bit(s)	Init.	Access 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.
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.

8.2.4.16.9 Flow Director Filters Other Mask - FDIRM (0x0000EE70)

Note: Global Status / Statistics Registers.

Field	Bit(s)	Init.	Access 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.

Field	Bit(s)	Init.	Access Type	Description
L4P	3	0b	RW	Mask L4 Protocol. When cleared the UDP/TCP/SCTP protocol type is meaningful for the filtering functionality. Note that for the Flow Director filtering aspects, SCTP is treated as if it is TCP.
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.
RESERVED	31:6	0x0	RSV	Reserved.

8.2.4.16.10 Flow Director Filters Free - FDIRFREE (0x0000EE38)

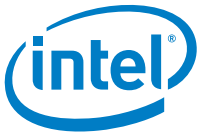
Field	Bit(s)	Init.	Access Type	Description
Reserved	30:16	0x8000	RW	Reserved.

8.2.4.16.11 Flow Director Filters Length - FDIRLEN (0x0000EE4C)

Field	Bit(s)	Init.	Access Type	Description
Reserved	30:16	0x0	RC	Reserved.

8.2.4.16.12 Flow Director Filters Usage Statistics - FDIRUSTAT (0x0000EE50)

Field	Bit(s)	Init.	Access Type	Description
ADD	15:0	0x0	RC	Number Of Added Filters. This field counts the number of added filters to the Flow Director Filters logic. The counter is stacked at 0xFFFF and cleared on read.
REMOVE	31:16	0x0	RC	Number Of Removed Filters. This field counts the number of removed filters to the Flow Director Filters logic. The counter is stacked at 0xFFFF and cleared on read.



8.2.4.16.13 Flow Director Filters Failed Usage Statistics - FDIRFSTAT (0x0000EE54)

Field	Bit(s)	Init.	Access Type	Description
FADD	7:0	0x0	RC	Number of filters addition events that do not change the number of free (non-programmed) filters in the flow director filters logic (FDIRFREE.FREE). These events can be either filters update, filters collision, or tentative of filter additions when there is no sufficient space remaining in the filter table.
FREMOVE	15:8	0x0	RC	Number Of Failed Removed Filters. The counter is stacked at 0xFF and cleared on read.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.16.14 Flow Director Filters Match Statistics - FDIRMATCH (0x0000EE58)

Field	Bit(s)	Init.	Access Type	Description
PCNT	31:0	0x0	RC	Number of packets that matched any Flow Director filter. The counter is stacked at 0xFF...F and cleared on read. Note that this counter might include packets that match the L2 filters or 5 tuple filters or Son filters even if they are enabled for queue assignment.

8.2.4.16.15 Flow Director Filters Miss Match Statistics - FDIRMISS (0x0000EE5C)

Note: Flow Programming Registers.

Field	Bit(s)	Init.	Access Type	Description
PCNT	31:0	0x0	RC	Number of packets that missed matched any Flow Director filter. The counter is stacked at 0xFF...9 F and cleared on read.

8.2.4.16.16 Flow Director Filters Source IPv6 - FDIRSIPV6[n] (0x0000EE0C + 0x4*n, n=0...2)

Field	Bit(s)	Init.	Access Type	Description
IP6SA	31:0	0x0	RW	3 MS DWords of the Source IPv6 address. While the LSB of FDIRSIPv6[0] is first on the wire. The FDIRIPSA contains the LS DWord of the IP6 address while its MS byte is last on the wire.

8.2.4.16.17 Flow Director Filters IP SA - FDIRIPSA (0x0000EE18)

Field	Bit(s)	Init.	Access Type	Description
IP4SA	31:0	0x0	RW	Source IPv4 address or LS DWord of the Source IPv6 address. While the field is defined in Big Endian (LS byte is first on the wire).

8.2.4.16.18 Flow Director Filters IP DA - FDIRIPDA (0x0000EE1C)

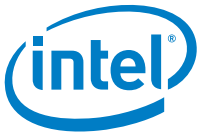
Field	Bit(s)	Init.	Access Type	Description
IP4DA	31:0	0x0	RW	Destination IPv4 address. While the field is defined in Big Endian (LS byte is first on the wire).

8.2.4.16.19 Flow Director Filters Port - FDIRPORT (0x0000EE20)

Field	Bit(s)	Init.	Access Type	Description
SOURCE	15:0	0x0	RW	Source port number while the field is defined in Little Endian (MSB is first on the wire). Note that for SCTP filter the Source and Destination port numbers must be set to zero (while hardware does not check it).
DESTINATION	31:16	0x0	RW	Destination port number while the field is defined in Little Endian (MSB is first on the wire). Note that for SCTP filter the Source and Destination port numbers must be set to zero (while hardware does not check it).

8.2.4.16.20 Flow Director Filters VLAN and FLEX bytes - FDIRVLAN (0x0000EE24)

Field	Bit(s)	Init.	Access Type	Description
VLAN	15:0	0x0	RW	VLAN tag while the field is defined in Little Endian (MSB is first on the wire). The CFI bit must be set to zero while it is not checked by hardware.
FLEX	31:16	0x0	RW	Flexible tuple data as defined by the *Flex-Offset* field in the FDIRCTRL register while the field is defined in Big Endian (LSB is first on the wire).



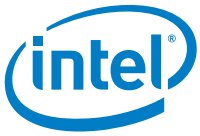
8.2.4.16.21 Flow Director Filters Hash Signature - FDIRHASH (0x0000EE28)

Field	Bit(s)	Init.	Access Type	Description
HASH	14:0	0x0	RW	Bucket hash value that identifies a Filter's linked list.
BUCKET_VALID	15	0b	RW	The *Valid* bit is set by hardware each time there is at least one filter assigned to this hash.
SIGNATURE_SW_INDEX	30:16	0x0	RW	Flow Director Filter Signature for Signature filters and software-index for perfect match filters.
RESERVED	31	0b	RSV	Reserved.

8.2.4.16.22 Flow Director Filters Command register - FDIRCMD (0x0000EE2C)

Field	Bit(s)	Init.	Access Type	Description
CMD	1:0	0x0	RW	Flow Director Filter Programming Command: 0 0b = 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	Valid Filter is found by the query command. This bit is set by the X540 following a Query Command completion.
FILTER_UPDATE	3	0b	RW	Filter Update Command. This bit is relevant only for Add Flow command and must be set to zero in any other commands. When cleared, the filter parameters do not override existing ones if exist while setting only the collision bit. When set to 1b the new filter parameters override existing ones if exist keeping the collision bit as is.
IPV6DMATCH	4	0b	RW	IP Destination match to IP6AT filter. This bit is meaningful only for perfect match IPv6 filters. Otherwise it should be cleared by software at programming time. When set to 1b the Destination IPv6 address should match the IP6AT. When cleared, the Destination IPv6 address should not match the IP6AT. This field might never match local VM to VM traffic.

Field	Bit(s)	Init.	Access Type	Description
L4TYPE	6:5	0x0	RW	L4 Packet Type. Defines the packet as one of the following L4 types: 00b = Reserved. 01b = UDP 10b = TCP 11b = SCTP **Note:** Encoding of the L4TYPE for the Flow Director filters is defined differently than the Protocol type encoding in the FTQF registers for the 128 x 5 tuple filters.
IPV6	7	0b	RW	IPv6 packet type when set to 1b and IPv4 packet type at 0b. Note that the IP type is checked always even if the filters do not check for IP address match.
CLEARHT	8	0b	RW	Clear the Flow Director Head and Tail Registers. This bit is set only as part of Flow Director init. During nominal Operation it must be kept at 0b.
DROP	9	0b	RW	Packet Drop Action. Receive packets that match a filter with an active *Drop* bit and do not exceed the maximum recommended linked list length defined in the FDIRCTRL.Max_Length field are posted to the global queue defined by FDIRCTRL.Drop-Queue. Receive packets that match a filter with an 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 an active FDIRErr(0) flag indicating that the Max_Length was exceeded. The Drop flag is useful only for perfect match filters and it should be cleared by software for signature filters. When the *Drop* bit is set, the Queue-EN flag must be set and Rx-Queue in this register must be valid as well. Otherwise, the result is unexpected.
INT	10	0b	RW	Matched packet generates a Low Latency Interrupt.
LAST	11	0b	RW	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 one when software programs the same multiple times. In Signature based filtering it is set when software programs a filter with the same Hash and Signature multiple times. It should be cleared by software when it adds a Flow. It might be set by hardware when 2 flows collide with the same Hash and signature. During reception, this bit is reported on the Rx descriptor of packets that match the filter. See bit 7 for description of the Query-Type.
RESERVED	14:13	0x0	RSV	Reserved.
QUEUE_EN	15	0b	RW	Enable routing matched packet to the queue defined by the Rx-Queue. Note that 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).
RESERVED	23	0b	RSV	Reserved.



Field	Bit(s)	Init.	Access Type	Description
POOL	29:24	0x0	RW	Pool number is meaningful when VT mode is enabled. When both VT is not enabled, this field must be set by software to 0x0.
RESERVED	31:30	0x0	RSV	Reserved.

8.2.4.17 MAC Registers

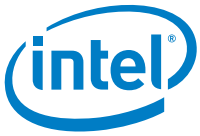
8.2.4.17.1 Core Control 0 Register - HLREG0 (0x00004240)

Field	Bit(s)	Init.	Access Type	Description
TXCRCEN	0	1b	RW	Tx CRC ENABLE: Enables a CRC to be appended by hardware to a Tx Packet if requested by user 1b = Enable CRC by hardware (default). 0b = No CRC appended, packets always passed unchanged.
RXCRCSTRP	1	1b	RW	Rx CRC STRIP. Causes the CRC to be stripped by hardware from all packets. TheRDRXCTL.CRCStrip must be set the same as this bit. 1b = Strip CRC by hardware (default). 0b = No CRC Strip by hardware.
JUMBOEN	2	0b	RW	Jumbo Frame Enable. Allows frames up to the size specified in Reg MAXFRS (31:16) 1b = Enable jumbo frames. 0b = Disable jumbo frames (default).
RESERVED	9:3	0x7F	RSV	Reserved. must be set to 1111111b.
TXPADEN	10	1b	RW	Tx Pad Frame Enable. Pad short Tx frames to 64 bytes if requested by the user. 1b = Pad frames (default). 0b = Transmit short frames with no padding.
RESERVED	11	1b	RSV	Reserved. must be set to 1b.
RESERVED	12	0b	RW	Reserved.
RESERVED	13	1b	RSV	Reserved. must be set to 1b.
RESERVED	14	0b	RW	Reserved.
LPBK	15	0b	RW	Loopback. Turn On Loopback Where Transmit Data Is Sent Back Through Receiver. 1b = Loopback enabled. 0b = Loopback disabled (default).

Field	Bit(s)	Init.	Access Type	Description
MDCSPD	16	1b	RW	MDC Speed. High Or Low Speed MDC Clock Frequency To PCS, XGXS, WIS, etc. MDCSPD Freq at 10 Gb/s, Freq at 1 Gb/s, Freq at 100 Mb/s, respectively: 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. 1b = Continuous MDC. 0b = MDC off between packets (default).
RESERVED	19:18	0b	RW	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 = 0 hex).
RESERVED	24	0b	RSV	Reserved.
RESERVED	26:25	0x0	RSV	Reserved.
RXLNGTHERREN	27	1b	RW	Rx Length Error Reporting. 1b = Enable reporting of rx_length_err events if length field < 0x0600. 0b = Disable reporting of all rx_length_err events.
RXPADSTRIPEN	28	0b	RW	Rx Padding Strip Enable. 1b = Reserved. 0b = Do not strip padding from Rx packets with length field < 64 (default).
RESERVED	31:29	0x0	RSV	Reserved.

8.2.4.17.2 Core Status 1 Register - HLREG1 (0x00004244)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	3:0	0x1	RSV	Reserved.
RESERVED	4	0b	RSV	Reserved.
RXERRSYM	5	0b	RO	Rx Error Symbol. Error symbol during Rx packet (latch high, clear on read). 1b = Error symbol received. 0b = No error symbol (default).
RXILLSYM	6	0b	RO	Rx Illegal Symbol. Illegal symbol during Rx packet (latch high, clear on read). 1b = Illegal symbol received. 0b = No illegal symbol received (default).



Field	Bit(s)	Init.	Access Type	Description
RXIDLERR	7	0b	RO	Rx Idle Error. Non-idle symbol during idle period (latch high, clear on read). 1b = Idle error received. 0b = No idle errors received (default).
RXLCLFLT	8	0b	RO	Rx Local Fault. Fault reported from PMD, PMA, or PCS (latch high, clear on read). 1b = Local fault is or was active. 0b = No local fault (default).
RXRMTFLT	9	0b	RO	Rx Remote Fault. Link partner reported remote fault (latch high, clear on read). 1b = Remote fault is or was active. 0b = No remote fault (default).
RESERVED	31:10	0x0	RSV	Reserved.

8.2.4.17.3 Pause and Pace Register - PAP (0x00004248)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	15:0	0xFFFF	RW	Reserved
PACE	19:16	0x0	RW	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. 1111b = 9.294196 Gb/s (WAN).
RESERVED	31:20	0x0	RSV	Reserved.

8.2.4.17.4 MDI Single Command and Address - MSCA (0x0000425C)

Field	Bit(s)	Init.	Access Type	Description
MDIADD	15:0	0x0000	RW	MDI Address. Address used For MDI accesses. (Default = 0000 hex).
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.

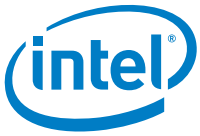
Field	Bit(s)	Init.	Access Type	Description
OPCODE	27:26	0x0	RW	OP Code. Two bits identifying operation to be performed (default - 00b). 00b = Address cycle. 01b = Write operation. 10b = Read, increment address. 11b = Read operation.
RESERVED	29:28	0x0	RSV	Reserved. Reads as 00b. Must be written as 00b.
MDICMD	30	0b	RW	MDI Command. Perform the MDI operation in this register; cleared when done. 1b = Perform operation, operation in progress. 0b = MDI ready, operation complete (default).
RESERVED	31	0b	RSV	Reserved.

8.2.4.17.5 MDI Single Read and Write Data - MSRWD (0x00004260)

Field	Bit(s)	Init.	Access Type	Description
MDIWRDATA	15:0	0x0	RW	MDI Write Data.
MDIRDDATA	31:16	0x0	RW	MDI Read Data (RO).

8.2.4.17.6 Max Frame Size - MAXFRS (0x00004268)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	15:0	0x0	RSV	Reserved.
MFS	31:16	0x5EE	RW	This field defines the Maximum Frame Size in bytes units from Ethernet MAC Addresses up to inclusive the CRC. Frames received that are larger than this value are dropped. This field is meaningful when Jumbo frames are enabled (HLREG0.JUMBOEN = 1). When Jumbo frames are not enabled the X540 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 packets. 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.



8.2.4.17.7 Link Status Register - LINKS (0x000042A4) MAC

Field	Bit(s)	Init.	Access Type	Description
FIFO_UNDERRUN	0	0b	RO	Indicates under-run condition in MAC elastic FIFO.
FIFO_OVERRUN	1	0b	RO	Indicates over-run condition in MAC elastic FIFO.
RF_STATE	2	0b	RO	MAC is in remote fault state.
LF_STATE	3	0b	RO	MAC is in local fault state.
RESERVED	6:4	0x0	RSV	Reserved.
LINK_STATUS	7	0b	RO	1b = Link is up, and there was no link down from last time read. 0b = Link is currently down or link was down since last time read. Self cleared upon read if the link is low and set if the link is up.
RESERVED	27:8	0x0	RSV	Reserved.
LINK_SPEED	29:28	0x0	RO	MAC Link Speed Status. 00b = Reserved. 01b = 100 Mb/s. 10b = 1 Gb/s. 11b = 10 Gb/s.
LINK_UP	30	0b	RO	Link up 1b = Link is up. 0b = Link is down.
RESERVED	31	0b	RSV	Reserved.

8.2.4.17.8 MAC Manageability Control Register - MMNGC (0x000042D0)

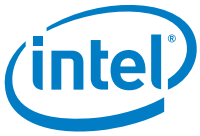
Field	Bit(s)	Init.	Access Type	Description
MNG_VETO	0	0b	RO	MNG_VETO (default 0b) access read/write by manageability, read only to the host. 0b = No specific constraints on link from manageability. 1b = Hold off any low-power link mode changes. This is done to avoid link loss and interrupting manageability activity. This register is RW for MNG, and RO for host accesses.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.4.17.9 TabMAC Flow Control Register - MFLCN (0x00004294)

Field	Bit(s)	Init.	Access Type	Description
PMCF	0	0b	RW	Pass MAC Control Frames. Filter out unrecognized pause (flow control opcode doesn't match) and other control frames. 0b = Filter unrecognized pause frames. 1b = Pass/forward unrecognized pause frames.
DPF	1	0b	RW	Discard Pause Frame. When set to 0b, Pause frames are sent to the host. Setting this bit to 1b causes PAUSE frames to be discarded only when RFCE or 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 X540 responds to the reception of Priority Flow Control packets. If auto negotiation is enabled this bit should be set by software to the negotiated fow 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. **Note:** Priority Flow control should be enabled in DCB mode only. **Note:** Receive Priority Flow Control and Receive Link Flow Control are mutually exclusive and user should not configure both of them to be enabled at the same time. **Note:** This bit should not be set if bit 3 is set.
RFCE	3	0b	RW	Receive Link Flow Control Enable. Indicates that the X540 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 set, the X540 responds to the reception of Priority Flow Control packets for UPn. When bit n is cleared, Priority Flow Control indication received for UPn is ignored and transmit from UPn is not paused.
RESERVED	31:12	0x0	RSV	Reserved.

8.2.4.17.10 MAC Control register - MACC (0x00004330)

Field	Bit(s)	Init.	Access 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	Enable MAC Rx to Tx loopback. 0b = No loopback, normal mode. 1b = Loopback enabled. Transmit path is driven from the receive path, at the MAC internal XGMII interface.



Field	Bit(s)	Init.	Access Type	Description
RESERVED	7:2	0b	RW	Reserved.
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.

8.2.4.18 Statistic 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 it should be noted that a packet is indicated as received if it passes the device filters and is placed into the packet buffer memory. A packet does not have to be DMA'd to host memory in order to be counted as received.
- Due to divergent paths between interrupt-generation and logging of relevant statistics counts, it might be possible to generate an interrupt to the system for a noteworthy event prior to the associated statistics count actually being incremented. This is extremely unlikely due to expected delays associated with the system interrupt-collection and ISR delay, but might be an explanation for interrupt statistics values that do not quite make sense. Hardware guarantees that any event noteworthy of inclusion in a statistics count is reflected in the appropriate count within 1 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 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 Statistics Hierarchy:

8.2.4.18.1 CRC Error Count - CRCERRS (0x00004000)

Note: A packet counted by the ILLERRC or the ERRBC statistics counter is not counted by the CRCERRS counter.

Field	Bit(s)	Init.	Access Type	Description
CEC	31:0	0x0	RC	CRC Error Count. Counts the number of receive packets with CRC errors. In order for a packet to be counted in this register, it must be 64 bytes or greater (from <Destination Address> through <CRC>, inclusively) in length. This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.4.18.2 Illegal Byte Error Count - ILLERRC (0x00004004)

Field	Bit(s)	Init.	Access 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.

8.2.4.18.3 Error Byte Packet Count - ERRBC (0x00004008)

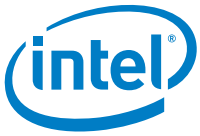
Field	Bit(s)	Init.	Access Type	Description
EBC	31:0	0x0	RC	Error Byte Packet Count. Counts the number of receive packets with Error bytes (such as, there is an Error symbol in the packet). This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.4.18.4 MAC Local Fault Count - MLFC (0x00004034)

Field	Bit(s)	Init.	Access Type	Description
MLFC	31:0	0x0	RC	Number of faults in the local MAC. This register is valid only when the link speed is 10 Gb/s.

8.2.4.18.5 MAC Remote Fault Count - MRFC (0x00004038)

Field	Bit(s)	Init.	Access Type	Description
MRFC	31:0	0x0	RC	Number of faults in the remote MAC. This register is valid only when the link speed is 10 Gb/s.



8.2.4.18.6 Receive Length Error Count - RLEC (0x00004040)

Field	Bit(s)	Init.	Access Type	Description
RLEC	31:0	0x0	RC	Number of packets with receive length errors. A length error occurs if an incoming packet length field in the MAC header doesn't match the packet length. To enable the receive length error count HLREG.RXLNGTHERREN bit needs to be set to 1. This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.4.18.7 Switch Security Violation Packet Count - SSVPC (0x00008780)

Field	Bit(s)	Init.	Access Type	Description
SSVPC	31:0	0x0	RC	Switch Security Violation Packet Count. This register counts all Tx packets dropped. For example, 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. Packets which are not routed to LAN nor to BMC are also counted in this register.

8.2.4.18.8 Link XON Received Count - LXONRXCNT (0x000041A4)

Field	Bit(s)	Init.	Access Type	Description
XONRXC	15:0	0x0	RC	Number of XON packets Received. Sticks to 0xFFFF. XON packets can use the global address, or the station address. This register counts any XON packet whether it is a legacy XON or a priority XON. Each XON packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only the LXOFFRXCNT counter is incremented.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.18.9 Link XOFF Received Count - LXOFFRXCNT (0x000041A8)

Field	Bit(s)	Init.	Access Type	Description
XOFFRXC	15:0	0x0	RC	Number of XOFF packets Received. Sticks to 0xFFFF. XOFF packets can use the global address, or the station address. This register counts any XOFF packet whether it is a legacy XOFF or a priority XOFF. Each XOFF packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only this counter is incremented.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.18.10 Priority XON Received Count - PXONRXCNT[n] (0x00004140 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
XONRXC	15:0	0x0	RC	Number of XON packets received per UP. Sticks to 0xFFFF.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.18.11 Priority XOFF Received Count - PXOFFRXCNT[n] (0x00004160 + 0x4*n, n=0...7)

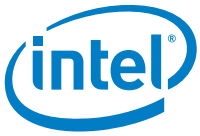
Field	Bit(s)	Init.	Access Type	Description
XOFFRXC	15:0	0x0	RC	Number of XOFF packets received per UP. Sticks to 0xFFFF.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.18.12 Packets Received [64 Bytes] Count - PRC64 (0x0000405C)

Field	Bit(s)	Init.	Access Type	Description
PRC64	31:0	0x0	RW	Number of good packets received that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.

8.2.4.18.13 Packets Received [65-127 Bytes] Count - PRC127 (0x00004060)

Field	Bit(s)	Init.	Access Type	Description
PRC127	31:0	0x0	RW	Number of packets received that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.



8.2.4.18.14 Packets Received [128-255 Bytes] Count - PRC255 (0x00004064)

Field	Bit(s)	Init.	Access Type	Description
PRC255	31:0	0x0	RW	Number of packets received that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.

8.2.4.18.15 Packets Received [256-511 Bytes] Count - PRC511 (0x00004068)

Field	Bit(s)	Init.	Access Type	Description
PRC511	31:0	0x0	RW	Number of packets received that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.

8.2.4.18.16 Packets Received [512-1023 Bytes] Count - PRC1023 (0x0000406C)

Field	Bit(s)	Init.	Access Type	Description
PRC1023	31:0	0x0	RW	Number of packets received that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.

8.2.4.18.17 Packets Received [1024 to Max Bytes] Count - PRC1522 (0x00004070)

Field	Bit(s)	Init.	Access Type	Description
PRC1522	31:0	0x0	RW	Number of packets received that are 1024-Max bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets. The maximum is dependent on the current receiver configuration and the type of packet being received. If a packet is counted in Receive Oversized Count, it is not counted in this register (see Receive Oversize Count - ROC). 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.

8.2.4.18.18 Broadcast Packets Received Count - BPRC (0x00004078)

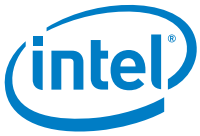
Field	Bit(s)	Init.	Access Type	Description
BPRC	31:0	0x0	RC	Number of good (non-erred) broadcast packets received. This register does not count received broadcast packets when the broadcast address filter is disabled. The counter counts packets regardless on receive enablement.

8.2.4.18.19 Multicast Packets Received Count - MPRC (0x0000407C)

Field	Bit(s)	Init.	Access Type	Description
MPRC	31:0	0x0	RC	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 on receive enablement.

8.2.4.18.20 Good Packets Received Count - GPRC (0x00004074)

Field	Bit(s)	Init.	Access Type	Description
GPRC	31:0	0x0	RC	Number of good (non-erred) Rx packets (from the network) that pass L2 filtering and has legal length as defined by Long Packet Enable. This register counts packets regardless on receive enablement.



8.2.4.18.21 Good Octets Received Count Low - GORCL (0x00004088)

Field	Bit(s)	Init.	Access Type	Description
CNT_L	31:0	0x0	RC	Lower 32 bits of the Good Octets Received counter. The GORCL and GORCH registers make up a logical 36-bit octet counter of the packets counted by the GPRC. This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively.

8.2.4.18.22 Good Octets Received Count High - GORCH (0x0000408C)

Field	Bit(s)	Init.	Access Type	Description
CNT_H	3:0	0x0	RC	Higher 4 bits of the Good Octets Received counter.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.18.23 Good Rx Non-Filtered Packet Counter - RXNFGPC (0x000041B0)

Field	Bit(s)	Init.	Access Type	Description
GPC	31:0	0x0	RC	Number of good (non-erred with legal length) Rx packets (from the network) regardless of packet filtering and receive enablement.

8.2.4.18.24 Good Rx Non-Filter Byte Counter Low - RXNFGBCL (0x000041B4)

Field	Bit(s)	Init.	Access Type	Description
BCL	31:0	0x0	RC	Low 32 bits of the 36 bit Byte Counter of good (non-erred) Rx packets that match the RXNFGPC. The counter counts all bytes from <Destination Address> field through the <CRC> field, inclusively.

8.2.4.18.25 Good Rx Non-Filter Byte Counter High - RXNFGBCH (0x000041B8)

Field	Bit(s)	Init.	Access Type	Description
BCH	3:0	0x0	RC	High 4 bits of the 36-bit Byte Counter associated to RXFGBCL.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.18.26 DMA Good Rx Packet Counter - RXDGPC (0x00002F50)

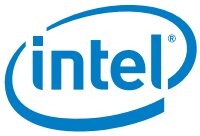
Field	Bit(s)	Init.	Access Type	Description
GPC	31:0	0x0	RC	Number of good (non-erred) Rx packets from the Network posted to the host memory. In case of packet replication (or mirrored) the counter counts each packet only once. The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register.

8.2.4.18.27 DMA Good Rx Byte Counter Low - RXDGBCL (0x00002F54)

Field	Bit(s)	Init.	Access Type	Description
GBCL	31:0	0x0	RC	Low 32 bits of the 36 bit Byte Counter of good (non-erred) Rx packets that match the RXDGPC. The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP.

8.2.4.18.28 DMA Good Rx Byte Counter High - RXDGBCH (0x00002F58)

Field	Bit(s)	Init.	Access Type	Description
GBCH	3:0	0x0	RC	High 4 bits of the 36 bit Byte Counter associated to RXDGBCL.
RESERVED	31:4	0x0	RSV	Reserved.



8.2.4.18.29 DMA Duplicated Good Rx Packet Counter - RXDDPC (0x00002F5C)

Field	Bit(s)	Init.	Access Type	Description
GPC	31:0	0x0	RC	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.

8.2.4.18.30 DMA Duplicated Good Rx Byte Counter Low - RXDDBCL (0x00002F60)

Field	Bit(s)	Init.	Access Type	Description
GBCL	31:0	0x0	RC	Low 32 bits of the 36 bit Byte Counter of good (non-erred) Rx packets that match the RXDDPC. The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP.

8.2.4.18.31 DMA Duplicated Good Rx Byte Counter High - RXDDBCH (0x00002F64)

Field	Bit(s)	Init.	Access Type	Description
GBCH	3:0	0x0	RC	High 4 bits of the 36 bit Byte Counter associated to RXDDBCL.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.18.32 DMA Good Rx LPBK Packet Counter - RXLPBKPC (0x00002F68)

Field	Bit(s)	Init.	Access Type	Description
GPC	31:0	0x0	RC	Number of good (non-erred) Rx packets from a local VM posted to the host memory. In case of packet replication (or mirrored) the counter counts each packet only once. The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register. The counter is not affected by RSC and FCoE DDP since both functions are not supported for LPBK traffic.

8.2.4.18.33 DMA Good Rx LPBK Byte Counter Low - RXLPBKBCL (0x00002F6C)

Field	Bit(s)	Init.	Access Type	Description
GBCL	31:0	0x0	RC	Low 32 bits of the 36-bit Byte Counter of good (non-erred) Rx packets that match the RXLPBKPC. The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP.

8.2.4.18.34 DMA Good Rx LPBK Byte Counter High - RXLPBKBCH (0x00002F70)

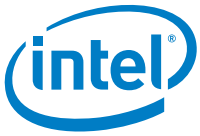
Field	Bit(s)	Init.	Access Type	Description
GBCH	3:0	0x0	RC	High 4 bits of the 36-bit Byte Counter associated to RXLPBKBCL.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.18.35 DMA Duplicated Good Rx LPBK Packet Counter - RXDLPBKPC (0x00002F74)

Field	Bit(s)	Init.	Access Type	Description
GPC	31:0	0x0	RC	Number of replicated or mirrored packets that meet the RXLPBKPC conditions. The sum of RXDLPBKPC and RXLPBKPC is the total good (non-erred) Rx packets from a local VM posted to the host. **Note:** The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register.

8.2.4.18.36 DMA Duplicated Good Rx LPBK Byte Counter Low - RXDLPBKBCL (0x00002F78)

Field	Bit(s)	Init.	Access Type	Description
GBCL	31:0	0x0	RC	Low 32 bits of the 36-bit Byte Counter of good (non-erred) Rx packets that match the RXDLPBKPC. The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP.



8.2.4.18.37 DMA Duplicated Good Rx LPBK Byte Counter High - RXDLPBKBCH (0x00002F7C)

Field	Bit(s)	Init.	Access Type	Description
GBCH	3:0	0x0	RC	High 4 bits of the 36-bit Byte Counter associated to RXDLPBKBCL.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.18.38 Good Packets Transmitted Count - GPTC (0x00004080)

Field	Bit(s)	Init.	Access Type	Description
GPTC	31:0	0x0	RC	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.

8.2.4.18.39 Good Octets Transmitted Count Low - GOTCL (0x00004090)

Field	Bit(s)	Init.	Access Type	Description
CNT_L	31:0	0x0	RC	Lower 32 bits of the Good Octets Transmitted counter. See complete description in the next register — GOTCH.

8.2.4.18.40 Good Octets Transmitted Count High - GOTCH (0x00004094)

Field	Bit(s)	Init.	Access Type	Description
CNT_H	3:0	0x0	RC	Higher 4 bits of the Good Octets Transmitted counter. The GOTCL and GOTCH registers make up a logical 36-bit counter of successfully transmitted octets (in packets counted by GPTC). This register includes transmitted bytes in a packet from the <Destination Address> field through the <CRC> field, inclusively.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.18.41 DMA Good Tx Packet Counter - TXDGPC (0x000087A0)

Field	Bit(s)	Init.	Access Type	Description
GPTC	31:0	0x0	RC	Number of Tx packets from the 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 case can happen only in VT mode when the local switch is enabled.Dropped packets counted in SSVPC register are not counted here.

8.2.4.18.42 DMA Good Tx Byte Counter Low - TXDGBCL (0x000087A4)

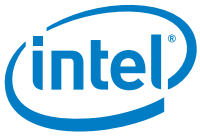
Field	Bit(s)	Init.	Access Type	Description
BCL	31:0	0x0	RC	Low 32 bits of the 36-bit Byte Counter of the Tx packets that match the TXDGPC. The counter counts all bytes posted by the host AND the VLAN (in case bytes added by hardware).Dropped packets counted in SSVPC register are not counted here.

8.2.4.18.43 DMA Good Tx Byte Counter High - TXDGBCH (0x000087A8)

Field	Bit(s)	Init.	Access Type	Description
BCH	3:0	0x0	RC	High 4 bits of the 36-bit Byte Counter associated to TXDGBCL.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.18.44 Receive Undersize Count - RUC (0x000040A4)

Field	Bit(s)	Init.	Access 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.



8.2.4.18.45 Receive Fragment Count - RFC (0x000040A8)

Field	Bit(s)	Init.	Access Type	Description
RFC	31:0	0x0	RC	Number of receive fragment errors (frame shorted than 64 bytes from <Destination Address> through <CRC>, inclusively) that have bad CRC (this is slightly different from the Receive Undersize Count register). This register counts packets regardless of L2 filtering and receive enablement.

8.2.4.18.46 Receive Oversize Count - ROC (0x000040AC)

Field	Bit(s)	Init.	Access 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.

8.2.4.18.47 Receive Jabber Count - RJC (0x000040B0)

Field	Bit(s)	Init.	Access Type	Description
RJC	31:0	0x0	RC	Number of receive jabber errors. This register counts the number of received packets regardless of L2 filtering and receive enablement, and are greater than maximum size and have bad CRC (this is slightly different from the Receive Oversize Count register). The packets length is counted from <Destination Address> through <CRC>, inclusively. This register counts packets regardless of L2 filtering and receive enablement.

8.2.4.18.48 Management Packets Received Count - MNGPRC (0x000040B4)

Field	Bit(s)	Init.	Access Type	Description
MNGPRC	31:0	0x0	RO	Number of management packets received. This register counts the total number of packets received that pass the management filters Management packets include RMCP and ARP packets. Any packets with errors are not counted, except that packets dropped because the management receives a jumbo packet or because the receive FIFO is full are counted.

8.2.4.18.49 Management Packets Dropped Count - MNGPDC (0x000040B8)

Field	Bit(s)	Init.	Access Type	Description
MPDC	31:0	0x0	RO	Number of management packets dropped. This register counts the total number of packets received that pass the management filters and then are dropped because the management receive FIFO is full or because a jumbo packet is received. Management packets include any packet directed to the manageability console (such as RMCP and ARP packets).

8.2.4.18.50 Total Octets Received Low - TORL (0x000040C0)

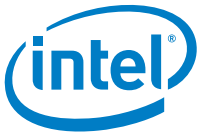
Field	Bit(s)	Init.	Access Type	Description
CNT_L	31:0	0x0	RC	Lower 32 bits of the Total Octets Received counter. See complete description in the next register — TORH.

8.2.4.18.51 Total Octets Received High - TORH (0x000040C4)

Field	Bit(s)	Init.	Access Type	Description
CNT_H	3:0	0x0	RC	Higher 4 bits of the Total Octets Received counter. The TORL and TORH registers make up a logical 36-bit counter of the total received octets (in the packets counted by the TPR counter). This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.18.52 Total Packets Received - TPR (0x000040D0)

Field	Bit(s)	Init.	Access Type	Description
TPR	31:0	0x0	RC	Number of all 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 enablement but excluding Flow Control packets.The TPR might count packets interrupted by link disconnect although they have a CRC error.



8.2.4.18.53 Total Packets Transmitted - TPT (0x000040D4)

Field	Bit(s)	Init.	Access Type	Description
TPT	31:0	0x0	RC	Number of all packets transmitted. This register counts the total number of all packets transmitted. This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets.

8.2.4.18.54 Packets Transmitted (64 Bytes) Count - PTC64 (0x000040D8)

Field	Bit(s)	Init.	Access Type	Description
PTC64	31:0	0x0	RC	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.

8.2.4.18.55 Packets Transmitted [65-127 Bytes] Count - PTC127 (0x000040DC)

Field	Bit(s)	Init.	Access Type	Description
PTC127	31:0	0x0	RC	Number of packets transmitted that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, and manageability packets.

8.2.4.18.56 Packets Transmitted [128-255 Bytes] Count - PTC255 (0x000040E0)

Field	Bit(s)	Init.	Access Type	Description
PTC255	31:0	0x0	RC	Number of packets transmitted that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, and manageability packets.

8.2.4.18.57 Packets Transmitted [256-511 Bytes] Count - PTC511 (0x000040E4)

Field	Bit(s)	Init.	Access Type	Description
PTC511	31:0	0x0	RC	Number of packets transmitted that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, and manageability packets.

8.2.4.18.58 Packets Transmitted [512-1023 Bytes] Count - PTC1023 (0x000040E8)

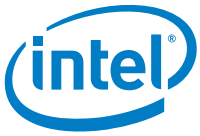
Field	Bit(s)	Init.	Access Type	Description
PTC1023	31:0	0x0	RC	Number of packets transmitted that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, and manageability packets.

8.2.4.18.59 Packets Transmitted [Greater than 1024 Bytes] Count - PTC1522 (0x000040EC)

Field	Bit(s)	Init.	Access Type	Description
PTC1522	31:0	0x0	RC	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.

8.2.4.18.60 Multicast Packets Transmitted Count - MPTC (0x000040F0)

Field	Bit(s)	Init.	Access Type	Description
MPTC	31:0	0x0	RC	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.



8.2.4.18.61 Broadcast Packets Transmitted Count - BPTC (0x000040F4)

Field	Bit(s)	Init.	Access Type	Description
BPTC	31:0	0x0	RC	Number of broadcast packets transmitted count. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets

8.2.4.18.62 MAC Short Packet Discard Count - MSPDC (0x00004010)

Field	Bit(s)	Init.	Access Type	Description
MSPDC	31:0	0x0	RC	Number of MAC short Packet Discard packets received.

8.2.4.18.63 XSUM Error Count - XEC (0x00004120)

Note: XSUM errors are not counted when a packet has any MAC error (CRC, length, under-size, over-size, byte error or symbol error).

Field	Bit(s)	Init.	Access Type	Description
XEC	31:0	0x0	RC	Number of Receive IPv4, TCP, UDP or SCTP XSUM Errors.

8.2.4.18.64 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; 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].

Note: 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.	Access Type	Description
Q_MAP_0	3:0	0x0	RW	For each register 'n', Q_MAP[0] defines the per queue statistic registers that are mapped to Rx queue '4*n+0'. (see examples that follow).

Field	Bit(s)	Init.	Access Type	Description
RESERVED	7:4	0x0	RSV	Reserved.
Q_MAP_1	11:8	0x0	RW	For each register 'n', Q_MAP[1] defines the per queue statistic registers that are mapped to Rx queue '4*n+1'. (see examples below)
RESERVED	15:12	0x0	RSV	Reserved.
Q_MAP_2	19:16	0x0	RW	For each register 'n', Q_MAP[2] defines the per queue statistic registers that are mapped to Rx queue '4*n+2'. (see examples below)
RESERVED	23:20	0x0	RSV	Reserved.
Q_MAP_3	27:24	0x0	RW	For each register 'n', Q_MAP[3] defines the per queue statistic registers that are mapped to Rx queue '4*n+3'. (see examples below)
RESERVED	31:28	0x0	RSV	Reserved.

8.2.4.18.65 Rx DMA Statistic Counter Control - RXDSTATCTRL (0x00002F40)

Field	Bit(s)	Init.	Access Type	Description
QSEL	4:0	0x0	RW	The Queue Select field controls which Rx queues are considered for the DMA Good Rx and DMA Duplicated counters as follows: 00000... 01111 - The counters relate to the same queues that are directed to the QPRC[QSEL] counter as defined by the RQSMR[n] registers. 10000 - The counter relates to all Rx queues; else, reserved.
RESERVED	31:5	0x0	RSV	Reserved.

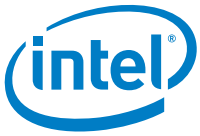
8.2.4.18.66 Transmit Queue Statistic Mapping Registers - TQSM[n] (0x00007300 + 0x4*n, n=0...7; RW)

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; QBTC.

Fields definitions are the same as defined on Section 8.2.4.18.67.

8.2.4.18.67 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; QBTC.



Field	Bit(s)	Init.	Access Type	Description
Q_MAP_0	3:0	0x0	RW	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	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	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	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.

8.2.4.18.68 Queue Packets Received Count - QPRC[n] (0x00001030 + 0x40*n, n=0...15)

Field	Bit(s)	Init.	Access Type	Description
PRC	31:0	0x0	RC	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).

8.2.4.18.69 Queue Packets Received Drop Count - QPRDC[n] (0x00001430 + 0x40*n, n=0...15)

Field	Bit(s)	Init.	Access Type	Description
PRDC	31:0	0x0	RC	Number of receive packets dropped for the queue. Packets are dropped per queue in one of three cases: 1. Rx queue is disabled in the RXDCTL[n] register. 2. No free descriptors in the Rx queue while hardware is set to Drop En in the SRRCTL[n] register or in the PFQDE register. 3. Packet size is larger than RLPML while RLPML_EN is set in the RXDCTL[n] register.

8.2.4.18.70 Queue Bytes Received Count Low - QBRC_L[n] (0x00001034 + 0x40*n, n=0...15)

Field	Bit(s)	Init.	Access Type	Description
BRC_L	31:0	0x0	RC	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.

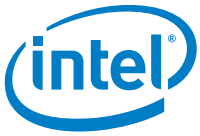
8.2.4.18.71 Queue Bytes Received Count High - QBRC_H[n] (0x00001038 + 0x40*n, n=0...15)

Field	Bit(s)	Init.	Access Type	Description
BRC_H	3:0	0x0	RC	Higher 4 bits of the statistic counter described in QBRC_L.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.18.72 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.	Access Type	Description
PTC	31:0	0x0	RC	Number of packets transmitted for the Queue. 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 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.



8.2.4.18.73 Queue Bytes Transmitted Count Low - QBTC_L[n] (0x00008700 + 0x8*n, n=0...15)

Field	Bit(s)	Init.	Access Type	Description
BTC_L	31:0	0x0	RC	Lower 32 bits of the statistic counter. The QBTC_L and QBTC_H registers make up a logical 36-bit counter of transmitted bytes of the packets counted by the matched QPTC counter. These registers count all bytes in the packets from the <Destination Address> field through the <CRC> field, inclusively. These registers must be accessed as two consecutive 32bit entities while QBTC_L register is read first, or a single 64bit read cycle. Each register is Read cleared. In addition, it sticks at 0xFF...F to avoid overflow.

8.2.4.18.74 Queue Bytes Transmitted Count High - QBTC_H[n] (0x00008704 + 0x8*n, n=0...15)

Field	Bit(s)	Init.	Access Type	Description
BTC_H	3:0	0x0	RC	Higher 4 bits of the statistic counter described in QBTC_L.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.18.75 FC CRC Error Count - FCCRC (0x00005118)

Field	Bit(s)	Init.	Access Type	Description
CRC_CNT	15:0	0x0	RC	FC CRC Count. Count the number of packets with good Ethernet CRC and bad FC CRC.
RESERVED	31:16	X	RSV	Reserved.

8.2.4.18.76 FCoE Rx Packets Dropped Count - FCOERPDC (0x0000241C)

Field	Bit(s)	Init.	Access Type	Description
RPDC	31:0	0x0	RC	Number of Rx packets dropped due to lack of descriptors.

8.2.4.18.77 FC Last Error Count - FCLAST (0x00002424)

Field	Bit(s)	Init.	Access Type	Description
LAST_CNT	15:0	0x0	RC	Number of packets received to valid FCoE contexts while their user buffers are exhausted.
RESERVED	31:16	X	RSV	Reserved.

8.2.4.18.78 FCoE Packets Received Count - FCOEPRC (0x00002428)

Field	Bit(s)	Init.	Access Type	Description
PRC	31:0	0x0	RC	Number of FCoE packets posted to the host. In nominal operation (no save bad frames) it equals to the number of good packets.

8.2.4.18.79 FCoE Packets Transmitted Count - FCOEPTC (0x00008784)

Field	Bit(s)	Init.	Access Type	Description
PTC	31:0	0x0	RC	Number of FCoE packets transmitted. Note that the counter does not include packets dropped due to anti-spoofing filtering or VLAN tag validation as described in Outbound Security. This rule is applicable if FCoE traffic is sent by a VF.

8.2.4.18.80 FCoE DWord Transmitted Count - FCOEDWTC (0x00008788)

Field	Bit(s)	Init.	Access Type	Description
DWTC	31:0	0x0	RC	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).

8.2.4.18.81 BMC2OS Packets Sent by BMC - B2OSPC (0x000041C0)

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 X540 due to congestion conditions.

The counter is cleared when read by the software device driver. The counter is also cleared by PCIe reset and software reset. When reaching maximum value counter does not wrap-around.

Field	Bit(s)	Init.	Access Type	Description
B2OSPC	31:0	0x0	RC	BMC2OS packets sent by BMC.

8.2.4.18.82 BMC2OS Packets Received by Host - B2OGPRC (0x00002F90)

This register counts the total number of packets originating from the BMC that were 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 software device driver. The counter is also cleared by PCIe reset and Software reset. When reaching maximum value counter does not wrap-around.

Field	Bit(s)	Init.	Access Type	Description
B2OGPRC	31:0	0x0	RC	BMC2OS packets received by host.

8.2.4.18.83 TabOS2BMC packets received by BMC - O2BGPTC (0x000041C4)

This register counts the total number of packets originating from the host that reached the NC-SI interface. The counter is cleared when read by software device driver. The counter is also cleared by PCIe reset and software reset. When reaching maximum value counter does not wrap-around.

Field	Bit(s)	Init.	Access Type	Description
O2BGPTC	31:0	0x0	RC	OS2BMC good packets transmitted count.

8.2.4.18.84 OS2BMC packets transmitted by host - O2BSPC (0x000087B0)

This register counts the total number of packets originating from the function that were sent to the manageability path. This includes packets received by the BMC and packet dropped in the X540 due to congestion conditions or due to anti-spoof check.

The counter is cleared when read by software device driver. The counter is also cleared by PCIe reset and Software reset. When reaching maximum value counter does not wrap-around.

Field	Bit(s)	Init.	Access Type	Description
O2BPC	31:0	0x0	RC	OS2BMC packets transmitted count.

8.2.4.18.85 BMC Total Unicast Packets Received - BUPRC (0x00004180)

This register counts the number of good (no errors) unicast packets received. This register does not count unicast packets received that fail to pass address filtering. This register only increments if receives are enabled. This register does not count packets counted by the Missed Packet Count (MPC) register. Packets sent to the manageability engine are included in this counter. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BUPRC	31:0	0x0	RC	Number of Unicast packets received.

8.2.4.18.86 BMC Total Multicast Packets Received - BMPRC (0x00004184)

This register counts the same events as the MPRC register (Multicast Packets Received Count) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BMPRC	31:0	0x0	RC	Number of good (non-erred) multicast packets received that pass L2 filtering (excluding Broadcast packets). This register does not count received flow control packets. This registers counts packets regardless on receive enablement.

8.2.4.18.87 BMC Total Broadcast Packets Received - BBPRC (0x00004188)

This register counts the same events as the BPRC register (Broadcast Packets Received Count) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BBPRC	31:0	0x0	RC	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.

8.2.4.18.88 BMC Total Unicast Packets Transmitted - BUPTC (0x0000418C)

This register counts the number of unicast packets transmitted. This register decrements only if transmits are enabled. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BUPTC	31:0	0x0	RC	Number of unicast packets transmitted.

8.2.4.18.89 BMC Total Multicast Packets Transmitted - BMPTC (0x00004190)

This register counts the same events as the MPTC register (Multicast Packets Transmitted Count) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BMPTC	31:0	0x0	RC	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.

8.2.4.18.90 BMC Total Broadcast Packets Transmitted - BBPTC (0x00004194)

This register counts the same events as the BPTC register (Broadcast Packets Transmitted Count) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BBPTC	31:0	0x0	RC	Number of broadcast packets transmitted count. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets

8.2.4.18.91 BMC FCS Receive Errors - BCRCERRS (0x00004198)

This register counts the same events as the CRCERRS register (CRC Error Count) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BCEC	31:0	0x0	RC	CRC error count. Counts the number of receive packets with CRC errors. In order for a packet to be counted in this register, it must be 64 bytes or greater (from <Destination Address> through <CRC>, inclusively) in length. This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.4.18.92 BMC Pause XON Frames Received - BXONRXC (0x0000419C)

This register counts the same events as the LXONRXC register (Link XON Received Count) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BXONRXC	15:0	0x0	RC	Number of XON packets Received. Sticks to 0xFFFF. XON packets can use the global address, or the station address. This register counts any XON packet whether it is a legacy XON or a priority XON. Each XON packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only the LXOFFRXCNT counter is incremented.
RESERVED	31:16	0x0	RC	Reserved

8.2.4.18.93 BMC Pause XOFF Frames Received - BXOFFRXC (0x000041E0)

This register counts the same events as the LXOFFRXC register (Link XOFF Received Count) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BXOFFRXC	15:0	0x0	RC	Number of XOFF packets Received. Sticks to 0xFFFF. XOFF packets can use the global address, or the station address. This register counts any XOFF packet whether it is a legacy XOFF or a priority XOFF. Each XOFF packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only this counter is incremented.
RESERVED	31:16	0x0	RC	Reserved

8.2.4.18.94 BMC Pause XON Frames Transmitted - BXONTXC (0x000041E4)

This register counts the same events as the LXONTXC register (Link XON Transmitted Count) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BXONTXC	15:0	0x0	RC	Number of XON packets Transmitted. Sticks to 0xFFFF. XONTXC is incremented by one for each Link XON packet when MFLCN.RFCE is set and for each Priority XON packet when corresponding MFLCN.RPFCE bit is set.
RESERVED	31:16	0x0	RC	Reserved.

8.2.4.18.95 BMC Pause XOFF Frames Transmitted - BXOFFTXC (0x000041E8)

This register counts the same events as the LXOFFTXC register (Link XOFF Transmitted Count) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Access Type	Description
BXOFFTXC	15:0	0x0	RC	Number of XOFF packets Transmitted. Sticks to 0xFFFF. XOFFTXC is incremented by one for each Link XOFF packet when MFLCN.RFCE is set and for each Priority XOFF packet when corresponding MFLCN.RPFCE bit is set.
RESERVED	31:16	0x0	RC	Reserved.

8.2.4.19 Wake-Up Control Registers

8.2.4.19.1 Wake Up Control Register - WUC (0x00005800)

Note: The PME_En and PME_Status bits are reset when LAN_PWR_GOOD is 0. When AUX_PWR=0, these bits are also reset by the assertion of PE_RST_N.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	0	0b	RSV	Reserved.
PME_EN	1	0b	RW	PME_En. This bit is used by the driver to read the PME_En bit of the Power Management Control / Status Register (PMCSR) without writing to PCIe configuration space. Writing a 1b to this bit clears it.
PME_STATUS	2	0b	RW1C	PME_Status. This bit is set when the X540 receives a wakeup 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 is also cleared.
RESERVED	3	0b	RSV	Reserved.
RESERVED	4	1b	RSV	Reserved.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.4.19.2 Wake Up Filter Control Register - WUFC (0x00005808)

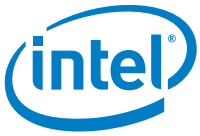
Note: This register is used to enable each of the pre-defined and flexible filters for wake up support. A value of 1 means the filter is turned on, and a value of 0 means the filter is turned off.

Field	Bit(s)	Init.	Access 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	Ignore TCO/managements packets for wake up. 0b = Ignore all TCO/management packets for wake up, except to packets that meet the criteria defined in the MANC2H register via the Host Enable field (i.e. are intended also to the Host and not only to the BMC). 1b = Ignore all TCO/management packets for wake up, even if in normal operation it is forwarded to the Host in addition to the BMC.
FLX0	16	0b	RW	Flexible Filter 0 Enable.
FLX1	17	0b	RW	Flexible Filter 1 Enable.
FLX2	18	0b	RW	Flexible Filter 2 Enable.
FLX3	19	0b	RW	Flexible Filter 3 Enable.
FLX4	20	0b	RW	Flexible Filter 4 Enable.
FLX5	21	0b	RW	Flexible Filter 5 Enable.
RESERVED	31:22	0x0	RSV	Reserved.

8.2.4.19.3 Wake Up Status Register - WUS (0x00005810) RX-Filter

Notes: This register is used to record statistics about all Wake Up packets received. If a packet matches multiple criteria than 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 software device driver.



Field	Bit(s)	Init.	Access Type	Description
LNKC	0	0b	RW1C	Link Status Changed.
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 whose hashed to a value that corresponded 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	Indicates that a Manageability event that should cause a PME happened.
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.
FLX5	21	0b	RW1C	Flexible Filter 5 Match.
RESERVED	31:22	0x0	RSV	Reserved.

8.2.4.19.4 IP Address Valid - IPAV (0x00005838) RX-Filter

The IP Address Valid indicates whether the IP addresses in the IP Address Table are valid.

Field	Bit(s)	Init.	Access 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.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	15:4	0x0	RSV	Reserved.
V60	16	0b	RW	IPv6 Address 0 Valid.
RESERVED	31:17	0x0	RSV	Reserved.

8.2.4.19.5 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.	Access Type	Description
IPV4ADDR	31:0	X	RW	IPv4 Address 'n', 'n' = 0...3.

8.2.4.19.6 IPv6 Address Table - IP6AT[n] (0x00005880 + 0x4*n, n=0...3)

1 x IPv6 addresses for Neighbor Discovery packet filtering and Directed IPv6 packet wake up. According to the Power Management section; One Ipv6 address is supported and it is programmed in the Ipv6 Address Table (IP6AT).

Field	Bit(s)	Init.	Access Type	Description
IPV6ADDR	31:0	X	RW	4 x Register IPv6 filter. Register 'n' contains bytes '4*n' up to '4*n+3' of the IPv6 address. LS byte of register 0b is first on the wire.

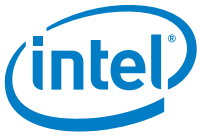
8.2.4.19.7 Filter 0 Dword Lower - FHFT_FILTER0_DW_L[n] (0x00009000 + 0x10*n, n=0...14)

Each of the 6 Flexible Host Filters Table registers (FHFT) 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 FHFT register.

Each 128B filter is composed of 32 Dword entries, where each 2 Dwords are accompanied by an 8-bit mask, one bit per filter byte.

Note: The length field must be 8 bytes aligned. For filtering packets shorter than 8 bytes aligned the values should be rounded up to the next 8 bytes aligned value, hardware implementation compares 8 bytes at a time so it should get extra zero masks (if needed) until the end of the length value.

In case the actual length which is defined by the length field register and the mask bits is not 8 bytes aligned there might be a case that a packet which is shorter then the actual required length pass the flexible filter. This might happen due to a comparison of up to 7 bytes that come after the packet but are not a real part of the packet.



The last Dword of each filter contains a length field defining the number of bytes from the beginning of the packet compared by this filter, the length field should be 8 bytes aligned value. If the actual packet length is less than (length - 8) (length is the value specified by the length field), the filter fails.

Otherwise, it depends on the result of actual byte comparison. The value should not be greater than 128. See registers structure and description in Flexible Filter.

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.8 Filter 0 Dword Upper - FHFT_FILTER0_DW_U[n] (0x00009004 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.19.9 Filter 0 Mask - FHFT_FILTER0_MASK[n] (0x00009008 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.10 Filter 0 Reserved - FHFT_FILTER0_RESERVED[n] (0x0000900C + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:0	0x0	RW	

8.2.4.19.11 Filter 0 DW30 - FHFT_FILTER0_DW30 (0x000090F0)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.12 Filter 0 DW31 - FHFT_FILTER0_DW31 (0x000090F4)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.19.13 Filter 0 Mask[120:127] - FHFT_FILTER0_MASK_120_127 (0x000090F8)

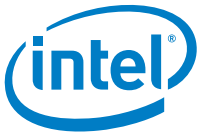
Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.14 Filter 0 Length - FHFT_FILTER0_LENGTH (0x000090FC)

Field	Bit(s)	Init.	Access Type	Description
LENGTH	6:0	0x0	RW	
RESERVED	31:7	0x0	RSV	

8.2.4.19.15 Filter 1 Dword Lower - FHFT_FILTER1_DW_L[n] (0x00009100 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	



8.2.4.19.16 Filter 1 Dword Upper - FHFT_FILTER1_DW_U[n] (0x00009104 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.19.17 Filter 1 Mask - FHFT_FILTER1_MASK[n] (0x00009108 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.18 Filter 1 Reserved - FHFT_FILTER1_RESERVED[n] (0x0000910C + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:0	0x0	RW	

8.2.4.19.19 Filter 1 Dword 30 - FHFT_FILTER1_DW30 (0x000091F0)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.20 Filter 1 Dword 31 - FHFT_FILTER1_DW31 (0x000091F4)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.19.21 Filter 1 Mask[120:127] - FHFT_FILTER1_MASK_120_127 (0x000091F8)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.22 Filter 1 Length - FHFT_FILTER1_LENGTH (0x000091FC)

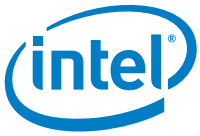
Field	Bit(s)	Init.	Access Type	Description
LENGTH	6:0	0x0	RW	
RESERVED	31:7	0x0	RSV	

8.2.4.19.23 Filter 2 Dword Lower - FHFT_FILTER2_DW_L[n] (0x00009200 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.24 Filter 2 Dword Upper - FHFT_FILTER2_DW_U[n] (0x00009204 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	



8.2.4.19.25 Filter 2 Mask - FHFT_FILTER2_MASK[n] (0x00009208 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.26 Filter 2 Reserved - FHFT_FILTER2_RESERVED[n] (0x0000920C + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:0	0x0	RW	

8.2.4.19.27 Filter 2 Dword 30 - FHFT_FILTER2_DW30 (0x000092F0)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.28 Filter 2 Dword 31 - FHFT_FILTER2_DW31 (0x000092F4)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.19.29 Filter 2 Mask[120:127] - FHFT_FILTER2_MASK_120_127 (0x000092F8)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.30 Filter 2 Length - FHFT_FILTER2_LENGTH (0x000092FC)

Field	Bit(s)	Init.	Access Type	Description
LENGTH	6:0	0x0	RW	
RESERVED	31:7	0x0	RSV	

8.2.4.19.31 Filter 3 Dword Lower - FHFT_FILTER3_DW_L[n] (0x00009300 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.32 Filter 3 Dword Upper - FHFT_FILTER3_DW_U[n] (0x00009304 + 0x10*n, n=0...14)

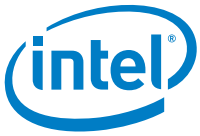
Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.19.33 Filter 3 Mask - FHFT_FILTER3_MASK[n] (0x00009308 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.34 Filter 3 Reserved - FHFT_FILTER3_RESERVED[n] (0x0000930C + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:0	0x0	RW	



8.2.4.19.35 Filter 3 Dword 30 - FHFT_FILTER3_DW30 (0x000093F0)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.36 Filter 3 Dword 31 - FHFT_FILTER3_DW31 (0x000093F4)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.19.37 Filter 3 Mask[120:127] - FHFT_FILTER3_MASK_120_127 (0x000093F8)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.38 Filter 3 Length - FHFT_FILTER3_LENGTH (0x000093FC)

Field	Bit(s)	Init.	Access Type	Description
LENGTH	6:0	0x0	RW	
RESERVED	31:7	0x0	RSV	

8.2.4.19.39 Filter 4 Dword Lower - FHFT_FILTER4_DW_L[n] (0x00009800 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.40 Filter 4 Dword Upper - FHFT_FILTER4_DW_U[n] (0x00009804 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.19.41 Filter 4 Mask - FHFT_FILTER4_MASK[n] (0x00009808 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.42 Filter 4 Reserved - FHFT_FILTER4_RESERVED[n] (0x0000980C + 0x10*n, n=0...14)

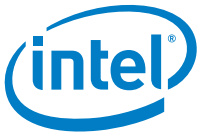
Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:0	0x0	RW	

8.2.4.19.43 Filter 4 DW30 - FHFT_FILTER4_DW30 (0x000098F0)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.44 Filter 4 DW31 - FHFT_FILTER4_DW31 (0x000098F4)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	



8.2.4.19.45 Filter 4 Mask[120:127] - FHFT_FILTER4_MASK_120_127 (0x000098F8)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.46 Filter 4 Length - FHFT_FILTER4_LENGTH (0x000098FC)

Field	Bit(s)	Init.	Access Type	Description
LENGTH	6:0	0x0	RW	
RESERVED	31:7	0x0	RSV	

8.2.4.19.47 Filter 5 Dword Lower - FHFT_FILTER5_DW_L[n] (0x00009900 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.48 Filter 5 Dword Upper - FHFT_FILTER5_DW_U[n] (0x00009904 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.19.49 Filter 5 Mask - FHFT_FILTER5_MASK[n] (0x00009908 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.19.50 Filter 5 Reserved - FHFT_FILTER5_RESERVED[n] (0x0000990C + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:0	0x0	RW	

8.2.4.19.51 Filter 5 DW30 - FHFT_FILTER5_DW30 (0x000099F0)

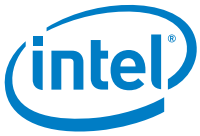
Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.19.52 Filter 5 DW31 - FHFT_FILTER5_DW31 (0x000099F4)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.19.53 Filter 5 Mask[120:127] - FHFT_FILTER5_MASK_120_127 (0x000099F8)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	



8.2.4.19.54 Filter 5 Length - FHFT_FILTER5_LENGTH (0x000099FC)

Field	Bit(s)	Init.	Access Type	Description
LENGTH	6:0	0x0	RW	
RESERVED	31:7	0x0	RSV	

8.2.4.20 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.

8.2.4.20.1 Management VLAN TAG Value - MAVTV[n] (0x00005010 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
VID	11:0	0x0	RW	Contains the VLAN ID that should be compared with the incoming packet if the corresponding bit in MFVAL.VLAN is set.
RESERVED	31:12	0x0	RSV	Reserved.

8.2.4.20.2 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.). Note that SCTP packets do not match the MFUTP filters.

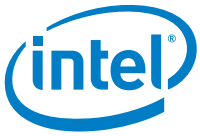
Field	Bit(s)	Init.	Access Type	Description
MFUTP_2N	15:0	0x0	RW	(2n)-th Management Flex UDP/TCP port.
MFUTP_2N_1	31:16	0x0	RW	(2n+1)-th Management Flex UDP/TCP port.

8.2.4.20.3 Management Ethernet Type Filters - METF[n] (0x00005190 + 0x4*n, n=0...3

Field	Bit(s)	Init.	Access Type	Description
ETYPE	15:0	0x0	RW	EtherType value to be compared against the L2 EtherType field in the Rx packet. **Note:** Appears in Little Endian order (high byte first on the wire).
RESERVED	29:16	0x0	RSV	Reserved.
POLARITY	30	0b	RW	0b = Positive filter - Filter enters the decision filters if a match occurred. 1b = Negative filter - Filter enters the decision filters if a match did not occur.
RESERVED	31	0b	RSV	Reserved.

8.2.4.20.4 Management Control Register - MANC (0x00005820)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	16:0	0x0	RSV	Reserved.
RCV_TCO_EN	17	0b	RW	Receive TCO Packets Enabled. When this bit is set it enables the receive flow to the manageability block.This bit should be set only if at least one of MANC.EN_BMC2OS or MANC.EN_BMC2NET bits are set.
RESERVED	18	0b	RSV	Reserved.
RCV_ALL	19	0b	RW	Receive All Enable. When set, all packets are received from the wire and passed to the manageability block.
MCST_PASS_L2	20	0b	RW	Receive all multicast: When set, all received multicast packets pass L2 filtering and might be directed to the MNG or Host by a one of the decision filters. Broadcast packets are not forwarded by this bit.
EN_MNG2HOST	21	0b	RW	Enable Manageability packets to host memory. This bit enables the functionality of the MANC2H register. When set the packets that are specified in the MANC2H registers are forwarded to the host memory too, if they pass manageability filters.
BYPASS_VLAN	22	0b	RW	When set, VLAN filtering is bypassed for MNG packets.
EN_XSUM_FILTER	23	0b	RW	When set this bit enables Xsum filtering to Manageability. Meaning, only packets that pass L3, L4 checksum are sent to the manageability block. Note that this capability is not provided for tunneled packets.
EN_IPV4_FILTER	24	0b	RW	Enable IPv4 address Filters, when set, the last 128 bits of the MIPAF register are used to store 4 IPv4 addresses for IPv4 filtering. When cleared, these bits store a single IPv6 filter.



Field	Bit(s)	Init.	Access Type	Description
FIXED_NET_TYPE	25	0b	RW	Fixed Next Type. If set, only packets matching the net type defined by the NET_TYPE field passes to manageability; otherwise, both tagged and un-tagged packets might be forwarded to the manageability engine.
NET_TYPE	26	0b	RW	Net Type. 0b = Pass only un-tagged packets. 1b = Pass only VLAN tagged packets. Valid only if FIXED_NET_TYPE is set.
RESERVED	27	0b	RW	Reserved.
EN_BMC2OS	28	0b	RW	Enable BMC2OS and OS2BMC Traffic. 0b = The BMC can not communicate with the operating system. 1b = The BMC can communicate with the operating system. When cleared the BMC traffic is not forwarded to the operating system, even if the Host MAC address filter and VLANs (RAH/L, MTA, VFTA and PFVLVF registers) indicate that it should. When cleared the operating system 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 can not communicate with the network. 1b = The BMC can communicate with the network. When cleared the BMC traffic is not forwarded to the network and the network traffic is not forwarded to the BMC even if the decision filters indicates it should. This bit does not impact the host to BMC traffic. **Note:** This bit can change while the host is sending or receiving traffic.
RESERVED	31:30	0x0	RSV	Reserved.

8.2.4.20.5 Manageability Filters Valid - MFVAL (0x00005824)

Field	Bit(s)	Init.	Access Type	Description
MAC	3:0	0x0	RW	MAC. Indicates if the MAC unicast filter registers (MMAH, MMAL) contain valid Ethernet MAC Addresses. Bit 0 corresponds to filter 0, etc.
RESERVED	7:4	0x0	RSV	Reserved.
VLAN	15:8	0x0	RW	VLAN. Indicates if the VLAN filter registers (MAVTV) contain valid VLAN tags. Bit 8 corresponds to filter 0, etc.
IPV4	19:16	0x0	RW	IPv4. Indicates if the IPv4 address filters (MIPAF) contain valid IPv4 addresses. Bit 16 corresponds to IPv4 address 0. These bits apply only when IPv4 address filters are enabled (MANC.EN_IPv4_FILTER=1).

Field	Bit(s)	Init.	Access Type	Description
RESERVED	23:20	0x0	RSV	Reserved.
IPV6	27:24	0x0	RW	IPv6. Indicates if the IPv6 address filter registers (MIPAF) contain valid IPv6 addresses. Bit 24 corresponds to address 0, etc. Bit 27 (filter 3), applies only when IPv4 address filters are not enabled. (MANC.EN_IPv4_FILTER=0)
RESERVED	31:28	0x0	RSV	Reserved.

8.2.4.20.6 Management Control To Host Register - MANC2H (0x00005860)

Field	Bit(s)	Init.	Access Type	Description
HOST_ENABLE	7:0	0x0	RW	Host Enable, when set, indicates that packets routed by the manageability filters to manageability are also sent to the host. Bit 0 corresponds to decision filter (MDEF[0] and MDEF_EXT[0]), bit 1 corresponds to decision filter (MDEF[1] and MDEF_EXT[1]), etc. The MANC2H routing is further enabled by a global MANC.EN_MNG2HOST bit.
RESERVED	31:8	0x0	RSV	Reserved.

8.2.4.20.7 Manageability Decision Filters - MDEF[n] (0x00005890 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
UNICAST_AND	0	0b	RW	Unicast. Controls the inclusion of unicast address filtering in the manageability filter decision (AND section).
BROADCAST_AND	1	0b	RW	Broadcast. Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section).
VLAN_AND	2	0b	RW	VLAN. Controls the inclusion of VLAN address filtering in the manageability filter decision (AND section).
IP_ADDRESS_AND	3	0b	RW	IP Address. Controls the inclusion of IP address filtering in the manageability filter decision (AND section).
UNICAST_OR	4	0b	RW	Unicast. Controls the inclusion of unicast address filtering in the manageability filter decision (OR section).
BROADCAST_OR	5	0b	RW	Broadcast. Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section).
MULTICAST_AND	6	0b	RW	Multicast. Controls the inclusion of Multicast address filtering in the manageability filter decision (AND section). Broadcast packets are not included by this bit. The packet must pass some L2 filtering to be included by this bit, either by the MANC.MCST_PASS_L2 or by some dedicated Ethernet MAC Address.

Field	Bit(s)	Init.	Access Type	Description
ARP_REQUEST_OR	7	0b	RW	ARP Request. Controls the inclusion of ARP Request filtering in the manageability filter decision (OR section).
ARP_RESPONSE_OR	8	0b	RW	ARP Response. Controls the inclusion of ARP Response filtering in the manageability filter decision (OR section).
NEIGHBOR_DISCOVERY_OR	9	0b	RW	Neighbor Discovery. 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.
PORT_0X298_OR	10	0b	RW	Port 0x298. Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section).
PORT_0X26F_OR	11	0b	RW	Port 0x26F. Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section).
FLEX_PORT_OR	27:12	0x0	RW	Flex port. Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc.
FLEX_TCO_OR	31:28	0x0	RW	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.

8.2.4.20.8 Manageability Decision Filters Ext - MDEF_EXT[n] (0x00005160 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
L2_ETHERTYPE_AND	3:0	0x0	RW	L2 EtherType. Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section).
RESERVED	7:4	0x0	RSV	Reserved. for additional L2 EtherType AND filters.
L2_ETHERTYPE_OR	11:8	0x0	RW	L2 EtherType. Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section).
RESERVED	15:12	0x0	RSV	Reserved. for additional L2 EtherType OR filters.
RESERVED	29:16	0x0	RSV	Reserved.
APPLY_TO_NETWORK_TRAFFIC	30	0b	RW	0b = This decision filter does not apply to traffic received from the network. 1b = This decision filter applies to traffic received from the network.
APPLY_TO_HOST_TRAFFIC	31	0b	RW	0b = This decision filter does not apply to traffic received from the host. 1b = This decision filter applies to traffic received from the host.

8.2.4.20.9 BMC IP address Register - BMCIP[n] (0x00005050 + 0x4*n, n=0...3)

These registers contains the BMC IP Address table.

Field	Bit(s)	Init.	Access Type	Description
IPADDR	31:0	0x0	RW	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, IPSRXIPADDR 0...2 must be written with zeros. **Note:** Field is defined in Big Endian (LS byte is first on the wire).

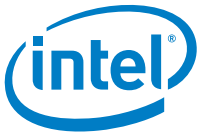
8.2.4.20.10 BMC IP Valid Register - BMCIPVAL (0x00005060)

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.	Access Type	Description
IPADDR_TYPE	0	0b	RW	0b = IPv4. 1b = IPv6.
IPADDR_VALID	1	0b	RW	0b = IP address in BMCIP is not valid. 1b = IP address in BMCIP is valid.
RESERVED	31:2	0x0	RSV	Reserved.

8.2.4.20.11 Manageability IP Address Filter - MIPAF[n,m] (0x000058B0 + 0x4*n + 0x10*m, n=0...3, m=0...3)

Field	Bit(s)	Init.	Access 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 = 0b = MIPAF[m,n] register holds Dword 'n' of IPv6 filter 'm' (4 x IPv6 filters). For each n, m, m=0...3, n=0...3 while MANC.EN_IPv4_FILTER = 1b = 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 (LSB, LS address is first on the wire).



8.2.4.20.12 Manageability Ethernet MAC Address Low - MMAL[n] (0x00005910 + 0x8*n, n=0...3)

Field	Bit(s)	Init.	Access Type	Description
MMAL	31:0	X	RW	Manageability Ethernet MAC Address Low. The lower 32 bits of the 48-bit Ethernet MAC address. **Note:** Appear in Big Endian order (LSB of MMAL is first on the wire).

8.2.4.20.13 Manageability Ethernet MAC Address High - MMAH[n] (0x00005914 + 0x8*n, n=0...3)

Field	Bit(s)	Init.	Access Type	Description
MMAH	15:0	X	RW	Manageability Ethernet MAC Address High. The upper 16 bits of the 48-bit Ethernet MAC address. **Note:** Appear in Big Endian order (MSB of MMAH is last on the wire).
RESERVED	31:16	0x0	RSV	Reserved. Reads as 0x0. Ignored on write.

8.2.4.20.14 FTFT Filter 0 Dword Lower - FTFT_FILTER0_DW_L[n] (0x00009400 + 0x10*n, n=0...14)

Each of the 4 Flexible TCO Filters 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 128B filter is composed of 32 Dword entries, where each 2 Dwords are accompanied by an 8-bit mask, one bit per filter byte. 15:8] etc.... The mask field is set so that bit 0 in the mask masks byte 0, bit 1 masks byte 1 etc.... 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.

Notes:

The mask field must be 8 bytes aligned even if the length field is not 8 bytes aligned as hardware implementation compares 8 bytes at a time so it should get extra masks until the end of the next quad word. Any mask bit that is located after the length should be set to 0 indicating no comparison should be done.

In case the actual length which is defined by the length field register and the mask bits is not 8 bytes aligned there might be a case that a packet which is shorter then the actual required length passes the flexible filter. This might happen due to comparison of up to 7 bytes that come after the packet but are not a real part of the packet.

The last Dword 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. Host write/read access to these registers should be avoided. See registers structure and high level description in Flexible 128 Bytes Filter (TCO Filter).

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.20.15 FTFT Filter 0 Dword Upper - FTFT_FILTER0_DW_U[n] (0x00009404 + 0x10*n, n=0...14)

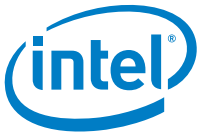
Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.20.16 FTFT Filter 0 Mask - FTFT_FILTER0_MASK[n] (0x00009408 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.20.17 FTFT Filter 0 Reserved - FTFT_FILTER0_RESERVED[n] (0x0000940C + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:0	0x0	RW	



8.2.4.20.18 FTFT Filter 0 DW30 - FTFT_FILTER0_DW30 (0x000094F0)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.20.19 FTFT Filter 0 DW31 - FTFT_FILTER0_DW31 (0x000094F4)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.20.20 FTFT Filter 0 Mask[120:127] - FTFT_FILTER0_MASK_120_127 (0x000094F8)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.20.21 FTFT Filter 0 Length - FTFT_FILTER0_LENGTH (0x000094FC)

Field	Bit(s)	Init.	Access Type	Description
LENGTH	6:0	0x0	RW	
RESERVED	31:7	0x0	RSV	

8.2.4.20.22 FTFT Filter 1 Dword Lower - FTFT_FILTER1_DW_L[n] (0x00009500 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.20.23 FTFT Filter 1 Dword Upper - FTFT_FILTER1_DW_U[n] (0x00009504 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.20.24 FTFT Filter 1 Mask - FTFT_FILTER1_MASK[n] (0x00009508 + 0x10*n, n=0...14)

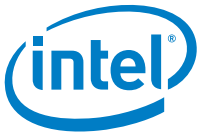
Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.20.25 FTFT Filter 1 Reserved - FTFT_FILTER1_RESERVED[n] (0x0000950C + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:0	0x0	RW	

8.2.4.20.26 FTFT Filter 1 DW30 - FTFT_FILTER1_DW30 (0x000095F0)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	



8.2.4.20.27 FTFT Filter 1 DW31 - FTFT_FILTER1_DW31 (0x000095F4)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.20.28 FTFT Filter 1 Mask[120:127] - FTFT_FILTER1_MASK_120_127 (0x000095F8)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.20.29 FTFT Filter 1 Length - FTFT_FILTER1_LENGTH (0x000095FC)

Field	Bit(s)	Init.	Access Type	Description
LENGTH	6:0	0x0	RW	
RESERVED	31:7	0x0	RSV	

8.2.4.20.30 FTFT Filter 2 Dword Lower - FTFT_FILTER2_DW_L[n] (0x00009600 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.20.31 FTFT Filter 2 Dword Upper - FTFT_FILTER2_DW_U[n] (0x00009604 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.20.32 FTFT Filter 2 Mask - FTFT_FILTER2_MASK[n] (0x00009608 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.20.33 FTFT Filter 2 Reserved - FTFT_FILTER2_RESERVED[n] (0x0000960C + 0x10*n, n=0...14)

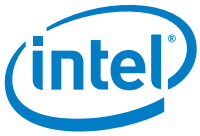
Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:0	0x0	RW	

8.2.4.20.34 FTFT Filter 2 DW30 - FTFT_FILTER2_DW30 (0x000096F0)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.20.35 FTFT Filter 2 DW31 - FTFT_FILTER2_DW31 (0x000096F4)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	



8.2.4.20.36 FTFT Filter 2 Mask[120:127] - FTFT_FILTER2_MASK_120_127 (0x000096F8)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.20.37 FTFT Filter 2 Length - FTFT_FILTER2_LENGTH (0x000096FC)

Field	Bit(s)	Init.	Access Type	Description
LENGTH	6:0	0x0	RW	
RESERVED	31:7	0x0	RSV	

8.2.4.20.38 FTFT Filter 3 Dword Lower - FTFT_FILTER3_DW_L[n] (0x00009700 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.20.39 FTFT Filter 3 Dword Upper - FTFT_FILTER3_DW_U[n] (0x00009704 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.20.40 FTFT Filter 3 Mask - FTFT_FILTER3_MASK[n] (0x00009708 + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	
RESERVED	31:8	0x0	RSV	

8.2.4.20.41 FTFT Filter 3 Reserved - FTFT_FILTER3_RESERVED[n] (0x0000970C + 0x10*n, n=0...14)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:0	0x0	RW	

8.2.4.20.42 FTFT Filter 3 DW30 - FTFT_FILTER3_DW30 (0x000097F0)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	

8.2.4.20.43 FTFT Filter 3 DW31 - FTFT_FILTER3_DW31 (0x000097F4)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	

8.2.4.20.44 FTFT Filter 3 Mask[120:127] - FTFT_FILTER3_MASK_120_127 (0x000097F8)

Field	Bit(s)	Init.	Access Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:8	0x0	RSV	

8.2.4.20.45 FTFT Filter 3 Length - FTFT_FILTER3_LENGTH (0x000097FC)

Field	Bit(s)	Init.	Access Type	Description
LENGTH	6:0	0x0	RW	
RESERVED	31:7	0x0	RSV	

8.2.4.21 Time Sync (IEEE 1588) Registers

8.2.4.21.1 Rx Time Sync Control Register - TSYNCRXCTL (0x00005188)

Field	Bit(s)	Init.	Access Type	Description
RXTT	0	0b	RO	Rx Time Stamp Valid. Equals 1b when a valid value for Rx time stamp is captured in the Rx time stamp register. Cleared by a read of the Rx time stamp register RXSTMPH).
TYPE	3:1	0x0	RW	Type of Packets to Time Stamp. 000b = Time stamp L2 (V2) packets only (Sync or Delay_req depends on message type in Rx message type register low - RXMTRL and packets with message ID 2 and 3). 001b = Time stamp L4 (V1) packets only (Sync or Delay_req depends on message type in Rx message type register low - RXMTRL). 010b = Times tamp V2 (L2 and L4) packets (Sync or Delay_req depends on message type in Rx message type register low - RXMTRL and packets with message ID 2 and 3). 101b = Time stamp all packets in which message id bit 3 is zero, which means timestamp all event packets. This is applicable for V2 packets only. 011b, 100b, 110b and 111b = Reserved.
EN	4	0b	RW	Enable Rx Time Stamp. 0x0 = Time stamping disabled. 0x1 = Time stamping enabled.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.4.21.2 Rx Timestamp Low - RXSTMPL (0x000051E8)

Field	Bit(s)	Init.	Access Type	Description
RXSTMPL	31:0	0x0	RO	Rx timestamp LSB value.

8.2.4.21.3 Rx Timestamp High - RXSTMPH (0x000051A4)

Field	Bit(s)	Init.	Access Type	Description
RXSTMPH	31:0	0x0	RO	Rx Time Stamp MSB Value.

8.2.4.21.4 Rx Timestamp attributes low - RXSATRL (0x000051A0)

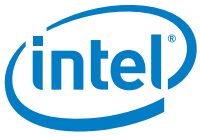
Field	Bit(s)	Init.	Access Type	Description
SOURCEIDL	31:0	0x0	RO	SourceUUID Low. Captured bytes 24-27 in the PTP message while the MSB is last on the wire. In V1 PTP packet it is the 4 LSBs of the SourceUUID field and in V2 PTP packet it is part of the Source Port ID field.

8.2.4.21.5 Rx Timestamp Attributes High - RXSATRH (0x000051A8)

Field	Bit(s)	Init.	Access Type	Description
SOURCEIDH	15:0	0x0	RO	SourceUUID High. Captured bytes 22-23 in the PTP message while the LSB is first on the wire. In V1 PTP packet it is the 2 MS bytes of the SourceUUID field and in V2 PTP packet it is part of the Source Port ID field.
SEQUENCEID	31:16	0x0	RO	SequenceID. Captured value of the SequenceID field in the PTP Rx packet while LSB first on the wire.

8.2.4.21.6 Rx Message Type Register Low - RXMTRL (0x00005120)

Field	Bit(s)	Init.	Access Type	Description
CTRLT	7:0	0x0	RW	V1 Control to Time Stamp.
MSGT	15:8	0x0	RW	V2 Message ID to Time Stamp.



Field	Bit(s)	Init.	Access Type	Description
UDPT	31:16	0x13F	RW	UDP Port Number to Time Stamp.

8.2.4.21.7 Tx Time Sync Control Register - TSYNCTXCTL (0x00008C00)

Field	Bit(s)	Init.	Access Type	Description
TXTT	0	0b	RO	Tx Time Stamp Valid. Equals 1b when a valid value for Tx time stamp is captured in the Tx time stamp register. It is cleared by a read of the Tx times tamp register TXSTMPH).
RESERVED	3:1	0x0	RSV	Reserved.
EN	4	0b	RW	Enable Tx Time Stamp 0x0 = Time stamping disabled. 0x1 = Time stamping enabled.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.4.21.8 Tx Timestamp Value Low - TXSTMPL (0x00008C04)

Field	Bit(s)	Init.	Access Type	Description
TXSTMPL	31:0	0x0	RO	Tx Time Stamp LSB Value.

8.2.4.21.9 Tx Timestamp Value High - TXSTMPH (0x00008C08)

Field	Bit(s)	Init.	Access Type	Description
TXSTMPH	31:0	0x0	RO	Tx Time Stamp MSB Value.

8.2.4.21.10 System Time Register Low - SYSTIMEL (0x00008C0C)

Field	Bit(s)	Init.	Access Type	Description
STL	31:0	0x0	RW	System Time LSB Register.

8.2.4.21.11 System Time Register High - SYSTIMEH (0x00008C10)

Field	Bit(s)	Init.	Access Type	Description
STH	31:0	0x0	RW	System Time MSB Register.

8.2.4.21.12 Increment Attributes Register - TIMEINCA (0x00008C14)

Field	Bit(s)	Init.	Access Type	Description
INCVALUE	30:0	0x0	RW	Increment value to the SYSTIME registers on each DMA clock.
RESERVED	31	0b	RSV	Reserved.

8.2.4.21.13 Time Adjustment Offset Register Low - TIMEADJL (0x00008C18)

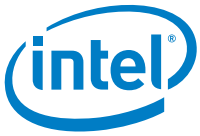
Field	Bit(s)	Init.	Access Type	Description
TADJL	31:0	0x00	RW	Time Adjustment Value – Low.

8.2.4.21.14 Time Adjustment Offset Register High - TIMEADJH (0x00008C1C)

Field	Bit(s)	Init.	Access Type	Description
TADJH	30:0	0x00	RW	Time Adjustment Value – High.
SIGN	31	0b	RW	Sign ("0"="+", "1"="-").

8.2.4.21.15 TimeSync Auxiliary Control Register - TSAUXC (0x00008C20)

Field	Bit(s)	Init.	Access Type	Description
EN_TT0	0	0b	RW	Enable Target Time 0. **Note**: Target Time 0 is mutually exclusive with clock out. Therefore, setting both EN_CLK and EN_TT0 is forbidden.



Field	Bit(s)	Init.	Access Type	Description
EN_TT1	1	0b	RW	Enable Target Time 1.
EN_CLK	2	0b	RW	Enable Configurable TimeSync Clock Out. **Note:** Clock out is mutually exclusive with Target Time 0. Therefore, setting both EN_CLK and EN_TT0 is forbidden.
SYNCLK	3	0b	RW	The SYNCLK bit controls the initial time the Clock Out signal starts. This bit is meaningful when setting the EN_CLK bit in this register. 0b = The Clock Out starts immediately. 1b = The Clock Out starts at the time defined by TRGTTIME(0) register.
SDP2_INT	4	0b	RW	When set to 1b any sampled transition on SDP2 input sets the TimeSync bit in the EICR registers as a potential interrupt source.This bit is validated at 1b only when SDP2 is used as Auxiliary Time Stamp 0 input.
SDP3_INT	5	0b	RW	When set to 1b any sampled transition on SDP3 input sets the TimeSync bit in the EICR registers as a potential interrupt source.This bit is validated at 1b only when SDP3 is used as Auxiliary Time Stamp 1 input.
SDP0_INT	6	1b	RW	When set to 1b any sampled transition on Target Time 0 / Clock out sets the TimeSync bit in the EICR registers as a potential interrupt source.This bit might be set to 1b only if Target Time 0 or Clock out are driven to SDP0.
SDP1_INT	7	0b	RW	When set to 1b any sampled transition on Target Time 1 sets the TimeSync bit in the EICR registers as a potential interrupt source.This bit might be set to 1b only if Target Time 1 is driven to SDP1.
EN_TS0	8	0b	RW	Enable Hardware Time Stamp 0.
AUTT0	9	0b	RW	Auxiliary timestamp taken. This is a read only bit field. At 0b the Auxiliary Time Stamp 0 is enabled. This bit is set to 1b by hardware following a level change of the input SDP. Reading the AUXSTMPH0 register clears this bit.
EN_TS1	10	0b	RW	Enable Hardware Time Stamp 1.
AUTT1	11	0b	RW	Auxiliary Time Stamp Taken. This is a read only bit field. At 0b the Auxiliary Time Stamp 1 is enabled. This bit is set to 1b by hardware following a level change of the input SDP. Reading the AUXSTMPH1 register clears this bit.
RESERVED	31:12	0x0	RSV	Reserved.

8.2.4.21.16 Target Time Register 0 Low - TRGTTIMEL0 (0x00008C24)

Field	Bit(s)	Init.	Access Type	Description
TTL	31:0	0x0	RW	Target Time 0 LSB Register.

8.2.4.21.17 Target Time Register 0 High - TRGTTIMEH0 (0x00008C28)

Field	Bit(s)	Init.	Access Type	Description
TTH	31:0	0x0	RW	Target Time 0 MSB Register.

8.2.4.21.18 Target Rime Register 1 Low - TRGTTIMEL1 (0x00008C2C)

Field	Bit(s)	Init.	Access Type	Description
TTL	31:0	0x0	RW	Target Time 1 LSB Register.

8.2.4.21.19 Target Rime Register 1 High - TRGTTIMEH1 (0x00008C30)

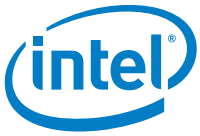
Field	Bit(s)	Init.	Access Type	Description
TTH	31:0	0x0	RW	Target Time 1 MSB Register.

8.2.4.21.20 Clock Phase Time Low Register - CLKTIMEL (0x00008C34)

Field	Bit(s)	Init.	Access Type	Description
CLKTL	31:0	0x0	RW	TimeSync Clock Phase Duration LSB Value.

8.2.4.21.21 Clock Phase Time High Register - CLKTIMEH (0x00008C38)

Field	Bit(s)	Init.	Access Type	Description
CLKTH	31:0	0x0	RW	TimeSync Clock Phase Duration MSB Value.



8.2.4.21.22 Auxiliary Time Stamp 0 Register Low - AUXSTMPL0 (0x00008C3C)

Field	Bit(s)	Init.	Access Type	Description
TST_LOW	31:0	0x0	RO	Auxiliary Time Stamp 0 LSB Value.

8.2.4.21.23 Auxiliary Time Stamp 0 Register High - AUXSTMPH0 (0x00008C40)

Field	Bit(s)	Init.	Access Type	Description
TST_HI	31:0	0x0	RO	Auxiliary Time Stamp 0 MSB Value.

8.2.4.21.24 Auxiliary Time Stamp 1 Register Low - AUXSTMPL1 (0x00008C44)

Field	Bit(s)	Init.	Access Type	Description
TST_LOW	31:0	0x0	RO	Auxiliary Time Stamp 1 LSB Value.

8.2.4.21.25 Auxiliary Time Stamp 1 Register High - AUXSTMPH1 (0x00008C48)

Field	Bit(s)	Init.	Access Type	Description
TST_HI	31:0	0x0	RO	Auxiliary Time Stamp 1 MSB Value.

8.2.4.22 Virtualization PF Registers

8.2.4.22.1 PF Virtual Control Register - PFVTCTL (0x000051B0)

Field	Bit(s)	Init.	Access Type	Description
VT_ENA	0	0b	RW	Virtualization Enabled Mode. When set, the X540 supports either 16, 32, or 64 Pools. When cleared, Rx traffic is handled internally as if it belongs to VF zero while VF zero is enabled. This bit should be set the same as MTQC.VT_Ena.

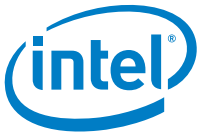
Field	Bit(s)	Init.	Access Type	Description
RESERVED	6:1	0x0	RSV	Reserved.
DEF_PL	12:7	0x0	RW	Default Pool. Pool assignment for packets that do not pass any pool queuing decision. Enabled by the Dis_Def_Pool bit.
RESERVED	28:13	0x0	RSV	Reserved.
DIS_DEF_POOL	29	0b	RW	Disable Default Pool - Determines 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 when set to 1b.
RESERVED	31	0b	RSV	Reserved.

8.2.4.22.2 PF Mailbox - PFMAILBOX[n] (0x00004B00 + 0x4*n, n=0...63)

Field	Bit(s)	Init.	Access Type	Description
STS	0	0b	WO	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 VFMailbox.
ACK	1	0b	WO	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.
VFU	2	0b	RW	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	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 PFMBICR, VFREQ, and VFACK fields.
RESERVED	31:5	0x0	RSV	Reserved.

8.2.4.22.3 PF Mailbox Interrupt Causes Register - PFMBICR[n] (0x00000710 + 0x4*n, n=0...3)

Field	Bit(s)	Init.	Access Type	Description
VFREQ	15:0	0x0	RW1C	Each bit in the VFREQ field is set when VF number (16*n+j) wrote a message in its mailbox. While 'n' is the register index, n=0...3 and 'j' is the index of the bits in the VFREQ, j=0...15.



Field	Bit(s)	Init.	Access Type	Description
VFACK	31:16	0x0	RW1C	Each bit in the VFACK field is set when VF number (16*n+j) acknowledged a PF message. While 'n' is the register index, n=0...3 and '16+j' is the index of the bits in the VFACK, j=0...15.

8.2.4.22.4 PF Mailbox Interrupt Mask Register - PFMBIMR[n] (0x00000720 + 0x4*n, n=0...1)

Field	Bit(s)	Init.	Access Type	Description
VFIM	31:0	0xFF	RW	Bit j - Mailbox indication from VF # (32*n+j) might cause an interrupt to the PF...

8.2.4.22.5 PF VFLR Events Clear - PFVFLREC[n] (0x00000700 + 0x4*n, n=0...1)

Field	Bit(s)	Init.	Access Type	Description
CLEAR_VFLE	31:0	0x0	RW1C	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.

8.2.4.22.6 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, on BME bit clear or on VF software reset. See VF Receive Enable - PFVFRE / VF Transmit Enable - PFVFTE for more details.

Field	Bit(s)	Init.	Access Type	Description
PFVFRE	31:0	0x0	RW	Bit j - Enables receiving packets to VF# (32*n+j). Each bit is cleared by the relevant VFLR.

8.2.4.22.7 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, on BME bit clear or on VF software reset. See VF Receive Enable -- PFVFRE / VF Transmit Enable -- PFVFTE for more details.

Field	Bit(s)	Init.	Access Type	Description
PFVFTE	31:0	0x0	RW	Bit j - Enables transmitting packets from VF# (32*n+j). Each bit is cleared by the relevant VFLR.

8.2.4.22.8 PF VM 0:31 Error Count Mask - PFVMECM0 (0x00008790)

Field	Bit(s)	Init.	Access Type	Description
FILTER	31:0	0x0	RW	Defines if a packet dropped from pools 0 to 31, respectively, is counted in the SSVPC counter.

8.2.4.22.9 PF VM 32:63 Error Count Mask - PFVMECM1 (0x00008794)

Field	Bit(s)	Init.	Access Type	Description
FILTER	31:0	0x0	RW	Defines if a packet dropped from pools 32 to 63 respectively is counted in the SSVPC counter.

8.2.4.22.10 PF Queue Drop Enable Register - PFQDE (0x00002F04)

Field	Bit(s)	Init.	Access Type	Description
PFQDE	0	0b	RW	Enable drop of packets from Rx queue Queue_Index. This bit overrides the SRRCTL.drop_en bit of each queue. For example, if either of the bits is set, a packet received when no descriptor is available is dropped.
RESERVED	3:1	0x0	RSV	Reserved (see WE and RE bits that follow).
RESERVED	7:4	0x0	RSV	Reserved.
QUEUE_INDEX	14:8	0x0	RW	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 3:0 are written into the relevant queue context. Bits 3:1 are reserved. This bit should never be set together with the RE bit in this register.

Field	Bit(s)	Init.	Access Type	Description
RE	17	0b	RW	Read Enable. When this bit is set, the content of bits 3:0 are read from the relevant queue context. Bits 3:1 are reserved. This bit should never be set together with the WE bit in this register.
RESERVED	31:18	0x0	RSV	Reserved.

8.2.4.22.11 PF VM Tx Switch Loopback Enable - PFVMTXSW[n] (0x00005180 + 0x4*n, n=0...1)

Field	Bit(s)	Init.	Access Type	Description
LLE	31:0	0x0	RW	Local Loopback Enable. For each register 'n', and bit 'i', i=0..31, enables Local loopback for pool 32*n+1. When set, a packet originating from a specific pool and destined to the same pool is allowed to be looped back. If cleared, the packet is dropped.

8.2.4.22.12 PFVF Anti Spoof Control - PFVFSPOOF[n] (0x00008200 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
MACAS	7:0	0x00	RW	For each register 'n', and bit 'i', i=0..7, Enables anti-spoofing filter on Ethernet MAC Addresses for VF(8*n+1).
VLANAS	15:8	0x00	RW	For each register 'n', and bit '8+i', i=0..7, Enables anti-spoofing filter on VLAN tag for VF(8*n+i). **Note:** If VLANAS is set for a specific pool, then the respective MACAS bit must be set as well.
RESERVED	31:16	0x00	RSV	Reserved.

8.2.4.22.13 PF DMA Tx General Switch Control - PFDTXGSWC (0x00008220)

Field	Bit(s)	Init.	Access Type	Description
LBE	0	0b	RW	Enables VMDQ Loopback.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.4.22.14 PF VM VLAN Insert Register - PFVMVIR[n] (0x00008000 + 0x4*n, n=0...63)

Field	Bit(s)	Init.	Access Type	Description
PORT_VLAN_ID	15:0	0x0	RW	Port VLAN tag to insert if the VLANA field = 01b.
RESERVED	29:16	0x0	RSV	Reserved.
VLANA	31:30	0x0	RW	VLAN Action: 00b = Use descriptor command. 01b = Always insert Default VLAN. 10b = Never insert VLAN. 11b = Reserved.

8.2.4.22.15 PF VM L2Control Register - PFVML2FLT[n] (0x0000F000 + 0x4*n, n=0...63)

This register controls per VM Inexact L2 Filtering.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	21:0	0x00	RSV	Reserved.
UPE	22	0b	RW	Unicast Promiscuous.
VPE	23	0b	RW	VLAN Promiscuous Enable.
AUPE	24	0b	RW	Accept Untagged Packets Enable. When set, packets without VLAN tag can be forwarded to this queue, assuming they pass the Ethernet MAC Address queuing mechanism.
ROMPE	25	0b	RW	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.
MPE	28	0b	RW	Multicast Promiscuous.
RESERVED	31:29	0x00	RSV	Reserved.

8.2.4.22.16 PF VM VLAN Pool Filter - PFVLVF[n] (0x0000F100 + 0x4*n, n=0...63)

Software should initialize these registers before transmit and receive are enabled.

Field	Bit(s)	Init.	Access Type	Description
VLAN_ID	11:0	X	RW	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.

8.2.4.22.17 PF VM VLAN Pool Filter Bitmap - PFVLVFB[n] (0x0000F200 + 0x4*n, n=0...127)

Software should initialize these registers before transmit and receive are enabled.

Field	Bit(s)	Init.	Access 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. Each bit when set, enables packet reception with the associated Pools as described below: Bit 'i' in register '2*n' is associated with POOL 'i' Bit 'i' in register '2*n+1' is associated with POOL '32+i'.

8.2.4.22.18 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. 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 beginning operation.

Field	Bit(s)	Init.	Access Type	Description
BIT_VECTOR	31:0	X	RW	Word wide bit vector specifying 32 bits in the unicast destination address filter table.

8.2.4.22.19 PF Mirror Rule Control - PFMRCTL[n] (0x0000F600 + 0x4*n, n=0...3)

This register defines mirroring rules for each of 4 destination pools.

Field	Bit(s)	Init.	Access Type	Description
VPME	0	0b	RW	Virtual Pool Mirroring Enable. Enables mirroring of certain pools as defined in the PFMRVM registers.
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.
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.

8.2.4.22.20 PF Mirror Rule VLAN - PFMRVLAN[n] (0x0000F610 + 0x4*n, n=0...7)

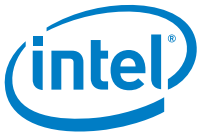
This register defines the VLAN values as listed in the PFVLVF table taking part in the VLAN mirror rule. Registers 0, 4 correspond to rule 0, registers 1, 5 correspond to rule 1, etc. Registers 0-3 correspond to the LSB in the PFVLVF table. For example, register 0 corresponds to VLAN filters 31:0, while register 4 corresponds to VLAN filters 63:32.

Field	Bit(s)	Init.	Access Type	Description
VLAN	31:0	0x0	RW	Bitmap listing which VLANs participate in the mirror rule.

8.2.4.22.21 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, 4 correspond to rule 0, registers 1, 5 correspond to rule 1, etc. Registers 0-3 correspond to the LSB in the pool list. For example, register 0 corresponds to pools 31:0, while register 4 corresponds to pools 63:32.

Field	Bit(s)	Init.	Access Type	Description
POOL	31:0	0x0	RW	Bitmap listing which pools participate in the mirror rule.



8.2.4.23 Diagnostic Registers

8.2.4.23.1 TX Buffer Access Control - TXBUFCTRL (0x0000C600)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	4:0	0x0	RSV	Reserved (accesses are 16-byte aligned).
ADD	18:5	0x0	RW	Address.
RESERVED	29:19	0x0	RSV	Reserved.
RDREQ	30	0b	RW	Read Request. Set to 1b to read the data from the Tx packet buffer address as defined by TXBUFCTRL.ADD, self clears at end of flow (data valid on TXBUFDATA[3:0] registers).
WRREQ	31	0b	RW	Write Request. Set to 1b to write the data to the Tx packet buffer address as defined by TXBUFCTRL.ADD, self clearing on end of flow (write data should be valid on TXBUFDATA[3:0] registers).

8.2.4.23.2 TX Buffer DATA - TXBUFDATA[n] (0x0000C610 + 0x4*n, n=0...3)

The Tx buffer is organized internally in lines of 128 bits. Each line is accessed by the 4 x TXBUFDATA registers. All 4 registers are used in a single read or write cycle.

Field	Bit(s)	Init.	Access Type	Description
DATA	31:0	X	RW	Reg 0 = Transmit Buffer data read/write bits 31:0. Reg 1 = Transmit Buffer data read/write bits 63:32. Etc....

8.2.4.23.3 Mailbox Memory CSR - MEM_CSR (0x00000740)

Field	Bit(s)	Init.	Access Type	Description
ECC_EN	0	1b	RW	Enables ECC on mailbox memory.
ECC_INVERT1	1	0b	RW	ECC invert1 for mailbox memory.
ECC_INVERT2	2	0b	RW	ECC invert2 for mailbox memory.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.4.23.4 Transmit Memories Control - TXRAMCTL (0x0000CF78)

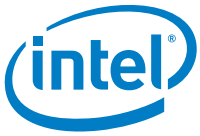
Field	Bit(s)	Init.	Access Type	Description
PB	2:0	0x3	RW	PB Memory RWM Input.
MNG	5:3	0x3	RW	MNG Buffer RWM Input.
RESERVED	14:6	0x0	RSV	Reserved.
FAST_SIM	15	0b	RW	Reserved.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.23.5 RX Buffer Access Control - RXBUFCTRL (0x00003600)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	4:0	0x0	RSV	Reserved (accesses are 32-byte aligned).
ADD	18:5	0x0	RW	Address.
RESERVED	29:19	0x0	RSV	Reserved.
RDREQ	30	0b	RW	Read Request. Set to 1b to read the data from the Rx packet buffer address as defined by RXBUFCTRL.ADD, self clears at end of flow (data valid on RXBUFDATA[7:0] registers).
WRREQ	31	0b	RW	Write Request. Set to 1b to write the data to the Rx packet buffer address as defined by RXBUFCTRL.ADD, self clearing on end of flow. (write data should be valid on RXBUFDATA[7:0] registers).

8.2.4.23.6 RX Write Buffer Pointers - RXWRPTR[n] (0x00003100 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
WRSTATUS HEAD	13:0	0x0	RO	Write status head offset in 32-byte resolution.
RESERVED	15:14	0x0	RSV	Reserved.
WRSTATUS TAIL	29:16	0x0	RO	Write status tail offset in 32-byte resolution.
RESERVED	31:30	0x0	RSV	Reserved.



8.2.4.23.7 RX Buffer Used Space - RXUSED[n] (0x00003120 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
USEDSPACE	19:0	0x0	RO	Current memory usage size in line resolution.
RESERVED	31:20	0x0	RSV	Reserved.

8.2.4.23.8 RX Read Buffer Pointers - RXRDPTR[n] (0x00003140 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
RDSTATUSHEAD	13:0	0x0	RO	Read status head offset in 32-byte resolution.
RESERVED	15:14	0x0	RSV	Reserved.
RDSTATUSTAIL	29:16	0x0	RO	Read status tail offset in 32 Byte resolution.
RESERVED	31:30	0x0	RSV	Reserved.

8.2.4.23.9 RX Read Write Offset Pointers - RXRDWRPTR[n] (0x00003160 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
RDDATAOFFSET	13:0	0x0	RO	Read data offset pointer.
RESERVED	15:14	0x0	RSV	Reserved.
WRDATAOFFSET	29:16	0x0	RO	Write data offset pointer.
RESERVED	31:30	0x0	RSV	Reserved.

8.2.4.23.10 TX Switch Buffer Error - TXSWERR (0x00003F80)

Field	Bit(s)	Init.	Access Type	Description
FREESPACE	9:0	0x0	RW	Intermediate buffer free space current state in 32-bit granularity.

Field	Bit(s)	Init.	Access Type	Description
LENERR	10	0b	RW	Sticky Bit. Loaded on receiving RX switch buffer fullness (should never happen).
FULL	11	0b	RW	Sticky Bit. Loaded on intermediate buffer full state.
RESERVED	31:12	0x0	RSV	Reserved.

8.2.4.23.11 TX Write Buffer Pointers - TXWRPTR[n] (0x0000C100 + 0x4*n, n=0...7)

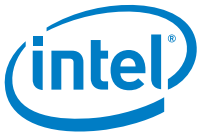
Field	Bit(s)	Init.	Access Type	Description
WRSTATUS HEAD	13:0	0x0	RO	Write status head offset in 32-byte resolution.
RESERVED	15:14	0x0	RSV	Reserved.
WRSTATUS TAIL	29:16	0x0	RO	Write status tail offset in 32-byte resolution.
RESERVED	31:30	0x0	RSV	Reserved.

8.2.4.23.12 TX Read Buffer Pointers - TXRDPTR[n] (0x0000C140 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
RDSTATUSH EAD	13:0	0x0	RO	Read status head offset in 32-byte resolution.
RESERVED	15:14	0x0	RSV	Reserved.
RDSTATUST AIL	29:16	0x0	RO	Read status tail offset in 32-byte resolution.
RESERVED	31:30	0x0	RSV	Reserved.

8.2.4.23.13 TX Read Write Offset Pointers - TXRDWRPTR[n] (0x0000C160 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
RDDATAOFF SET	13:0	0x0	RO	Read data offset pointer.



Field	Bit(s)	Init.	Access Type	Description
RESERVED	15:14	0x0	RSV	Reserved.
WRDATAOFFSET	29:16	0x0	RO	Write data offset pointer.
RESERVED	31:30	0x0	RSV	Reserved.

8.2.4.23.14 RX Buffer DATA - RXBUFDATA[n] (0x00003610 + 0x4*n, n=0...7)

The Rx buffer is organized internally in lines of 256 bits. Each line is accessed by the 8 x RXBUFDATA registers. All 8 registers are used in a single read or write cycle.

Field	Bit(s)	Init.	Access Type	Description
DATA	31:0	X	RW	Reg 0 - Receive Buffer data read/write bits 31:0Reg 1 - Receive Buffer data read/write bits 63:32Etc....

8.2.4.23.15 Receive Memories Control - RXRAMCTL (0x00003F78)

Field	Bit(s)	Init.	Access Type	Description
PB	2:0	0x3	RW	PB Memory RWM Input.
TXS	5:3	0x3	RW	TX Switch Buffer RWM Input.
STFIFO	8:6	0x3	RW	Status FIFO RWM Input.
PLMEM	11:9	0x3	RW	Pool MEM RWM Input.
FILMEM	14:12	0x3	RW	Filter MEM RWM Input.
FAST_SIM	15	0b	RW	Reserved.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.23.16 Receive DMA Memory Access Control Register - RDMAM (0x00002F30)

Note: The DMA-Rx must be idle (no Rx traffic) while accessing these registers.

Field	Bit(s)	Init.	Access Type	Description
ADDRESS	8:0	0x0	RW	Address of memory line to access.

Field	Bit(s)	Init.	Access Type	Description
DWORD_INDEX	12:9	0x0	RW	Dword Index Within the Line (one of 16 DWords):0x0 bits 31:0 (bytes 3:0). 0x1 bits 63:32 (bytes 7:4). . . . 0xF bits 511:480 (bytes 63:60).
MEMORY_SELECT	16:13	0x0	RW	Select RX DMA Memory Unit (must be set to 0x0 during normal operation). 0x1Tabdmarx_desc_completion_fifo. 0x2Tabdmarx_dfc_cmd_status_fifo. 0x3Tabdmarx_rsc_header_addr. 0x4Tabdmarx_tcn_status_ram. 0x5Tabdmarx_wb_collector_fifo. 0x6Tabdmarx_qsc_cnt_ram. 0x7Tabdmarx_qsc_fcoe_ram. 0x8Tabdmarx_qsc_queue_cnt_ram. 0xATabdmarx_qsc_queue_ram. 0xBTabdmarx_qsc_rsc_ram.
RESERVED	31:17	0x0	RSV	Reserved.

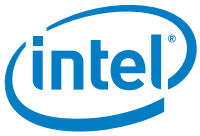
8.2.4.23.17 Receive DMA Memory Data Register - RDMAD (0x00002F34)

Note: The DMA-Rx must be idle (no Rx traffic) while accessing these registers.

Field	Bit(s)	Init.	Access Type	Description
DATA	31:0	X	RO	The register contains data - result of indirect read selected DMARX memory. Data is valid after write cycle to RDMAM register.

8.2.4.23.18 Receive DMA Status Register - RDRXSTAT (0x00002F14)

Field	Bit(s)	Init.	Access Type	Description
WR_FULL_ERR	0	0b	RW	tcmux write full error.
PIPE_MON_STOP	1	0b	RW	dmarx pipe mon qsc stop.
RESERVED	31:2	0x0	RSV	Reserved.



8.2.4.23.19 Receive DMA ECC Control 2 Register - DRECCCTL2 (0x00002F8C)

Field	Bit(s)	Init.	Access Type	Description
GLOBAL_ECC_ERROR_MASK	0	0b	RW	When set, blocks the ECC_ERROR indication from any DMA-RX memory.
SPECIFIC_ECC_ERROR_MASK	10:1	0x0	RW	When set, blocks the ECC_ERROR indication from the specific DMA-RX memory, bit 1: dmarx_wb_collector_fifobit 2: dmarx_desc_completion_fifobit 3: dmarx_dfc_cmd_status_fifobit 4: dmarx_rcs_header_addrbit 5: qsc_rsc_rambit 6: qsc_fcoe_rambit 7: qsc_queue_rambit 8: qsc_qu_cnt_rambit 9: qsc_vf_cnt_rambit 10: qsc_tcn_status_ram
RESERVED	31:11	0x0	RSV	Reserved.

8.2.4.23.20 Tx Descriptor Handler Memory Page Number - TDHMPN (0x000082FC)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	27:0	0x0	RSV	Reserved.
TDHFRZ	28	0b	RW	Descriptor handler machine freeze.
RESERVED	31:29	0x0	RSV	Reserved.

8.2.4.23.21 Tx Descriptor Handler Memory Read Data 0 - TIC_DW0 (0x000082B0)

Field	Bit(s)	Init.	Access Type	Description
TIC_DW0	31:0	0x0	RO	TX Data of iCache [31:0].

8.2.4.23.22 Tx Descriptor Handler Memory Read Data 1 - TIC_DW1 (0x000082B4)

Field	Bit(s)	Init.	Access Type	Description
TIC_DW1	31:0	0x0	RO	TX Data of iCache [63:32].

8.2.4.23.23 Tx Descriptor Handler Memory Read Data 2 - TIC_DW2 (0x000082B8)

Field	Bit(s)	Init.	Access Type	Description
TIC_DW2	31:0	0x0	RO	TX Data of iCache [95:64].

8.2.4.23.24 Tx Descriptor Handler Memory Read Data 3 - TIC_DW3 (0x000082BC)

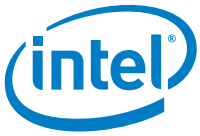
Field	Bit(s)	Init.	Access Type	Description
TIC_DW3	31:0	0x0	RO	TX Data of iCache [127:96].

8.2.4.23.25 Tx Descriptor Icache Indirect Access Register - TXDESCIC (0x000082CC)

Field	Bit(s)	Init.	Access Type	Description
ADDRESS	12:0	0x0	RW	Contains the icache address for read access.
RESERVED	30:13	0x0	RSV	Reserved.
READ_DONE	31	0b	RW	Icache Access Done. Hardware clears this bit when address is written to *Address* field. Hardware sets this bit when iCache read is done, and data is ready at TIC_DWx registers.

8.2.4.23.26 Tx DMA Performance IDLE Count - TXIDLE (0x000082A4)

Field	Bit(s)	Init.	Access Type	Description
SINGLECNT	15:0	0x0	RC	The counter counts the transitions from Idle to single state as long as the TXCYCLE counter does not equal 0xFFFF. The counter is cleared on read.
IDLECNT	31:16	0x0	RC	The counter counts the cycles on which the TxDMA is in idle for more then one cycle as long as the TXCYCLE counter does not equal 0xFFFF. The counter is cleared on read.



8.2.4.23.27 Tx DMA Performance Descriptor Fetch Count - TXDESC (0x000082A0)

Field	Bit(s)	Init.	Access Type	Description
BURSTCNT	15:0	0x0	RC	The counter counts the transitions from Idle to burst state as long as the TXCYCLE counter does not equal 0xFFFF. The counter is cleared on read.
DESCCNT	31:16	0x0	RC	The counter counts the number of descriptors fetched as long as the TXCYCLE counter does not equal 0xFFFF. The counter is cleared on read.

8.2.4.23.28 RT Tx Descriptor PB Space Counter - RTDTPBSPC[n] (0x00004970 + 0x4*n, n=0...3)

Note: 0x04970 _ 0x0497C

Field	Bit(s)	Init.	Access Type	Description
PB0_SPACE	15:0	0x0	RO	Used PB Space 0. This field in register 'n' (n=0...3), defines the used space (at a given time) in the Tx PB of TC 2xn in 16 bytes units. The DMA Tx increments the Pb0_space by the packet size when requesting it from the Dhost. The DBU decrements the Pb0_space by the packet size after it fetches the whole packet from the PB.
PB1_SPACE	31:16	0x0	RO	Used PB Space 1. This field in register 'n' (n=0...3), defines the used space (at a given time) in the Tx PB of TC 2xn+1 in 16 bytes units. The DMA Tx increments the Pb1_space by the packet size when requesting it from the Dhost. The DBU decrements the Pb1_space by the packet size after it fetches the whole packet from the PB.

8.2.4.23.29 FLEEP ECC - FLEEPECC (0x00010214)

Field	Bit(s)	Init.	Access Type	Description
ECCEN	0	1b	RW	Enables ECC for FLEEP memory.
INVERT0	1	0b	RW	Inverts bit 0 on next write cycle to the memory.
INVERT1	2	0b	RW	Inverts bit 1 on next write cycle to the memory.
UNCORERR	3	0b	RO	ECC uncorrectable error occurred. This bit is read only (write to this bit ignored).Bit cleared by read.
SRAMITR	4	0b	RW	Shadow RAM Interrupt.This bit reflects the Shadow RAM interrupt in case of uncorrectable ECC error. This bit is read only (write to this bit ignored).Bit cleared by read.

Field	Bit(s)	Init.	Access Type	Description
DISSRAMITR	5	0b	RW	Disable Shadow RAM Interrupt. Setting this bit disables the Shadow RAM interrupt in case of uncorrectable ECC error.
CORERR	6	0b	RW	ECC Correctable Error Occurred. This bit is read only (write to this bit ignored).
RESERVED	15:7	0x0	RSV	Reserved.
CORERRCNT	31:16	0x0000	RO	ECC Correctable Error Counter. This counter counts up to 0xFFFF. This counter is read only (writes to this counter ignored). Field cleared by read.

8.2.4.23.30 PCIe Diagnostic 1 - PCIE_DIAG1 (0x00011090)

Note: This register is shared for both LAN functions.

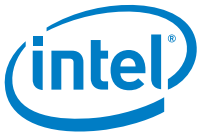
Field	Bit(s)	Init.	Access Type	Description
OFF1	15:0	0x0	RO	NVM GIO offset 1.
OFF2	31:16	0x0	RO	NVM GIO offset 2.

8.2.4.23.31 PCIe Diagnostic 2 - PCIE_DIAG2 (0x00011094)

Field	Bit(s)	Init.	Access Type	Description
OFF3	15:0	0x0	RO	NVM GIO offset 3.
OFF4	31:16	0x0	RO	NVM GIO offset 4.

8.2.4.23.32 PCIe Diagnostic 3 - PCIE_DIAG3 (0x00011098)

Field	Bit(s)	Init.	Access Type	Description
OFF5	15:0	0x0	RO	NVM GIO offset 5.
OFF6	31:16	0x0	RO	NVM GIO offset 6.



8.2.4.23.33 PCIe Diagnostic 4 - PCIE_DIAG4 (0x0001109C)

Field	Bit(s)	Init.	Access Type	Description
OFF7	15:0	0x0	RO	NVM GIO offset 7.
OFF8	31:16	0x0	RO	NVM GIO offset 8.

8.2.4.23.34 PCIe Diagnostic 5 - PCIE_DIAG5 (0x000110A0)

Field	Bit(s)	Init.	Access Type	Description
OFF9	15:0	0x0	RO	NVM GIO offset 9.
OFF10	31:16	0x0	RO	NVM GIO offset 10.

8.2.4.23.35 PCIe Diagnostic 6 - PCIE_DIAG6 (0x000110A4)

Field	Bit(s)	Init.	Access Type	Description
OFF11	15:0	0x0	RO	NVM GIO offset 11.
RESERVED	31:16	0x0	RSV	NVM GIO offset 12.

8.2.4.23.36 PCIe Diagnostic 7 - PCIE_DIAG7 (0x000110A8)

Field	Bit(s)	Init.	Access Type	Description
F0OFF1	15:0	0x0	RO	NVM GIO per func Func 0 odffset1.
F1OFF1	31:16	0x0	RO	NVM GIO per func Func 1 odffset1.

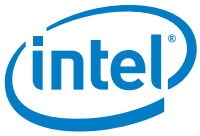
8.2.4.23.37 PCIe Diagnostic 8 - PCIE_DIAG8 (0x000110AC)

Field	Bit(s)	Init.	Access Type	Description
F0OFF2	15:0	0x0	RO	NVM GIO per func Func 0 odffset2.
F1OFF2	31:16	0x0	RO	NVM GIO per func Func 1 odffset2.

8.2.4.23.38 PCIe ECC FIX Status - PCIEECCSF (0x000110E4)

Note: This register is shared for both LAN functions.

Field	Bit(s)	Init.	Access Type	Description
ECC_ERR_CDQ0	0	0b	RW	ECC Read fix status for Completion Data Queue (CDQ) memory 0.
ECC_ERR_CDQ1	1	0b	RW1C	ECC Read fix status for CDQ memory 1.
ECC_ERR_CDQ2	2	0b	RW1C	ECC Read fix status for CDQ memory 2.
ECC_ERR_CDQ3	3	0b	RW1C	ECC Read fix status for CDQ memory 3.
ECC_ERR_CDQ4	4	0b	RW	ECC Read fix status for CDQ memory 4.
ECC_ERR_CDQ5	5	0b	RW	ECC Read fix status for CDQ memory 5.
ECC_ERR_CDQ6	6	0b	RW1C	ECC Read fix status for CDQ memory 6.
ECC_ERR_CDQ7	7	0b	RW1C	ECC Read fix status for CDQ memory 7.
ECC_ERR_ICLUT	8	0b	RW	ECC Read fix status for ICLUT memory.
ECC_ERR_PNP	9	0b	RW	ECC Read fix status for IB PNP memory.
ECC_ERR_WR_DATA0	10	0b	RW	ECC Read fix status for WR-REQ-Data memory 0.
ECC_ERR_WR_DATA1	11	0b	RW	ECC Read fix status for WR-REQ-Data memory 1.
ECC_ERR_WR_CMD0	12	0b	RW1C	ECC Read fix status for WR-REQ-Cmd memory 0.
ECC_ERR_WR_CMD1	13	0b	RW	ECC Read fix status for WR-REQ-Cmd memory 1.
ECC_ERR_RD_CMD0	14	0b	RW	ECC Read fix status for RD-REQ-Cmd memory 0.
ECC_ERR_RD_CMD1	15	0b	RW1C	ECC Read fix status for RD-REQ-Cmd memory 1.
ECC_ERR_MSIX	16	0b	RW	ECC Read fix status for MSIX memory.
RESERVED	31:17	0x0	RSV	Reserved.



8.2.4.23.39 Receive Filter Validation Register - RFVAL (0x000050A4)

Field	Bit(s)	Init.	Access Type	Description
MAC_FILTER_BAD_SCEN_1	0	0b	RO	This field indicates that a bad scenario of SOF with EOF occurred on the MAC Rx Filter interface.
MAC_FILTER_BAD_SCEN_2	1	0b	RO	This field indicates that a bad scenario of too small gap between packets occurred on the MAC Rx Filter interface.
RESERVED	31:2	0x0	RSV	

8.2.4.23.40 PCIe ECC Control 0 Register - PCIEECCCTL0 (0x00011100)

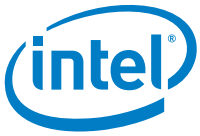
Field	Bit(s)	Init.	Access Type	Description
ECC_EN_TXTL	0	1b	RW	ECC enable for TXTL write-data memory.
ECC_INV1_TXTL	1	0b	RW	ECC invert1 for TXTL write-data memory.
ECC_INV2_TXTL	2	0b	RW	ECC invert2 for TXTL write-data memory.
ECC_EN_TXTL_RWCMD	3	1b	RW	ECC enable for TXTL write& read command memory.
ECC_INV1_TXTL_WCMD	4	0b	RW	ECC invert1 for TXTL write-command memory.
ECC_INV2_TXTL_WCMD	5	0b	RW	ECC invert2 for TXTL write-command memory.
ECC_INV1_TXTL_RCMD	6	0b	RW	ECC invert1 for TXTL read-command memory.
ECC_INV2_TXTL_RCMD	7	0b	RW	ECC invert2 for TXTL read-command memory.
ECC_EN_RXTL	8	1b	RW	ECC enable for RXTL all Completion Data Queue (CDQ) memories.
ECC_INV1_RXTL_MEM0	9	0b	RW	ECC invert1 for RXTL CDQ memory0.
ECC_INV2_RXTL_MEM0	10	0b	RW	ECC invert2 for RXTL CDQ memory0.

Field	Bit(s)	Init.	Access Type	Description
ECC_INV1_RXTL_MEM1	11	0b	RW	ECC invert1 for RXTL CDQ memory1.
ECC_INV2_RXTL_MEM1	12	0b	RW	ECC invert2 for RXTL CDQ memory1.
ECC_INV1_RXTL_MEM2	13	0b	RW	ECC invert1 for RXTL CDQ memory2.
ECC_INV2_RXTL_MEM2	14	0b	RW	ECC invert2 for RXTL CDQ memory2.
ECC_INV1_RXTL_MEM3	15	0b	RW	ECC invert1 for RXTL CDQ memory3.
ECC_INV2_RXTL_MEM3	16	0b	RW	ECC invert2 for RXTL CDQ memory3.
RESERVED	31:17	0x0	RSV	Reserved

8.2.4.23.41 PCIe ECC Control 1 Register - PCIEECCCTL1 (0x00011104)

Note: This register is shared for both LAN functions.

Field	Bit(s)	Init.	Access Type	Description
ECC_EN_ICLUT_PNP	0	1b	RW	ECC enable for RXTL ICLUT & PNP memories. This field is shared among all the functions.
ECC_INV1_ICLUT	1	0b	RW	ECC invert1 for RXTL ICLUT memory. This field is shared among all the functions.
ECC_INV2_ICLUT	2	0b	RW	ECC invert2 for RXTL ICLUT memory. This field is shared among all the functions.
ECC_INV1_PNP	3	0b	RW	ECC invert1 for RXTL PNP memory. This field is shared among all the functions.
ECC_INV2_PNP	4	0b	RW	ECC invert2 for RXTL PNP memory. This field is shared among all the functions.
ECC_EN_MSIX	5	1b	RW	ECC enable for MSIX memory. This field is shared among all the functions.
ECC_INV1_MSIX	6	0b	RW	ECC invert1 for MSIX memory. This field is shared among all the functions.
ECC_INV2_MSIX	7	0b	RW	ECC invert2 for MSIX memory. This field is shared among all the functions.
RESERVED	31:8	0x0	RSV	Reserved.



8.2.4.23.42 PCIe ECC Status Register - ECC_STATUS (0x000110E0)

Note: This register is shared for both LAN functions.

Field	Bit(s)	Init.	Access Type	Description
ECC_ERR_CDQ0	0	0b	RW	ECC Read error status for Completion Data Queue (CDQ) memory 0.
ECC_ERR_CDQ1	1	0b	RW1C	ECC Read error status for CDQ memory 1.
ECC_ERR_CDQ2	2	0b	RW	ECC Read error status for CDQ memory 2.
ECC_ERR_CDQ3	3	0b	RW	ECC Read error status for CDQ memory 3.
ECC_ERR_CDQ4	4	0b	RW1C	ECC Read error status for CDQ memory 4.
ECC_ERR_CDQ5	5	0b	RW1C	ECC Read error status for CDQ memory 5.
ECC_ERR_CDQ6	6	0b	RW	ECC Read error status for CDQ memory 6.
ECC_ERR_CDQ7	7	0b	RW1C	ECC Read error status for CDQ memory 7.
ECC_ERR_ICLUT	8	0b	RW	ECC Read error status for ICLUT memory.
ECC_ERR_PNP	9	0b	RW1C	ECC Read error status for IB PNP memory.
ECC_ERR_WR_DATA0	10	0b	RW1C	ECC Read error status for WR-REQ-Data memory 0.
ECC_ERR_WR_DATA1	11	0b	RW1C	ECC Read error status for WR-REQ-Data memory 1.
ECC_ERR_WR_CMD0	12	0b	RW	ECC Read error status for WR-REQ-Cmd memory 0.
ECC_ERR_WR_CMD1	13	0b	RW	ECC Read error status for WR-REQ-Cmd memory 1.
ECC_ERR_RD_CMD0	14	0b	RW	ECC Read error status for RD-REQ-Cmd memory 0.
ECC_ERR_RD_CMD1	15	0b	RW1C	ECC Read error status for RD-REQ-Cmd memory 1.
ECC_ERR_MSIX	16	0b	RW1C	ECC Read error status for MSIX memory.
RESERVED	31:17	0x0	RSV	Reserved.

8.2.4.23.43 PCIe Misc. Register - PCIEMISC (0x000110F0)

Note: This register is shared for both LAN functions.

Field	Bit(s)	Init.	Access Type	Description
DUMP_ON_ECC_ERROR	0	0b	RW	Mark a TLP with ECC error that cannot be fixed (two errors & more) with EDB (End-Bad) symbol, to the link layer.
ABORT_ON_BE_ERROR	1	0b	RW	Abort transaction in FUNC in case of BE error in memory transactions.
TO_EXTENSION	4:2	0x2	RW	Extension to FUNC timeout in memory transactions: 000b = Timeout: 0x03FF cycles (~4 µs). 001b = Timeout: 0x07FF cycles (~8 µs). 010b = Timeout: 0x0FFF cycles (~16 µs). 011b = Timeout: 0x1FFF cycles (~32 µs. 100b = Timeout: 0x3FFF cycles (~65 µs). 101b = Timeout: 0x7FFF cycles (~131 µs). 110b = Timeout: 0xFFFF cycles (~262 µs). 111b = Timeout: 0xFFFF cycles (~262 µs).
RESERVED	5	0b	RW	Reserved.
CHQRST	6	0b	RW	When set, the Completion Header Queue (CHQ) counters are reset faster on FUNC reset.
RESERVED	31:7	0x0	RSV	Reserved.

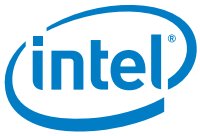
8.2.4.23.44 PCIe PF Pending request Register - PF_PEND (0x00011108)

Note: Cleared on PCIe* RST.

Field	Bit(s)	Init.	Access Type	Description
PF_PENDING_REQUEST	0	0b	RW	ICLUT indication for pending request of the PF.
RESERVED	31:1	0x0	RSV	Reserved.

8.2.4.23.45 PCIe VF Pending request Register High - VFH_PEND (0x00011110)

Note: Cleared on PCIe* RST.



Field	Bit(s)	Init.	Access Type	Description
VF_PENDING_REQUESTS	31:0	0x0	RW	ICLUT indication for pending request of VFs 32 to 63 (bits 0 to 31, respectively).

8.2.4.23.46 PCIe VF Pending request Register Low - VFL_PEND (0x0001110C)

Note: Cleared on PCIe* RST.

Field	Bit(s)	Init.	Access Type	Description
VF_PENDING_REQUESTS	31:0	0x0	RW	ICLUT indication for pending request of VFs 0 to 31 (bits 0 to 31, respectively).

8.2.4.23.47 Tx Memories ECC Register - TXDBUECC (0x0000CF70) DBU-TX

Field	Bit(s)	Init.	Access Type	Description
ECC_ENABLE	7:0	0x3	RW	Enables ECC. Each bit when set to 0x1 enables a memory. Bit 0 = mem_dbutx_pb_shell_10240x128. Bit 1 = mem_dbutx_mng_shell_256x128. Bit 7:2 = Reserved.
E1	15:8	0x0	RW	Inject Single Error Enable. When enabled, a single error is injected in the first write access to memory.
E2	23:16	0x0	RW	Inject Double Error Enable. When enabled two errors are injected into the first write access to memory.
RESERVED	31:24	0x0	RSV	Reserved.

8.2.4.23.48 SEC TX SA ECC Control - SECTXSAECC (0x00008814)

Field	Bit(s)	Init.	Access Type	Description
ECC_ENABLE	0	1b	RW	ECC Enable. When disabled, no ECC generation/check is performed.
E1	1	0b	RW	Inject Single Error. When set, a single error is injected in first write access to memory.

Field	Bit(s)	Init.	Access Type	Description
E2	2	0b	RW	Inject Double Error. When set, a double errors is injected in the first write access to memory.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.4.23.49 SEC TX Buffer ECC Control - SECTXBUFFECC (0x00008818)

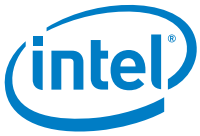
Field	Bit(s)	Init.	Access Type	Description
ECC_ENABLE	0	1b	RW	ECC Enable. When disabled, no ECC generation/check is performed.
E1	1	0b	RW	Inject Single Error. When set, a single error is injected in first write access to memory.
E2	2	0b	RW	Inject Double Error. When set, a double error is injected in the first write access to memory.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.4.23.50 SEC TX AES ECC Control - SECTXAESECC (0x0000881C)

Field	Bit(s)	Init.	Access Type	Description
ECC_ENABLE	0	1b	RW	ECC Enable. When disabled, no ECC generation/check is performed.
E1	1	0b	RW	Inject Single Error. When set, a single error is injected in first write access to memory.
E2	2	0b	RW	Inject Double Error. When set, a double errors is injected in the first write access to memory.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.4.23.51 SEC RX SA ECC Control - SECRXSAECC (0x00008D0C)

Field	Bit(s)	Init.	Access Type	Description
ECC_ENABLE	0	1b	RW	ECC Enable. When disabled, no ECC generation/check is performed.
E1	1	0b	RW	Inject Single Error. When set, a single error is injected in first write access to memory.



Field	Bit(s)	Init.	Access Type	Description
E2	2	0b	RW	Inject Double Error. When set, a double errors is injected in the first write access to memory.
RESERVED	31:3	0x0	RSV	Reserved.

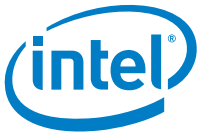
8.2.4.23.52 SEC RX AES ECC Control - SECRXAESECC (0x00008D10)

Field	Bit(s)	Init.	Access Type	Description
ECC_ENABLE	0	1b	RW	ECC Enable. When disabled, no ECC generation/check is performed.
E1	1	0b	RW	Inject Single Error. When set, a single error is injected in first write access to memory.
E2	2	0b	RW	Inject Double Error. When set, a double errors is injected in the first write access to memory.
RESERVED	31:3	0x0	RSV	Reserved.

8.2.4.23.53 Probe Select Register - PROBE_SEL (0x00016000)

Field	Bit(s)	Init.	Access Type	Description
BUS_SELECT	6:0	0x0	RW	Selects probe bus inside unit. See details in Fuses and Probe Mode Map.

Field	Bit(s)	Init.	Access Type	Description
UNIT_SELECT	10:7	0x0	RW	Selects unit for probe. Selection depends on bits 12:11. For top: 0x0 = car_comm 0x1 = car_pcie 0x2 = fleep 0x3 = mng 0x4 = dft_tap0x6 0x5 = Reserved 0x7 = mng 0xF:0x8 = Reserved. For each core: 0x0 = car_lan 0x1 = mac_cluster 0x2 = rx_filter 0x3 = dma_rx 0x4 = dma_tx 0x5 = dhost 0x6 = dbu_tx 0x7 = dbu_rx 0x8 = led 0x9 = stat_regs 0xA = target_logic 0xB = interrupt_block 0xC = sec_r x0xD = sec_t x0xF:0xE = Reserved.
CORE_SELECT	12:11	0x0	RW	Selects core or top for probe. 00b - top 01b - core_1 10b - core_0 11b - pcie_g2



Field	Bit(s)	Init.	Access Type	Description
CLOCK_SELECT	18:13	0x0	RW	Selects clock to be probed out with unit probe bus (TEST_POINT_CLK). Selection depends on bits 12:11. For top: 0x00 = car_comm_xtal_clk 0x01 = car_dft_prod_clk 0x02 = car_ee_clk 0x04:0x03 = Reserved. 0x05 = car_mii_clk 0x06 = car_mng_2core_clk_0 0x07 = car_mng_2core_clk_1 0x08 = car_mng_clk 0x09 = car_mng_inv_clk 0x0A = car_rmii_clk 0x0B = Reserved. 0x0C = Reserved. 0x0D = car_wake_dma_clk_0 0x0E =car_wake_dma_clk_1 0x3F:0F = Reserved. For each core: 0x00 = phy_pll_50_clk 0x01 = car_dma_clk 0x02 = car_dma_slow_clk 0x03 = car_dma_mem_clk 0x0F:0x04 = Reserved. 0x10 = car_mac_clk 0x11 = phy_mac_rx_xgmii_clk 0x16:0x12 = Reserved. 0x17 = car_secrx_clk 0x18 = car_sectx_clk 0x19 = car_wake_dma_clk 0x1A = car_wake_dma_mem_clk 0x1B = Reserved. 0x1C = car_comm_xtal_clk 0x1D = car_mng_2core_clk 0x3F:1E = Reserved.
RESERVED	31:19	0x0	RSV	

8.2.4.23.54 Probe Out Register - PROBE_OUT (0x00016038)

Field	Bit(s)	Init.	Access Type	Description
PROBE_OUTPUT	31:0	0x0	RW	Sampled Probe Mode Output Bus Value.

8.2.4.23.55 Receive DBU ECC - RXDBUECC (0x00003F70)

Note: For generating an ECC uncorrectable error, both Invert-0 and Invert-1 bits of the relevant memory must be set to 1b.

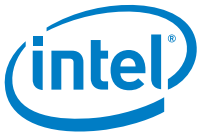
Field	Bit(s)	Init.	Access Type	Description
ECC_ENA	7:0	0x37	RW	Enables ECC for Rx DBU Memories. 0 = RX packet buffer bit. 1 = TX intermediate buffer bit. 2 = Pre-status shell bit. 3 = Reserved bit. 4 = Pool shell bit. 5 = Filter shell bit. 7:6 = Reserved bits.
INVERT_0	15:8	0x0	RW	Inverts bit 0 on next write cycle for each memory.
INVERT_1	23:16	0x0	RW	Inverts bit 1 on next write cycle for each memory.
RESERVED	31:24	0x0	RSV	Reserved.

8.2.4.23.56 DMA Rx ECC Status - DRECCSTAT (0x00002F0C)

Field	Bit(s)	Init.	Access Type	Description
ECC_ERROR	9:0	0x0	RW	ECC Error latch for RAM, cleared on write: 9 = qsc_rsc_ram_ecc_error_ltch 8 = qsc_fcoe_ram_ecc_error_ltch 7 = qsc_queue_ram_ecc_error_ltch 6 = qsc_qu_cnt_ram_ecc_error_ltch 5 = qsc_vf_cnt_ram_ecc_error_ltch 4 = qsc_tcn_status_ram_ecc_error_ltch 3 = dmarx_wb_collector_fifo_ecc_error_ltch 2 = dmarx_desc_completion_fifo_ecc_error_ltch 1 = dmarx_dfc_cmd_status_fifo_ecc_error_ltch 0 = dmarx_rcs_header_addr_ecc_error_ltch
RESERVED	31:10	0x0	RSV	

8.2.4.23.57 Receive DMA WB Offset - RDWBOFST (0x00002F18)

Field	Bit(s)	Init.	Access Type	Description
WBOFF	15:0	0x0	RW	Testability Descriptor WB Offset. The WB offset is defined in descriptor units. The WB address of an Rx descriptor equals to the Rx descriptor address plus WBOFF x 16 bytes. During normal operation the WBOFF must be set to 0x0.
RESERVED	31:16	0x0	RSV	Reserved.



8.2.4.23.58 DMA Rx ECC Control - DRECCCTL (0x00002F08) DMA-RX

Field	Bit(s)	Init.	Access Type	Description
ECC_EN0	0	1b	WO	dp ecc en dmarx wb collector FIFO.
INVERT0_0	1	0b	WO	dp ecc invert 1 dmarx wb collector fifo.
INVERT0_1	2	0b	WO	dp ecc invert 2 dmarx wb collector fifo.
ECC_EN1	3	1b	WO	dp ecc en dmarx desc completion fifo.
INVERT1_0	4	0b	WO	dp ecc invert 1 dmarx desc completion fifo.
INVERT1_1	5	0b	WO	dp ecc invert 2 dmarx desc completion fifo.
ECC_EN2	6	1b	WO	dp ecc en dmarx dfc cmd status fifo.
INVERT2_0	7	0b	WO	dp ecc invert 1 dmarx dfc cmd status fifo.
INVERT2_1	8	0b	WO	dp ecc invert 2 dmarx dfc cmd status fifo.
ECC_EN3	9	1b	WO	dp ecc en dmarx rcs header addr.
INVERT3_0	10	0b	WO	dp ecc invert 1 dmarx rcs header addr.
INVERT3_1	11	0b	WO	dp ecc invert 2 dmarx rcs header addr.
ECC_EN4	12	1b	WO	ecc en qsc rsc ram.
INVERT4_0	13	0b	WO	ecc invert 1 qsc rsc ram.
INVERT4_1	14	0b	WO	ecc invert 2 qsc rsc ram.
ECC_EN5	15	1b	WO	ecc en qsc fcoe ram.
INVERT5_0	16	0b	WO	ecc invert 1 qsc fcoe ram.
INVERT5_1	17	0b	WO	ecc invert 2 qsc fcoe ram.
ECC_EN6	18	1b	WO	ecc en qsc queue ram.
INVERT6_0	19	0b	WO	ecc invert 1 qsc queue ram.
INVERT6_1	20	0b	WO	ecc invert 2 qsc queue ram.
ECC_EN7	21	1b	WO	ecc en qsc qu cnt ram.
INVERT7_0	22	0b	WO	ecc invert 1 qsc qu cnt ram.
INVERT7_1	23	0b	WO	ecc invert 2 qsc qu cnt ram.
ECC_EN8	24	1b	WO	ecc en qsc vf cnt ram.
INVERT8_0	25	0b	WO	ecc invert 1 qsc vf cnt ram.
INVERT8_1	26	0b	WO	ecc invert 2 qsc vf cnt ram.

Field	Bit(s)	Init.	Access Type	Description
ECC_EN9	27	1b	WO	ecc en qsc tcn status ram.
INVERT9_0	28	0b	WO	ecc invert 1 qsc tcn status ram.
INVERT9_1	29	0b	WO	ecc invert 2 qsc tcn status ram.
RESERVED	30	0b	RSV	Reserved.
TGT_DESC_BPASS	31	0b	WO	qsc target dreccctl bypass lb.

8.2.4.23.59 Rx Filter ECC Err Insertion 0 - RXFECCERR0 (0x000051B8)

Field	Bit(s)	Init.	Access Type	Description
ECC_ENA	8:0	0x1FF	RW	Enable ECC Logic at 1b for the following memories: 0 = FCoE context. 1 = input FIFO. 2 = MTA memory. 3 = MPSAR memory. 4 = Reserved. 5 = PFUTA memory. 6 = VFTA memory. 7 = PFVLVFB memory. 8 = Wake up memory.
ECCFLT_EN	9	0b	RW	Filter ECC Error Indication Enablement. When set to 1b, enables the ECC-INT + the RXF- blocking during ECC-ERR in one of the Rx filter memories. At 0b, the ECC logic can still functions overcoming only single errors while dual or multiple errors may be ignored silently. **Note:** This bit field is also documented in Rx Filter ECC Err Insertion 0 - RXFECCERR0.
INVERT_0	18:10	0x0	RW	Inverts bit 0 on next write cycle for each memory 10 = FCoE context. 11 = input FIFO. 12 = MTA memory. 13 = MPSAR memory. 14 = Reserved. 15 = PFUTA memory. 16 = VFTA memory. 17 = PFVLVFB memory. 18 = Wake up memory.
RESERVED	19	0b	RSV	Reserved.

Field	Bit(s)	Init.	Access Type	Description
INVERT_1	28:20	0x0	RW	Inverts bit 1 on next write cycle for each memory. 20 = FCoE context. 21 = input FIFO. 22 = MTA memory. 23 = MPSAR memory. 24 = Reserved. 25 = PFUTA memory. 26 = VFTA memory. 27 = PFVLVFB memory. 28 = Wake up memory.
RESERVED	31:29	0x0	RSV	Reserved.

8.2.4.23.60 Rx Filter ECC Err Insertion 1 - RXFECCERR1 (0x000051C0)

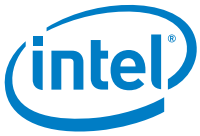
Field	Bit(s)	Init.	Access Type	Description
ECC_ENA	4:0	0x1F	RW	Enables ECC for RXF Flex-Filters Memories. 0 = Flex memorie0 - TCO 0,1. 1 = Flex memorie1 - TCO 2,3. 2 = Flex memorie2 - Wakeup 0,1. 3 = Flex memorie3 - Wakeup 2,3. 4 = Flex memorie4 - Wakeup 4,5.
RESERVED	7:5	0x0	RSV	Reserved.
INVERT_0	12:8	0x0	RW	Inverts bit 0 on next write cycle for each memory. 8 = Flex memorie0 - TCO 0,1. 9 = Flex memorie1 - TCO 2,3. 10 = Flex memorie2 - Wakeup 0,1. 11 = Flex memorie3 - Wakeup 2,3. 12 = Flex memorie4 - Wakeup 4,5.
RESERVED	15:13	0x0	RSV	Reserved.
INVERT_1	20:16	0x0	RW	Inverts bit 0 on next write cycle for each memory. 16 = Flex memorie0 - TCO 0,1. 17 = Flex memorie1 - TCO 2,3. 18 = Flex memorie2 - Wakeup 0,1. 19 = Flex memorie3 - Wakeup 2,3. 20 = Flex memorie4 - Wakeup 4,5.
RESERVED	31:21	0x0	RSV	Reserved.

8.2.4.23.61 Rx Filter ECC Correctable Status - RXFECCSTATC (0x000051C8)

Field	Bit(s)	Init.	Access Type	Description
CORRECTABLE_ERROR	13:0	0x0	RW	ECC Correctable Error latched for the following memories. Cleared on write 1's: 0 = FCoE context. 1 = input FIFO. 2 = MTA memory. 3 = MPSAR memory. 4 = Reserved. 5 = PFUTA memory. 6 = VFTA memory. 7 = PFVLVFB memory. 8 = Wake up memory. 9 = Flex memorie0 - TCO 0,1. 10 = Flex memorie1 - TCO 2,3. 11 = Flex memorie2 - Wakeup 0,1. 12 = Flex memorie3 - Wakeup 2,3. 13 = Flex memorie4 - Wakeup 4,5.
RESERVED	31:14	0x0	RSV	Reserved.

8.2.4.23.62 Rx Filter ECC Uncorrectable Status - RXFECCSTATUC (0x000051D0)

Field	Bit(s)	Init.	Access Type	Description
UN_CORRECTABLE_ERROR	13:0	0x0	RW1C	ECC Uncorrectable Error latched for the following memories. Cleared on write 1's: 0 = FCoE context. 1 = input FIFO. 2 = MTA memory. 3 = MPSAR memory. 4 = Reserved. 5 = PFUTA memory. 6 = VFTA memory. 7 = PFVLVFB memory. 8 = Wake up memory. 9 = Flex memorie0 - TCO 0,1. 10 = Flex memorie1 - TCO 2,3. 11 = Flex memorie2 - Wakeup 0,1. 12 = Flex memorie3 - Wakeup 2,3. 13 = Flex memorie4 - Wakeup 4,5.
RESERVED	31:14	0x0	RSV	Reserved.



8.2.4.23.63 Tx DMA QCN VM-ARB ECC - RTTDTECC (0x00004990)

Field	Bit(s)	Init.	Access Type	Description
QCN_PARITY_ENABLE	0	1b	RW	Enables parity for Tx DMA QCN and VM arbitration memories.
QCN_PARITY_INV_1	1	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
QCN_PARITY_INV_2	2	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
QCN_PARITY_INT	3	1b	RW	When set, ECC error for QCN memory triggers an interrupt.
RESERVED	7:4	0x0	RSV	Reserved.
VM_ARBITER_PARITY_ENABLE	8	1b	RW	Enables parity for VM arbiter memories.
VM_ARBITER_INV_1	9	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
VM_ARBITER_INV_2	10	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
VM_ARBITER_PARITY_INT	11	1b	RW	When set, parity error for VM arbiter memory triggers an interrupt.
RESERVED	31:12	0x0	RSV	Reserved.

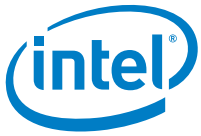
8.2.4.23.64 Tx DMA descriptor processor ECC - RTTDPECC (0x0000810C)

Field	Bit(s)	Init.	Access Type	Description
VLAN_MODE_ECC_ENABLE	0	1b	RW	Enables ECC for VLAN mode memories.
VLAN_MODE_ECC_INV_1	1	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
VLAN_MODE_ECC_INV_2	2	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
LSO_DESCRIPTORS_ECC_ENABLE	3	1b	RW	Enables ECC for LSO descriptors memories.

Field	Bit(s)	Init.	Access Type	Description
LSO_DESCRIPTORS_INV_1	4	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
LSO_DESCRIPTORS_INV_2	5	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
CONTEXT_STORE_ECC_ENABLE	6	1b	RW	Enables ECC for Context store memories.
CONTEXT_STORE_ECC_INV_1	7	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
CONTEXT_STORE_ECC_INV_2	8	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
LSO_HEADER_DESCRIPTORS_ECC_ENABLE	9	1b	RW	Enables ECC for LSO header descriptors memories.
LSO_HEADER_DESCRIPTORS_INV_1	10	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
LSO_HEADER_DESCRIPTORS_INV_2	11	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
VLAN_ECC_INT	12	1b	RW	When set, ECC error for VLAN memory triggers an interrupt.
LSO_DESCRIPTORS_ECC_INT	13	1b	RW	When set, ECC error for LSO descriptor memory triggers an interrupt.
CONTEXT_STORE_ECC_INT	14	1b	RW	When set, ECC error for Context store memory triggers an interrupt.
LSO_HEADER_ARBITER_ECC_INT	15	1b	RW	When set, ECC error for LSO header arbiter memory triggers an interrupt.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.23.65 Tx DMA ECC - TXDMAECC (0x00004A9C)

Field	Bit(s)	Init.	Access Type	Description
REGFILE_FIFO_ECC_ENABLE	0	1b	RW	Enables ECC for regfile FIFO memories.



Field	Bit(s)	Init.	Access Type	Description
REGFILE_FIFO_ECC_INV_1	1	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
REGFILE_FIFO_ECC_INV_2	2	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
DPTL_TEMP_ECC_ENABLE	3	1b	RW	Enables ECC for Dptl temp memories.
DPTL_TEMP_INV_1	4	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
DPTL_TEMP_INV_2	5	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
DH_ICACHE_ECC_ENABLE	6	1b	RW	Enables ECC for DH Icache memories.
DH_ICACHE_ECC_INV_1	7	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
DH_ICACHE_ECC_INV_2	8	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
DH_REGISTER_FILE_PARITY_ENABLE	9	1b	RW	Enables Parity for DH register file memories.
DH_REGISTER_FILE_PARITY_INV_1	10	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
DH_REGISTER_FILE_PARITY_INV_2	11	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
DPTL_TEMP_FIFO_ECC_INT	12	1b	RW	When set, ECC error for Dptl temp fifo memory triggers an interrupt.
DH_ICACHE_ECC_INT	13	1b	RW	When set, ECC error for DH icache memory triggers an interrupt.
DH_REGFILE_ECC_INT	14	1b	RW	When set, ECC error for DH regfile memory triggers an interrupt.
RESERVED	31:15	0x0	RSV	Reserved.

8.2.4.23.66 Tx DMA Statistics ECC - TXDMASTATECC (0x000087AC)

Field	Bit(s)	Init.	Access Type	Description
STAT_ECC_ENABLE	0	1b	RW	Enables ECC for DMA-TX statistics memory.
STAT_ECC_INV_1	1	0b	RW	Inverts bit 0 on next write cycle to the specific memories.
STAT_ECC_INV_2	2	0b	RW	Inverts bit 1 on next write cycle to the specific memories.
STAT_ECC_INT	3	1b	RW	When set, ECC error for statistics memory triggers an interrupt.
RESERVED	31:4	0x0	RSV	Reserved.

8.2.4.23.67 Tx DMA General ECC - TXDMAGENECC (0x00004AA0)

Field	Bit(s)	Init.	Access Type	Description
Reserved	0	1b	RW	Reserved.
Reserved	1	1b	RW	Reserved.
Reserved	2	1b	RW	Reserved.
Reserved	3	0b	RW	Reserved.
ECC_TRAFFIC_BLOCK	4	1b	RW	When set, all traffic from DMA-TX is halted upon ECC error.
RESERVED	31:5	0x0	RSV	Reserved

8.2.4.23.68 Tx Descriptor Handler Task Priority 0 - TDHTP_0 (0x00008280)

Field	Bit(s)	Init.	Access Type	Description
FREEZE	4:0	0x0	RW	Priority of freeze task.
WB_CONT	9:5	0x4	RW	Priority of Write back continuation task.
WB_COLL	14:10	0x5	RW	Priority of Write back collection task.
RX_DP_WR_H	19:15	0x1F	RW	Priority of rx_dp_wr_high task.

Field	Bit(s)	Init.	Access Type	Description
COMPLETION	24:20	0x7	RW	Priority of completion task.
TX_WB_ARB	29:25	0x8	RW	Priority of Tx Write back completion task.
RESERVED	31:30	0x0	RSV	Reserved.

8.2.4.23.69 Tx Descriptor Handler Task Priority 1 - TDHTP_1 (0x00008284)

Field	Bit(s)	Init.	Access Type	Description
RX_WB_ARB	4:0	0x1F	RW	Priority of Rx Write back completion.
RX_DP_WR_L	9:5	0x1F	RW	Priority of rx_dp_wr_low task.
RD_REQ_S	14:10	0x9	RW	Priority of read request short task.
RD_REQ_F	19:15	0xA	RW	Priority of read request fetch task. Fetch priority must always be higher than pre-fetch priority.
RD_REQ_PF	24:20	0xB	RW	Priority of read request fetch task.
TB	29:25	0xC	RW	Priority of tail bump task.
RESERVED	31:30	0x0	RSV	Reserved.

8.2.4.23.70 Tx Descriptor Handler Task Priority 2 - TDHTP_2 (0x00008288)

Field	Bit(s)	Init.	Access Type	Description
ADD	4:0	0x1	RW	Priority of add queue task.
REMOVE	9:5	0x2	RW	Priority of remove queue task.
ZERO_WB	14:10	0x3	RW	Priority of zero writeback clean task.
RX_DP_WR_SKIPTAB	19:15	0x1F	RW	Priority of Rx DP write skip task.
TX_DP_WR	24:20	0x6	RW	Priority of tx_dp_wr task.
EMPTY1	29:25	0x1F	RW	Priority of empty task 1.
RESERVED	31:30	0x0	RSV	Reserved.

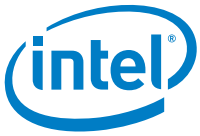
8.2.4.23.71 Tx Descriptor Handler Task Priority 3 - TDHTP_3 (0x0000828C)

Field	Bit(s)	Init.	Access Type	Description
EMPTY2	4:0	0x1F	RW	Priority of empty task 2.
EMPTY3	9:5	0x1F	RW	Priority of empty task 3.
RESERVED	31:10	0x0	RSV	Reserved.

8.2.4.23.72 Tx Descriptor Handler Task Enable - TDHTE (0x00008290)

Note: The PBTCWBCOL, PBTCCOMP, PBTCWBD, PBTCWBRS, PBTCWBTAIL bits enable performance of internal descriptor handler operations in parallel to software tail bumping process. Thus avoiding a performance impact when the tail is bumped frequently.

Field	Bit(s)	Init.	Access Type	Description
TASK_FREEZE	0	1b	RW	task_freeze
TASK_WB_CONTINUATION	1	1b	RW	task_wb_continuation
TASK_WB_COLLECTION	2	1b	RW	task_wb_collection
TASK_RX_DP_WR_HIGH	3	1b	RW	task_rx_dp_wr_high
TASK_COMPLETION	4	1b	RW	task_completion
TASK_TX_WB_ARBITRATIONTABTAB1	5	1b	RW	task_tx_wb_arbitration1
TASK_RX_WB_ARBITRATIONTABTAB1	6	1b	RW	task_rx_wb_arbitration1
TASK_RX_DP_WR_LOW	7	1b	RW	task_rx_dp_wr_low
TASK_RX_RD_REQ_SHORT_QTABTAB1	8	1b	RW	task_rx_rd_req_short_q1



Field	Bit(s)	Init.	Access Type	Description
TASK_RD_REQ_FETCH	9	1b	RW	task_rd_req_fetch
TASK_RD_REQ_PRE_FETCH	10	1b	RW	task_rd_req_pre_fetch
TASK_SW_HEAD_CHANGE	11	1b	RW	task_sw_head_change
TASK_ADD_Q	12	1b	RW	task_add_q
TASK_REMOVE_Q	13	1b	RW	task_remove_q
TASK_ZERO_WB_CLEAN	14	1b	RW	task_zero_wb_clean
TASK_RX_DP_WR_SKIP	15	1b	RW	task_rx_dp_wr_skip
TASK_TX_DP_WR	16	1b	RW	task_tx_dp_wr
TASK_EMPTY1	17	1b	RW	task_empty1
TASK_EMPTY2	18	1b	RW	task_empty2
TASK_EMPTY3	19	1b	RW	task_empty3
PBTCWBCOL	20	1b	RW	Piggy back tail change and write back collection.
PBTCCOMP	21	1b	RW	Piggy back tail change and completion.
PBTCWBD	22	1b	RW	Piggy back tail change and write back DMA.
PBTCWBRS	23	1b	RW	Piggy back tail change and write back on Request status bit set.
PBTCWBTAIL_TABTAB	24	1b	RW	Piggy back tail change and head write back.
FREEZE	25	0b	RW	Stop Descriptor Handler. When set, the descriptor handler stops at the next stable point. This enables reading of coherent values of all registers.
RESERVED	31:26	0x0	RSV	Reserved

8.2.4.23.73 Tx Descriptor Handler FSM States - TDH_STATES (0x00008294)

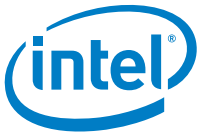
Field	Bit(s)	Init.	Access Type	Description
DH_STATE	15:0	0x0	RO	Bitmap that reports all the FSM states entered by the Descriptor Handler from reset. Clear on read. 0 = ALU_IDLE 1 = ALU_CALC_DSC_TO_FETCH 2 = ZERO_WB_CLEAN 3 = ALU_FIRST_RS_WB 4 = ALU_CALC_TAIL_WB 5 = ALU_SKIP_DIS_REQ 6 = ALU_INIT_ICACHE 7 = ALU_REMOVE_Q 8 = ALU_ADD_Q 9 = ALU_WBDMA_BURST_WR 10 = ALU_WBCOLC_BURST_RD 11 = ALU_DCDMA_BURST_RD 12 = ALU_RX_DP_BURST_WR 13 = ALU_TX_DP_BURST_WR 14 = ALU_RD_REQ_ALIGNER 15 = ALU_WR_REQ_ALIGNER
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.23.74 Last VM Misbehavior Cause - LVMMC (0x00008108)

Field	Bit(s)	Init.	Access Type	Description
LAST_MISBEHAVIOR	15:0	0x0	RW	Each unexpected behavior has a code designating the cause.
LAST_Q_MISBEHAVIOR_NUMBER	22:16	0x0	RC	The last transmit queue that showed unexpected behavior.
RESERVED	31:23	0x0	RSV	Reserved.

8.2.4.23.75 Tx Packet Buffer Flush Detect - TXMEMWRAP (0x0000C760)

Field	Bit(s)	Init.	Access Type	Description
TC0WRAP	2:0	0x0	RO	Packet Buffer 0 Wrap Around Counter. A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter wraps around to 000b on the next count.



Field	Bit(s)	Init.	Access Type	Description
TC0EMPTY	3	1b	RO	Packet Buffer 0 Empty. 0b = Packet buffer is not empty. 1b = Packet buffer is empty.
TC1WRAP	6:4	0x0	RO	Packet Buffer 1 Wrap Around Counter. A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter wraps around to 000b on the next count.
TC1EMPTY	7	1b	RO	Packet Buffer 1 Empty 0b = Packet buffer is not empty. 1b - Packet buffer is empty.
TC2WRAP	10:8	0x0	RO	Packet Buffer 2 Wrap Around Counter. A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter wraps around to 000b on the next count.
TC2EMPTY	11	1b	RO	Packet Buffer 2 Empty. 0b = Packet buffer is not empty. 1b = Packet buffer is empty.
TC3WRAP	14:12	0x0	RO	Packet Buffer 3 Wrap Around Counter. A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter wraps around to 000b on the next count.
TC3EMPTY	15	1b	RO	Packet Buffer 3 Empty. 0b = Packet buffer is not empty. 1b - Packet buffer is empty.
TC4WRAP	18:16	0x0	RO	Packet Buffer 4 Wrap Around Counter. A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter wraps around to 000b on the next count.
TC4EMPTY	19	1b	RO	Packet Buffer 4 Empty. 0b = Packet buffer is not empty. 1b = Packet buffer is empty.
TC5WRAP	22:20	0x0	RO	Packet Buffer 5 Wrap Around Counter. A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter wraps around to 000b on the next count.
TC5EMPTY	23	1b	RO	Packet Buffer 5 Empty. 0b = Packet buffer is not empty. 1b = Packet buffer is empty.
TC6WRAP	26:24	0x0	RO	Packet Buffer 6 Wrap Around Counter. A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter wraps around to 000b on the next count.
TC6EMPTY	27	1b	RO	Packet Buffer 6 Empty. 0b = Packet buffer is not empty. 1b = Packet buffer is empty.

Field	Bit(s)	Init.	Access Type	Description
TC7WRAP	30:28	0x0	RO	Packet Buffer 7 Wrap Around Counter. A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter wraps around to 000b on the next count.
TC7EMPTY	31	1b	RO	Packet Buffer 7 Empty. 0b = Packet buffer is not empty. 1b = Packet buffer is empty.

8.2.4.23.76 Transmit DBU ECC Counters - TXDBUEST (0x0000CF74)

Field	Bit(s)	Init.	Access Type	Description
CORCOUNT_0	6:0	0x0	RO	ECC Correctable Errors Count. For Tx packet buffer. clears on read, saturates on 127.
UNCORR_ERR0	7	0b	RO	ECC Uncorrectable Errors for Tx Packet Buffer. 1-bit latch until next read.
CORCOUNT_1	14:8	0x0	RO	ECC Correctable Errors Count. For Tx management buffer. clears on read, saturates on 127.
UNCORR_ERR1	15	0b	RO	ECC Uncorrectable Errors for Tx Intermediate Buffer. 1-bit latch until next read.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.23.77 Transmit Packet Buffer Used Space - TXPBUSED[n] (0x0000C120 + 0x4*n, n=0...7)

Field	Bit(s)	Init.	Access Type	Description
USED_LINES	15:0	0x0	RO	Number of used lines in the Tx packet buffer attached to TCn. Each line is 16 bytes long.
RESERVED	31:16	0x0	RSV	Reserved.

8.2.4.23.78 JTAG Clock Filter Register - TCK_FILTER_RATE (0x00016034)

Field	Bit(s)	Init.	Access Type	Description
JTAG_CLOCK_FILTER_RATE	7:0	0x04	RW	JTAG Clock Filter Rate.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:8	0x0	RSV	

8.3 Device Registers - VF

8.3.1 VF Registers Mapping in the PF Space

Abbreviation	Virtual Address	Physical Address
VFSRRCTL	0x00001014 + 0x40*n, n=0...7	0x00002100 + 0x4*n, n=0...15
VFPSRTYPE	0x00000300	0x00005480 + 0x4*n, n=0...15
VFDCA_RXCTRL	0x0000100C + 0x40*n, n=0...7	0x00002200 + 0x4*n, n=0...15

8.3.2 BAR3 Registers Summary

Table 8-3 BAR3 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Block	Section
MSI-X Register Summary VF - BAR 3				
0x00000000 + 0x10*n, n=0...2	MSIXTADD[n]	MSI-X Table Entry Lower Address	MSIX	Section 8.3.3.1.1
0x00000004 + 0x10*n, n=0...2	MSIXTUADD[n]	MSI-X Table Entry Upper Address	MSIX	Section 8.3.3.1.2
0x00000008 + 0x10*n, n=0...2	MSIXTMSG[n]	MSI-X Table Entry Message	MSIX	Section 8.3.3.1.3
0x0000000C + 0x10*n, n=0...2	MSIXVCTRL[n]	MSI-X Table Entry Vector Control	MSIX	Section 8.3.3.1.4
0x00002000	MSIXPBA	MSIXPBA Bit Description	MSIX	Section 8.3.3.1.5

8.3.3 Detailed Register Description - VF BAR3

8.3.3.1 MSI-X Register Summary VF - BAR 3

8.3.3.1.1 MSI-X Table Entry Lower Address - MSIXTADD[n] (0x00000000 + 0x10*n, n=0...2)

See MSI-X Table Structure for details of this register.

Field	Bit(s)	Init.	Access Type	Description
MSIXTADD10	1:0	0x0	RW	Message Address. For proper Dword alignment, software must always write zeros. to these two bits; otherwise, the result is undefined. The state of these bits after reset must be 0b. These bits are permitted to be read-only or read/write.
MSIXTADD	31:2	0x0	RW	Message Address. System-specified message lower address. For MSI-X messages, the contents of this field from an MSI-X table entry specifies the lower portion of the Dword-aligned address (AD[31:02]) for the memory write transaction. This field is read/write.

8.3.3.1.2 MSI-X Table Entry Upper Address - MSIXTUADD[n] (0x00000004 + 0x10*n, n=0...2)

See MSI-X Table Structure for details of this register.

Field	Bit(s)	Init.	Access 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.

8.3.3.1.3 MSI-X Table Entry Message - MSIXTMSG[n] (0x00000008 + 0x10*n, n=0...2)

See MSI-X Table Structure for details of this register.

Field	Bit(s)	Init.	Access 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.

8.3.3.1.4 MSI-X Table Entry Vector Control - MSIXVCTRL[n] (0x0000000C + 0x10*n, n=0...2)

See MSI-X Table Structure for details of this register.

Field	Bit(s)	Init.	Access 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.

8.3.3.1.5 MSIXPBA Bit Description - MSIXPBA (0x00002000)

Note: If a page size larger than 8 KB is programmed in the IOV structure, the address of the MSIX PBA table moves to be page aligned.

Field	Bit(s)	Init.	Access Type	Description
PENDING_BITS	2:0	0x0	RO	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.

8.3.4 BAR0 Registers Summary

Table 8-4 BAR0 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Block	Section
General Control Registers - VF				
0x00000000	VFCTRL	VF Control Register	Target	Section 8.3.5.1.1
0x00000008	VFSTATUS	VF Device Status Register	Target	Section 8.3.5.1.2
0x00000010	VFLINKS	VF Link Status Register	MAC	Section 8.3.5.1.3
0x00000048	VFFRTIMER	VF Free Running Timer	RX-Filter	Section 8.3.5.1.4
0x000002FC	VFMAILBOX	VF Mailbox	Target	Section 8.3.5.1.5
0x00000200 + 0x4*n, n=0...15	VFMBMEM[n]	VF Mailbox Memory	Target	Section 8.3.5.1.6
0x00003190	VFRXMEMWRAP	Rx Packet Buffer Flush Detect	DBU-RX	Section 8.3.5.1.7
Interrupt Registers - VF				
0x00000100	VFEICR	VF Extended Interrupt Cause	Interrupt	Section 8.3.5.2.1
0x00000104	VFEICS	VF Extended Interrupt Cause Set	Interrupt	Section 8.3.5.2.2
0x00000108	VFEIMS	VF Extended Interrupt Mask Set/ Read	Interrupt	Section 8.3.5.2.3
0x0000010C	VFEIMC	VF Extended Interrupt Mask Clear	Interrupt	Section 8.3.5.2.4
0x00000114	VFEIAM	VF Extended Interrupt Auto Mask Enable	Interrupt	Section 8.3.5.2.5
0x00000820 + 0x4*n, n=0...1	VFEITR[n]	VF Extended Interrupt Throttle Registers	Interrupt	Section 8.3.5.2.6
0x00000120 + 0x4*n, n=0...3	VFIVAR[n]	VF Interrupt Vector Allocation Registers	Interrupt	Section 8.3.5.2.7
0x00000140	VFIVAR_MISC	VF Interrupt Vector Allocation Registers Misc	Interrupt	Section 8.3.5.2.8
0x00000180 + 0x4*n, n=0...1	VFRSCINT[n]	VF RSC Enable Interrupt	Interrupt	Section 8.3.5.2.9
0x00000148	VFPBACL	VF MSI-X PBA Clear	PCIe	Section 8.3.5.2.10

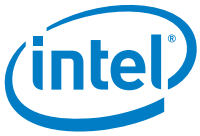


Table 8-4 BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
Receive DMA Registers - VF				
0x00001000 + 0x40*n, n=0...7	VFRDBAL[n]	VF Receive Descriptor Base Address Low	DMA-RX	Section 8.3.5.3.1
0x00001004 + 0x40*n, n=0...7	VFRDBAH[n]	VF Receive Descriptor Base Address High	DMA-RX	Section 8.3.5.3.2
0x00001008 + 0x40*n, n=0...7	VFRDLEN[n]	VF Receive Descriptor Length	DMA-RX	Section 8.3.5.3.3
0x00001010 + 0x40*n, n=0...7	VFRDH[n]	VF Receive Descriptor Head	DMA-RX	Section 8.3.5.3.4
0x00001018 + 0x40*n, n=0...7	VFRDT[n]	VF Receive Descriptor Tail	DMA-RX	Section 8.3.5.3.5
0x00001028 + 0x40*n, n=0...7	VFRXDCTL[n]	VF Receive Descriptor Control	DMA-RX	Section 8.3.5.3.6
0x00001014 + 0x40*n, n=0...7	VFSRRCTL[n]	VF Split and Replication Receive Control Registers	DMA-RX	Section 8.3.5.3.7
0x00000300	VFPSRTYPE	VF Replication Packet Split Receive Type Register	DBU-RX	Section 8.3.5.3.8
0x0000102C + 0x40*n, n=0...7	VFRSCCTL[n]	VF RSC Control	DMA-RX	Section 8.3.5.3.9
Transmit Registers - VF				
0x00002000 + 0x40*n, n=0...7	VFTDBAL[n]	VF Transmit Descriptor Base Address Low	DMA-TX	Section 8.3.5.4.1
0x00002004 + 0x40*n, n=0...7	VFTDBAH[n]	VF Transmit Descriptor Base Address High	DMA-TX	Section 8.3.5.4.2
0x00002008 + 0x40*n, n=0...7	VFTDLEN[n]	VF Transmit Descriptor Length	DMA-TX	Section 8.3.5.4.3
0x00002010 + 0x40*n, n=0...7	VFTDH[n]	VF Transmit Descriptor Head	DMA-TX	Section 8.3.5.4.4
0x00002018 + 0x40*n, n=0...7	VFTDT[n]	VF Transmit Descriptor Tail	DMA-TX	Section 8.3.5.4.5
0x00002028 + 0x40*n, n=0...7	VFTXDCTL[n]	VF Transmit Descriptor Control	DMA-TX	Section 8.3.5.4.6
0x00002038 + 0x40*n, n=0...7	VFTDWBAL[n]	VF Tx Descriptor Completion Write Back Address Low	DMA-TX	Section 8.3.5.4.7
0x0000203C + 0x40*n, n=0...7	VFTDWBAH[n]	VF Tx Descriptor Completion Write Back Address High	DMA-TX	Section 8.3.5.4.8
DCA Registers - VF				

Table 8-4 BAR0 Registers Summary (Continued)

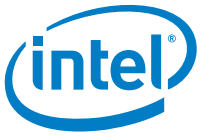
Offset / Alias Offset	Abbreviation	Name	Block	Section
0x0000100C + 0x40*n, n=0...7	VFDCA_RXCTRL[n]	VF Rx DCA Control Register	DMA-RX	Section 8.3.5.5.1
0x0000200C + 0x40*n, n=0...7	VFDCA_TXCTRL[n]	VF Tx DCA Control Registers	DMA-TX	Section 8.3.5.5.2
Statistic Register Descriptions - VF				
0x0000101C	VFGPRC	VF Good Packets Received Count	DMA-RX	Section 8.3.5.6.1
0x0000201C	VFGPTC	VF Good Packets Transmitted Count	STAT	Section 8.3.5.6.2
0x00001020	VFGORC_LSB	VF Good Octets Received Count Low	DMA-RX	Section 8.3.5.6.3
0x00001024	VFGORC_MSB	VF Good Octets Received Count High	DMA-RX	Section 8.3.5.6.4
0x00002020	VFGOTC_LSB	VF Good Octets Transmitted Count LSB	STAT	Section 8.3.5.6.5
0x00002024	VFGOTC_MSB	VF Good Octets Transmitted Count MSB	STAT	Section 8.3.5.6.6
0x00001034	VFMPRC	VF Multicast Packets Received Count	DMA-RX	Section 8.3.5.6.7

8.3.5 Detailed Register Description - VF BAR0

8.3.5.1 General Control Registers - VF

8.3.5.1.1 VF Control Register - VFCTRL (0x00000000)

Field	Bit(s)	Init.	Access 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.



8.3.5.1.2 VF Device Status Register - VFSTATUS (0x00000008; RO)

This register is a mirror of the PF status register. See Device Status Register - STATUS (0x00008; RO) for details of this register.

Fields definitions are the same as defined on Section 8.2.4.1.3.

8.3.5.1.3 VF Link Status Register - VFLINKS (0x00000010)

This register is a mirror of the PF LINKS register. See Link Status Register - LINKS (0x042A4; RO) for details of this register.

Field	Bit(s)	Init.	Access Type	Description
FIFO_UNDERRUN	0	0b	RO	Indicates underrun condition in MAC elastic FIFO.
FIFO_OVERRUN	1	0b	RO	Indicates overrun condition in MAC elastic FIFO.
RF_STATE	2	0b	RO	MAC is in remote fault state.
LF_STATE	3	0b	RO	MAC is in local fault state.
RESERVED	6:4	0x0	RSV	Reserved.
LINK_STATUS	7	0b	RO	1b = Link is up and there was no link down from last time read. 0b = Link is currently down or link was down since last time read. Self cleared upon read if the link is low and set if the link is up.
RESERVED	27:8	0x0	RSV	Reserved.
LINK_SPEED	29:28	X	RO	MAC Link Speed Status 00b = Reserved. 01b = 100 Mb/s. 10b = 1 Gb/s. 11b = 10 Gb/s.
LINK_UP	30	0b	RO	Link up 1b = Link is up. 0b = Link is down.
RESERVED	31	0b	RSV	Reserved.

8.3.5.1.4 VF Free Running Timer - VFFRTIMER (0x00000048; RO)

This register mirrors the value of a free running timer register in the PF - FRTIMER. The register is reset by a PCI reset and/or software reset. This register is a mirror of the PF register.

Fields definitions are the same as defined on Section 8.1.4.14.2.

8.3.5.1.5 VF Mailbox - VFMAILBOX (0x000002FC)

This register is cleared by VLFR (excepted to RSTI bit).

Field	Bit(s)	Init.	Access Type	Description
REQ	0	0b	WO	Request for PF Ready. Setting this bit, causes an interrupt to the PF. This bit always reads as zero. Setting this bit sets the corresponding bit in VFREQ field in PFMBICR register.
ACK	1	0b	WO	PF Message Received. Setting this bit, causes an interrupt to the PF. This bit always reads as zero. Setting this bit sets the corresponding bit in VFACK field in PFMBICR register.
VFU	2	0b	RW	Buffer is Taken by VF. This bit can be set only if the PFU bit is cleared and is mirrored in the VFU bit of the PFMailbox register.
PFU	3	0b	RW	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 wrote a message in the mailbox.
PFACK	5	0b	RC	PF acknowledged the VF previous message.
RSTI	6	1b	RO	Indicates that the PF had reset the shared resources and the reset sequence is in progress. This bit is not affected by VFLR.
RSTD	7	0b	RC	Indicates that a PF software reset completed.
RESERVED	31:8	0x0	RSV	Reserved.

8.3.5.1.6 VF Mailbox Memory - VFMBMEM[n] (0x00000200 + 0x4*n, n=0...15)

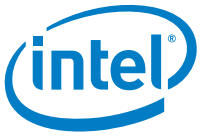
Mailbox memory for PF and VF drivers communication. The mailbox size for each VM is 64 bytes accessed by 16 x 32-bit registers. Locations can be accessed as 32-bit or 64-bit words.

Field	Bit(s)	Init.	Access Type	Description
MAILBOX_DATA	31:0	X	RW	Mailbox data composed of 16 x 4 byte registers.

8.3.5.1.7 Rx Packet Buffer Flush Detect - VFRXMEMWRAP (0x00003190; RO)

This register mirrors the PF RXMEMWRAP described in Rx Packet Buffer Flush Detect - RXMEMWRAP (0x03190; RO).

Fields definitions are the same as defined on Section 8.2.4.8.12.



8.3.5.2 Interrupt Registers - VF

8.3.5.2.1 VF Extended Interrupt Cause - VFEICR (0x00000100)

Field	Bit(s)	Init.	Access Type	Description
MSIX	2:0	0x0	RW1C	Indicates an interrupt cause mapped to MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.5.2.2 VF Extended Interrupt Cause Set - VFEICS (0x00000104)

Field	Bit(s)	Init.	Access Type	Description
MSIX	2:0	0x0	WO	Sets to corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.5.2.3 VF Extended Interrupt Mask Set/Read - VFEIMS (0x00000108)

Field	Bit(s)	Init.	Access Type	Description
MSIX	2:0	0x0	RWS	Set mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.5.2.4 VF Extended Interrupt Mask Clear - VFEIMC (0x0000010C)

Field	Bit(s)	Init.	Access Type	Description
MSIX	2:0	0x0	WO	Clear mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.5.2.5 VF Extended Interrupt Auto Mask Enable - VFEIAM (0x00000114)

Field	Bit(s)	Init.	Access Type	Description
MSIX	2:0	0x0	RW	Auto mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

8.3.5.2.6 VF Extended Interrupt Throttle Registers - VFEITR[n] (0x00000820 + 0x4*n, n=0...1; RWS)

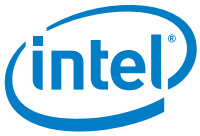
See register description in Extended Interrupt Throttle Registers - EITR[n].

Fields definitions are the same as defined on Section 8.2.4.5.16.

8.3.5.2.7 VF Interrupt Vector Allocation Registers - VFIVAR[n] (0x00000120 + 0x4*n, n=0...3)

These registers map interrupt causes into MSI-X vectors. See additional details in Mapping of Interrupt Causes.Transmit and receive queues mapping to VFIVAR registers is shown in the sections that follow.

Field	Bit(s)	Init.	Access Type	Description
INT_ALLOC_0	0	0b	RW	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	Valid bit for INT_Alloc[0].
INT_ALLOC_1	8	0b	RW	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	Valid bit for INT_Alloc[1].
INT_ALLOC_2	16	0b	RW	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	Valid bit for INT_Alloc[2].
INT_ALLOC_3	24	0b	RW	Defines the MSI-X vector (0 or 1) assigned to Tx queue '2*N+1' for IVAR 'N' register (N=0...3).



Field	Bit(s)	Init.	Access Type	Description
RESERVED	30:25	0x0	RSV	Reserved.
INT_ALLOC_VAL_3	31	0b	RW	Valid bit for INT_Alloc[3].

8.3.5.2.8 VF Interrupt Vector Allocation Registers Misc - VFIVAR_MISC (0x00000140)

This register maps the mail-box interrupt into MSI-X vector. See additional details in Mapping of Interrupt Causes.

Field	Bit(s)	Init.	Access Type	Description
INT_ALLOC_0	1:0	X	RW	Defines the MSI-X vector assigned to the mail-box interrupt.
RESERVED	6:2	0x0	RSV	Reserved.
INT_ALLOC_VAL_0	7	0b	RW	Valid bit for INT_Alloc[0].
RESERVED	31:8	0x0	RSV	Reserved.

8.3.5.2.9 VF RSC Enable Interrupt - VFRSCINT[n] (0x00000180 + 0x4*n, n=0...1; RW)

See register description in RSC Enable Interrupt - RSCINT[n].

Fields definitions are the same as defined on Section 8.2.4.5.23.

8.3.5.2.10 VF MSI-X PBA Clear - VFPBACL (0x00000148)

Field	Bit(s)	Init.	Access Type	Description
PENBIT	2:0	0x0	RW1C	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.

8.3.5.3 Receive DMA Registers - VF

8.3.5.3.1 VF Receive Descriptor Base Address Low - VFRDBAL[n] (0x00001000 + 0x40*n, n=0...7; RW)

See RDBAL description in Receive Descriptor Base Address Low - RDBAL[n].

Fields definitions are the same as defined on Section 8.2.4.8.1.

8.3.5.3.2 VF Receive Descriptor Base Address High - VFRDBAH[n] (0x00001004 + 0x40*n, n=0...7; RW)

See RDBAH description in Receive Descriptor Base Address High - RDBAH[n].

Fields definitions are the same as defined on Section 8.2.4.8.2.

8.3.5.3.3 VF Receive Descriptor Length - VFRDLEN[n] (0x00001008 + 0x40*n, n=0...7; RW)

See RDLEN description in Receive Descriptor Length - RDLEN[n].

Fields definitions are the same as defined on Section 8.2.4.8.3.

8.3.5.3.4 VF Receive Descriptor Head - VFRDH[n] (0x00001010 + 0x40*n, n=0...7; RO)

See RDH description in Receive Descriptor Head -- RDH[n].

Fields definitions are the same as defined on Section 8.2.4.8.4.

8.3.5.3.5 VF Receive Descriptor Tail - VFRDT[n] (0x00001018 + 0x40*n, n=0...7; RW)

See RDT description in Receive Descriptor Tail - RDT[n].

Fields definitions are the same as defined on Section 8.2.4.8.5.

8.3.5.3.6 VF Receive Descriptor Control - VFRXDCTL[n] (0x00001028 + 0x40*n, n=0...7; RW)

See RXDCTL description in Receive Descriptor Control - RXDCTL[n].

Fields definitions are the same as defined on Section 8.2.4.8.6.

8.3.5.3.7 VF Split and Replication Receive Control Registers - VFSRRCTL[n] (0x00001014 + 0x40*n, n=0...7; RW)

See SRRCTL description in Split Receive Control Registers - SRRCTL[n].

Fields definitions are the same as defined on Section 8.2.4.8.7.

8.3.5.3.8 VF Replication Packet Split Receive Type Register - VFPSRTYPE (0x00000300; RW)

See PSRTYPE description in Packet Split Receive Type Register - PSRTYPE[n].

Fields definitions are the same as defined on Section 8.2.4.7.4.

8.3.5.3.9 VF RSC Control - VFRSCCTL[n] (0x0000102C + 0x40*n, n=0...7; RW)

See RSCCTL description in RSC Control - RSCCTL[n].

Fields definitions are the same as defined on Section 8.2.4.8.13.

8.3.5.4 Transmit Registers - VF

8.3.5.4.1 VF Transmit Descriptor Base Address Low - VFTDBAL[n] (0x00002000 + 0x40*n, n=0...7; RW)

See TDBAL description in Transmit Descriptor Base Address Low - TDBAL[n].

Fields definitions are the same as defined on Section 8.2.4.9.5.

8.3.5.4.2 VF Transmit Descriptor Base Address High - VFTDBAH[n] (0x00002004 + 0x40*n, n=0...7; RW)

See TDBAH description in Transmit Descriptor Base Address High - TDBAH[n].

Fields definitions are the same as defined on Section 8.2.4.9.6.

8.3.5.4.3 VF Transmit Descriptor Length - VFTDLEN[n] (0x00002008 + 0x40*n, n=0...7; RW)

See TDLEN description in Transmit Descriptor Length - TDLEN[n].

Fields definitions are the same as defined on Section 8.2.4.9.7.

8.3.5.4.4 VF Transmit Descriptor Head - VFTDH[n] (0x00002010 + 0x40*n, n=0...7; RO)

See TDH description in Transmit Descriptor Head - TDH[n].

Fields definitions are the same as defined on Section 8.2.4.9.8.

8.3.5.4.5 VF Transmit Descriptor Tail - VFTDT[n] (0x00002018 + 0x40*n, n=0...7; RW)

See TDT description in Transmit Descriptor Tail - TDT[n].

Fields definitions are the same as defined on Section 8.2.4.9.9.

8.3.5.4.6 VF Transmit Descriptor Control - VFTXDCTL[n] (0x00002028 + 0x40*n, n=0...7; RW)

See TXDCTL description in Transmit Descriptor Control -- TXDCTL[n].

Fields definitions are the same as defined on Section 8.2.4.9.10.

8.3.5.4.7 VF Tx Descriptor Completion Write Back Address Low - VFTDWBAL[n] (0x00002038 + 0x40*n, n=0...7; RW)

See TDWBAL description in Tx Descriptor Completion Write Back Address Low - TDWBAL[n].

Fields definitions are the same as defined on Section 8.2.4.9.11.

8.3.5.4.8 VF Tx Descriptor Completion Write Back Address High - VFTDWBAH[n] (0x0000203C + 0x40*n, n=0...7; RW)

See TDWBAH description in Tx Descriptor Completion Write Back Address High - TDWBAH[n].

Fields definitions are the same as defined on Section 8.2.4.9.12.

8.3.5.5 DCA Registers - VF

8.3.5.5.1 VF Rx DCA Control Register - VFDCA_RXCTRL[n] (0x0000100C + 0x40*n, n=0...7; RW)

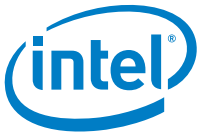
See DCA_RXCTRL description in Rx DCA Control Register - DCA_RXCTRL[n].

Fields definitions are the same as defined on Section 8.2.4.11.1.

8.3.5.5.2 VF Tx DCA Control Registers - VFDCA_TXCTRL[n] (0x0000200C + 0x40*n, n=0...7; RW)

See DCA_TXCTRL description in Tx DCA Control Registers - DCA_TXCTRL[n].

Fields definitions are the same as defined on Section 8.2.4.11.3.



8.3.5.6 Statistic Register Descriptions - VF

Registers in this section are RO by VF and RW by PF. Statistics are reset by PF clearing the register.

8.3.5.6.1 VF Good Packets Received Count - VFGPRC (0x0000101C)

Field	Bit(s)	Init.	Access Type	Description
VFGPRC	31:0	0x0	RW	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.

8.3.5.6.2 VF Good Packets Transmitted Count - VFGPTC (0x0000201C)

Field	Bit(s)	Init.	Access Type	Description
VFGPTC	31:0	0x0	RO	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.

8.3.5.6.3 VF Good Octets Received Count Low - VFGORC_LSB (0x00001020)

Field	Bit(s)	Init.	Access Type	Description
VFGORC_LSB	31:0	0x0	RW	Number of good octets received (32 LSBs 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. 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.

8.3.5.6.4 VF Good Octets Received Count High - VFGORC_MSB (0x00001024)

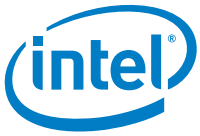
Field	Bit(s)	Init.	Access Type	Description
VFGORC_MSB	3:0	0x0	RW	Number of good octets received (4 MSBs of a 36-bit counter) by the queues allocated to this VF. See complete explanation in the description of the VFGORC_LSB register. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0.
RESERVED	31:4	0x0	RSV	Reserved.

8.3.5.6.5 VF Good Octets Transmitted Count LSB - VFGOTC_LSB (0x00002020)

Field	Bit(s)	Init.	Access Type	Description
VFGOTC_LSB	31:0	0x0	RO	Number of good octets transmitted (32 LSBits of a 36-bit counter) by the queues allocated to this VF. This register includes bytes transmitted in a packet from the <Destination Address> field through the <CRC> field, inclusively. This register counts octets of the packets counted by the VFGPTC register. Octets are counted on the VF interface rather than on the network interface.The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.5.6.6 VF Good Octets Transmitted Count MSB - VFGOTC_MSB (0x00002024)

Field	Bit(s)	Init.	Access Type	Description
VFGOTC_MSB	3:0	0x0	RO	Number of good octets transmitted (4 MSBs of a 36-bit counter) by the queues allocated to this VF. See complete explanation in the description of the VFGOTC-LSB register. 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.



8.3.5.6.7 VF Multicast Packets Received Count - VFMPRC (0x00001034)

Field	Bit(s)	Init.	Access Type	Description
VFMPRC	31:0	0x0	RO	Number of multicast good packets received by this VF (of any length) that pass 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 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.

8.4 Device Registers - PCI

8.4.1 PCI PHY Registers Summary

Table 8-5 PCI PHY Registers Summary

Offset / Alias Offset	Abbreviation	Name	Block	Section
Link/PHY Registers				
0x00000004	INITDISABLE	Init Disable	PCI PHY	Section 8.4.2.1.1
0x00000038	CLKGATE	Clock Gate	PCI PHY	Section 8.4.2.1.2
0x0000005C	BER	BER Meter Locked Register	PCI PHY	Section 8.4.2.1.3
0x00000060	SRBER0	BER Meter for Lane 0	PCI PHY	Section 8.4.2.1.4
0x00000064	SRBER1	BER Meter for Lane 1	PCI PHY	Section 8.4.2.1.5
0x00000068	SRBER2	BER Meter for Lane 2	PCI PHY	Section 8.4.2.1.6
0x0000006C	SRBER3	BER Meter for Lane 3	PCI PHY	Section 8.4.2.1.7
0x00000070	SRBER4	BER Meter for Lane 4	PCI PHY	Section 8.4.2.1.8
0x00000074	SRBER5	BER Meter for Lane 5	PCI PHY	Section 8.4.2.1.9

Table 8-5 PCI PHY Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00000078	SRBER6	BER Meter for Lane 6	PCI PHY	Section 8.4.2.1.10
0x0000007C	SRBER7	BER Meter for Lane 7	PCI PHY	Section 8.4.2.1.11
0x00000090	NFTS	NTFS Register	PCI PHY	Section 8.4.2.1.12
0x0000009C	REUTENGLSSEL0	REUTENGLSSEL0 Register	PCI PHY	Section 8.4.2.1.13
0x000000A0	REUTERRCED0	CRC Error Count Pport 0	PCI PHY	Section 8.4.2.1.14
0x000000A4	REUTE0	RCV Error Count	PCI PHY	Section 8.4.2.1.15
0x000000A8	REUTENGLTRON0	REUTENGLTRON0 Register	PCI PHY	Section 8.4.2.1.16
0x000000AC	GENTRANS	Speed Timeout Register	PCI PHY	Section 8.4.2.1.17
0x000000E8	CFGWAITACCEPT	Config.Lanenum.WaitAccept Register	PCI PHY	Section 8.4.2.1.18
0x00000114	TRAPPER0	Trapper Event Reg dw0	PCI PHY	Section 8.4.2.1.19
0x00000118	TRAPPER1	Trapper Event Reg dw1	PCI PHY	Section 8.4.2.1.20
0x0000011C	TRAPPER2	Trapper Event Reg dw2	PCI PHY	Section 8.4.2.1.21
0x00000120	TRAPPER3	Trapper Event Reg dw3	PCI PHY	Section 8.4.2.1.22
0x00000124	TRAPPERM0	Trapper Event Mask dw0	PCI PHY	Section 8.4.2.1.23
0x00000128	TRAPPERM1	Trapper Event Mask dw1	PCI PHY	Section 8.4.2.1.24
0x0000012C	TRAPPERM2	Trapper Event Mask dw2	PCI PHY	Section 8.4.2.1.25
0x00000130	TRAPPERM3	Trapper Event Mask dw3	PCI PHY	Section 8.4.2.1.26
0x00000134	TRAPPERC1	Trapped Event Counter	PCI PHY	Section 8.4.2.1.27
0x00000138	TRAPPERC2	Trapper Counter Clear	PCI PHY	Section 8.4.2.1.28

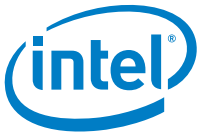


Table 8-5 PCI PHY Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00000140	PIPE	PIPE Register	PCI PHY	Section 8.4.2.1.29
0x00000144	CHKN	CHKN Register	PCI PHY	Section 8.4.2.1.30
0x00000160	REUTMCTR20	REUTMCTR20 Register	PCI PHY	Section 8.4.2.1.31
0x00000174	BERTESTSYM0	BER Meter Symbol Pattern 0	PCI PHY	Section 8.4.2.1.32
0x00000178	BERTESTSYM1	BER Meter Symbol Pattern 1	PCI PHY	Section 8.4.2.1.33
0x00000180	BERCFG1	BER Config 1	PCI PHY	Section 8.4.2.1.34
0x00000184	BERCFG2	BER Config 2	PCI PHY	Section 8.4.2.1.35
0x00000190	PLLLOCK	PLL Lock Register	PCI PHY	Section 8.4.2.1.36
0x00000194	EN	SWZL Register	PCI PHY	Section 8.4.2.1.37
0x0000019C	PCIE	Soft Reset Register	PCI PHY	Section 8.4.2.1.38
0x000001A8	FELB	Far-end Loopback Register	PCI PHY	Section 8.4.2.1.39
0x000001CC	BERCFG3	BER Config 3	PCI PHY	Section 8.4.2.1.40
0x000001E8	PCICMD20	PCICMD20 Register	PCI PHY	Section 8.4.2.1.41
0x000001EC	BCTRL0	BCTRL0 Register	PCI PHY	Section 8.4.2.1.42
0x000001F0	BIFCTL	BIFCTL Register	PCI PHY	Section 8.4.2.1.43
0x000001F4	PCIELER0	PCIELER0 Register	PCI PHY	Section 8.4.2.1.44
0x00000200	PCIELER1	PCIELER1 Register	PCI PHY	Section 8.4.2.1.45
0x0000021C	PCICMD0	SERR Enable Register	PCI PHY	Section 8.4.2.1.46
0x00000220	CLSR10	Cache Line Size Register	PCI PHY	Section 8.4.2.1.47

Table 8-5 PCI PHY Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00000224	REUTENGLD3	REUTENGLD3 Register	PCI PHY	Section 8.4.2.1.48
0x00000228	REUTENGLSCAP3	REUTENGLSCAP3 Register	PCI PHY	Section 8.4.2.1.49
0x0000022C	REUTENGLTRCON3	REUTENGLTRCON3 Register	PCI PHY	Section 8.4.2.1.50
0x00000234	REUTPHTDC3	REUTPHTDC3 Register	PCI PHY	Section 8.4.2.1.51
0x00000238	REUTPHTDS3	REUTPHTDS3 Register	PCI PHY	Section 8.4.2.1.52
0x0000023C	REUTPHRDS3	REUTPHRDS3 Register	PCI PHY	Section 8.4.2.1.53
0x00000244	REUTPHPIS3	REUTPHPIS3 Register	PCI PHY	Section 8.4.2.1.54
0x00000248	REUTPHPRE3	REUTPHPRE3 Register	PCI PHY	Section 8.4.2.1.55
0x0000024C	REUTPHPOST3	REUTPHPOST3 Register	PCI PHY	Section 8.4.2.1.56
0x00000254	REUTPATCCTR4	REUTPATCCTR4 Register	PCI PHY	Section 8.4.2.1.57
0x00000274	REUTPMR04	REUTPMR04 Register	PCI PHY	Section 8.4.2.1.58
0x00000278	REUTPMR24	REUTPMR24 Register	PCI PHY	Section 8.4.2.1.59
0x0000027C	REUTPMR33	REUTPMR33 Register	PCI PHY	Section 8.4.2.1.60
0x00000280	FPLACTRL0	FPLACTRL0 Register	PCI PHY	Section 8.4.2.1.61
0x00000284	FPLAAND0	FPLAAND0 Register	PCI PHY	Section 8.4.2.1.62
0x00000288	FPLAOR0	FPLAOR0 Register	PCI PHY	Section 8.4.2.1.63
0x0000028C	FPLAMSEL00	FPLAMSEL00 Register	PCI PHY	Section 8.4.2.1.64
0x00000290	FPLAMSEL10	FPLAMSEL10 Register	PCI PHY	Section 8.4.2.1.65
0x00000294	OBEINJCTL0	XP Outbound Error Injection Control TXN REG	PCI PHY	Section 8.4.2.1.66

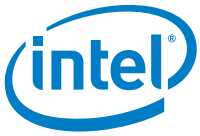


Table 8-5 PCI PHY Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Section
0x00000298	XPPRIVC0	XP Private Control TXN REG	PCI PHY	Section 8.4.2.1.67
0x0000029C	PXP0EINJMSK	PXP Outbound ERR Injection Mask Register TXN REG	PCI PHY	Section 8.4.2.1.68
0x000002B4	PSCGLBCTL	XP Global Control Register	PCI PHY	Section 8.4.2.1.69
0x000002D8	PXP0BNSBITS	Bonus Config Bits	PCI PHY	Section 8.4.2.1.70
0x000002E4	PXP0TXO	PXP Transmit Override / Configuration	PCI PHY	Section 8.4.2.1.71
0x000002F4	PXP0DEMPSEL	PXP De-emphasis Select Register	PCI PHY	Section 8.4.2.1.72
0x00000304	PXP0IBPRTCTL	IBIST Port Control Register	PCI PHY	Section 8.4.2.1.73
0x00000308	XPPGESTAT0	Lane Error Status Register	PCI PHY	Section 8.4.2.1.74
0x00000318	RASCAPREG	RAS Capability Register	PCI PHY	Section 8.4.2.1.75
0x0000031C	COMERRCNT0	Lane Error Status Register Low	PCI PHY	Section 8.4.2.1.76
0x00000320	COMERRCNT1	Lane Error Status Register High	PCI PHY	Section 8.4.2.1.77
0x00000338	CLSPHYCTL	Cluster Physical IO Control Register	PCI PHY	Section 8.4.2.1.78
0x0000033C	LEKBERR	Leaky Bucket Error Register	PCI PHY	Section 8.4.2.1.79
0x00000340	PRTPHYCTL0	Port Physical IO Control Register	PCI PHY	Section 8.4.2.1.80
0x00000344	PXPSQCNT0	PXP Cluster Squelch Count	PCI PHY	Section 8.4.2.1.81
0x00000348	PXP0PHYCTL3	PXP PHY Control Register 3	PCI PHY	Section 8.4.2.1.82
0x0000034C	PXP0PHYCTL4	PXP PHY Control Register 4	PCI PHY	Section 8.4.2.1.83
0x00000350	PXP0PHYCTL5	PXP PHY Control Register 5	PCI PHY	Section 8.4.2.1.84
0x00000354	PXP0PHYCTL6	PXP PHY Control Register 6	PCI PHY	Section 8.4.2.1.85

Table 8-5 PCI PHY Registers Summary (Continued)

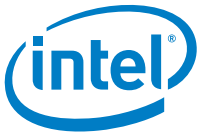
Offset / Alias Offset	Abbreviation	Name	Block	Section
0x0000036C	PXPDLLPCTRL	DLL Control Register	PCI PHY	Section 8.4.2.1.86
0x00000370	PXPRETRYCTRL	Retry Control Register	PCI PHY	Section 8.4.2.1.87
0x00000388	PXPLNKMISC0	Link Layer Misc Control Register	PCI PHY	Section 8.4.2.1.88
0x0000038C	PXP0CDTAVAIL0	Credit Availability Status Register 0	PCI PHY	Section 8.4.2.1.89
0x00000390	PXP0CDTAVAIL1	Credit Availability Status Register 1	PCI PHY	Section 8.4.2.1.90
0x00000394	EXPBERR0	Expected Bit Error Rate Register 0	PCI PHY	Section 8.4.2.1.91
0x00000398	EXPBERR1	Expected Bit Error Rate Register 1	PCI PHY	Section 8.4.2.1.92
0x000003A0	LMPMB	Link Layer PM	PCI PHY	Section 8.4.2.1.93
0x000003D0	RXPMCTRLGEN1	Rx PM Control Gen1 Register	PCI PHY	Section 8.4.2.1.94
0x000003D4	RXPMCTRLGEN2	Rx PM Control Gen2 Register	PCI PHY	Section 8.4.2.1.95
0x000003D8	PIPE1	PIPE DFT Register	PCI PHY	Section 8.4.2.1.96
0x000003DC	PIPE2	PIPE Register 2	PCI PHY	Section 8.4.2.1.97

8.4.2 Detailed Register Description

8.4.2.1 Link/PHY Registers

8.4.2.1.1 Init Disable - INITDISABLE (0x00000004)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	1:0	0b	RW	Reserved.
RESERVED	31:2	0x0	RSV	Reserved.



8.4.2.1.2 Clock Gate - CLKGATE (0x00000038)

Field	Bit(s)	Init.	Access Type	Description
DISABLE	1:0	0x0	RW	
RESERVED	31:2	0x0	RSV	Reserved

8.4.2.1.3 BER Meter Locked Register - BER (0x0000005C)

Field	Bit(s)	Init.	Access Type	Description
LOCKED	7:0	0x0	RO	BER Meter Locked (per lane).
RESERVED	31:8	0x0	RSV	Reserved.

8.4.2.1.4 BER Meter for Lane 0 - SRBER0 (0x00000060)

Field	Bit(s)	Init.	Access Type	Description
BER_LRGECNT	23:0	0x0	RO	BER Meter Errors Counter for Lane 0.
RESERVED	31:24	0x0	RSV	Reserved.

8.4.2.1.5 BER Meter for Lane 1 - SRBER1 (0x00000064)

Field	Bit(s)	Init.	Access Type	Description
BER_LRGECNT	23:0	0x0	RO	BER Meter Errors Counter for Lane 1.
RESERVED	31:24	0x0	RSV	Reserved.

8.4.2.1.6 BER Meter for Lane 2 - SRBER2 (0x00000068)

Field	Bit(s)	Init.	Access Type	Description
BER_LRGECNT	23:0	0x0	RO	BER Meter Errors Counter for Lane 2.
RESERVED	31:24	0x0	RSV	Reserved.

8.4.2.1.7 BER Meter for Lane 3 - SRBER3 (0x0000006C)

Field	Bit(s)	Init.	Access Type	Description
BER_LRGECNT	23:0	0x0	RO	BER Meter Errors Counter for Lane 3.
RESERVED	31:24	0x0	RSV	Reserved.

8.4.2.1.8 BER Meter for Lane 4 - SRBER4 (0x00000070)

Field	Bit(s)	Init.	Access Type	Description
BER_LRGECNT	23:0	0x0	RO	BER Meter Errors Counter for Lane 4.
RESERVED	31:24	0x0	RSV	Reserved.

8.4.2.1.9 BER Meter for Lane 5 - SRBER5 (0x00000074)

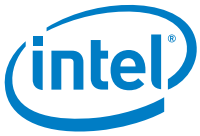
Field	Bit(s)	Init.	Access Type	Description
BER_LRGECNT	23:0	0x0	RO	BER Meter Errors Counter for Lane 5.
RESERVED	31:24	0x0	RSV	Reserved.

8.4.2.1.10 BER Meter for Lane 6 - SRBER6 (0x00000078)

Field	Bit(s)	Init.	Access Type	Description
BER_LRGECNT	23:0	0x0	RO	BER Meter Errors Counter for Lane 6.
RESERVED	31:24	0x0	RSV	Reserved.

8.4.2.1.11 BER Meter for Lane 7 - SRBER7 (0x0000007C)

Field	Bit(s)	Init.	Access Type	Description
BER_LRGECNT	23:0	0x0	RO	BER Meter Errors Counter for Lane 7.



Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:24	0x0	RSV	Reserved.

8.4.2.1.12 NTFS Register - NFTS (0x00000090)

Field	Bit(s)	Init.	Access Type	Description
GEN1FTS	7:0	0x7c	RW	Gen1 NFTS. The number of Fast Training Sequences required by the receiver to obtain reliable bit and symbol lock in Gen1.
GEN1FTSSENTOVD	15:8	0x0	RW	Gen1 FTS Sent Override. Override the number of Fast Training Sequences that should be sent when exiting L0s in Gen1.
GEN2FTS	23:16	0xf4	RW	Gen2 NFTS. The number of Fast Training Sequences required by the receiver to obtain reliable bit and symbol lock in Gen2.
GEN2FTSSENTOVD	31:24	0x0	RW	Gen2 FTS Sent Override. Override the number of Fast Training Sequences that should be sent when exiting L0s in Gen2.

8.4.2.1.13 REUTENGLSSEL0 Register - REUTENGLSSEL0 (0x0000009C)

Field	Bit(s)	Init.	Access Type	Description
LNKSPEEDSEL	31:0	0x2	RW	Bit[7] is CSR enable. Enables link speed overdie and override values from bits [3:0].

8.4.2.1.14 CRC Error Count Port 0 - REUTERRCED0 (0x000000A0)

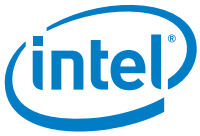
Field	Bit(s)	Init.	Access Type	Description
CRCCOUNT	14:0	0x0	RW1C	Count CRC errors in link layer for port 0.
CRCOVERFLOW	15	0b	RW1C	Overflow of the CRC error count for port 0.
RESERVED	31:16	0x0	RSV	Reserved.

8.4.2.1.15 RCV Error Count - REUTE0 (0x000000A4)

Field	Bit(s)	Init.	Access Type	Description
RCVERRCOUNT	14:0	0x0	RW1C	Receiver Error Count. Counter of the receiver 10b/8b errors.
RCVERROVERFLOW	15	0b	RW1C	Receiver Error Count Overflow.
RECCOUNT	30:16	0x0	RW1C	Recovery Count. Count the number of times the LTSSM entered Recovery.RcvrLock.
RECOVERFLOW	31	0b	RW1C	Recovery Counter Overflow.

8.4.2.1.16 REUTENGLTRON0 Register - REUTENGLTRON0 (0x000000A8)

Field	Bit(s)	Init.	Access Type	Description
PHYRESET	0	0b	RW	Force the LTSSM to be in detect state and reset the PHY.
RESERVED	4:1	0x0	RSV	Reserved
PHYINIT	5	1b	RW	PHY Init 1b = Initialize the Phy link when get to Detect.Quiet state. 0b = Don't Initialize the Phy link and stay in Detect.Quiet state.
RESERVED	7:6	0x0	RSV	Reserved.
INITMODE	10:8	0x0	RW	PHY Mode. 1 = Loopback. 2 = Short reset mode. 3 = Send compliance bit and loopback. 4 = Master loopback 5 = Slave loopback. 6 = Send compliance bit.
RESERVED	11	0b	RSV	Reserved.
INITABORTFRZ	12	0b	RW	Freeze LTSSM State.
DISAUTOCOMPL	13	0b	RW	No Compliance Mode. Effective only in LTSSM_X4_POL_ACTIVE.
RESERVED	21:14	0x0	RSV	Reserved.
ENABLESCRAM	22	1b	RW	Scramble Bits Enable.



Field	Bit(s)	Init.	Access Type	Description
RCVR_PROLONG	23	0b	RW	Enable 1024 TS send on recovery, after Elastic buffer full error.
RESERVED	30:24	0x0	RSV	Reserved.
BYPASSDET	31	0b	RW	Bypass the detect sequence in LTSSM.

8.4.2.1.17 Speed Timeout Register - GENTRANS (0x000000AC)

Field	Bit(s)	Init.	Access Type	Description
SPD_TIMEOUT	10:0	0x6A4	RW	Time out value on waiting for PIPE to finish rate transition. LSI defined 7 µs worst case (1700).
RESERVED	31:11	0x0	RSV	Reserved.

8.4.2.1.18 Config.Lanenum.WaitAccept Register - CFGWAITACCEPT (0x000000E8)

Field	Bit(s)	Init.	Access Type	Description
ACCEPT_TO_MS	1:0	0x0	RW	Timeout in ms to the LTSSM for staying in Config.Lanenum.Accept state.
ACCEPT_TO_CC	4:2	0x0	RW	Number of cycles to the LTSSM for staying in Config.Lanenum.Accept state (effective only if csr_cfg_accept_to_ms_cb = 0x0).
RESERVED	7:5	0x0	RSV	Reserved.
DIFF_ON_LANE_NUM_ONLY	8	0b	RW	Check for different lane numbers in Config.Lanenum.Wait. Only on non-PAD lanes.
WAIT2ACCEPT_ON_TS1	9	0b	RW	Move from Config.Lanenum.Wait to Config.Lanenum.Accept. Also when detect match link numbers and lane numbers on two consecutive TS1 Ordered Sets.
RESERVED	15:10	0x0	RSV	Reserved.
CMP_LINK_NUM_LANE_NUM_MATCH_DIS	16	0b	RW	Move from Config.Lanenum.Accept to Config.Complete. Only when the received link numbers and lane numbers in the received TS2 Ordered Sets match the transmitted link numbers and lane numbers.
RST_TS_COUNT	17	0b	RW	Reset the TS2 Ordered Sets counter in the transition from Config.Lanenum.Wait to Config.Lanenum.Accept.
LOAD_IN_CFGLNK_ONLY	18	0b	RW	0b = Load the new lane numbers both in Config.Linknum and Config.Lanenum.Wait. 1b = Load the new lane numbers only in Config.Linknum.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:19	0x0	RSV	Reserved.

8.4.2.1.19 Trapper Event Reg DW0 - TRAPPER0 (0x00000114)

Field	Bit(s)	Init.	Access Type	Description
EVENT0_DW0	31:0	0x0	RW	DLLP/TLP Trapper Event Reg Dword 0. 4 Dwords that defines the DLLP event to be trapped at the link layer input (inbound).

8.4.2.1.20 Trapper Event Reg DW1 - TRAPPER1 (0x00000118)

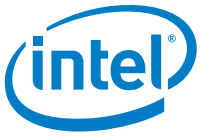
Field	Bit(s)	Init.	Access Type	Description
EVENT1_DW0	31:0	0x0	RW	DLLP/TLP Trapper Event Reg Dword 1. 4 Dwords that defines the DLLP event to be trapped at the link layer input (inbound).

8.4.2.1.21 Trapper Event Reg DW2 - TRAPPER2 (0x0000011C)

Field	Bit(s)	Init.	Access Type	Description
EVENT0_DW2	31:0	0x0	RW	DLLP/TLP Trapper Event Reg Dword 2. 4 Dwords that defines the DLLP event to be trapped at the link layer input (inbound).

8.4.2.1.22 Trapper Event Reg DW3 - TRAPPER3 (0x00000120)

Field	Bit(s)	Init.	Access Type	Description
EVENT0_DW3	31:0	0x0	RW	DLLP/TLP Trapper Event Reg Dword 3. 4 Dwords that defines the DLLP event to be trapped at the link layer input (inbound).



8.4.2.1.23 Trapper Event Mask DW0 - TRAPPERM0 (0x00000124)

Field	Bit(s)	Init.	Access Type	Description
MASKT0_DW0	31:0	0x0	RW	DLLP Trapper Event Mask Dword 0. 4 Dwords that defines the DLLP trapper event mask. The comparison is done only on bits set to 1b.

8.4.2.1.24 Trapper Event Mask DW1 - TRAPPERM1 (0x00000128)

Field	Bit(s)	Init.	Access Type	Description
MASKT0_DW1	31:0	0x0	RW	DLLP Trapper Event Mask Dword 1. 4 Dwords that defines the DLLP trapper event mask. The comparison is done only on bits set to 1b.

8.4.2.1.25 Trapper Event Mask DW2 - TRAPPERM2 (0x0000012C)

Field	Bit(s)	Init.	Access Type	Description
MASKT0_DW2	31:0	0x0	RW	DLLP Trapper Event Mask Dword 2. 4 Dwords that defines the DLLP trapper event mask. The comparison is done only on bits set to 1b.

8.4.2.1.26 Trapper Event Mask DW3 - TRAPPERM3 (0x00000130)

Field	Bit(s)	Init.	Access Type	Description
MASKT0_DW3	31:0	0x0	RW	DLLP Trapper Event Mask Dword 3. 4 Dwords that defines the DLLP trapper event mask. The comparison is done only on bits set to 1b.

8.4.2.1.27 Trapped Event Counter - TRAPPERC1 (0x00000134)

Field	Bit(s)	Init.	Access Type	Description
ENENTCNT	15:0	0x0	RW	DLLP/TLP trapped event counter. Counts the trapped events and also enables decrement using aging (leaky bucket).

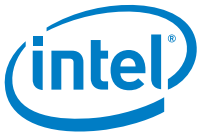
Field	Bit(s)	Init.	Access Type	Description
LB_THR	31:16	0x0	RW	DLLP/TLP Trapper Leaky Bucket Count Threshold. Leaky bucket counter resolution is ~1 µs in Gen1 and ~0.5 µs in Gen2 (128 cycles). When timer count reaches the threshold, the trapper event counter is decremented. A zero value in this register disables leaky bucket.

8.4.2.1.28 Trapper Counter Clear - TRAPPERC2 (0x00000138)

Field	Bit(s)	Init.	Access Type	Description
CNTCLR	0	0b	RW	Trapper Counter Clear. Writing 1b to this bit clears the event trapper counter.
RESERVED	31:1	0x0	RSV	Reserved.

8.4.2.1.29 PIPE Register - PIPE (0x00000140)

Field	Bit(s)	Init.	Access Type	Description
LANE_STG_MAX_PS_CNT	12:0	0x0	RW	P0-P1 Lane Power Down Staggering. Defines the delay between lanes turn on and off (P0/P1 power down state transition).
CHKN_PHYSTATUS	13	0b	RW	PHY Status AND Mode Between Lanes. when set to 1b, the PHY status from the lanes are ORed toward LTSSM.
EXTNDRXVALID	17:14	0x5	RW	Extends a pipe Rx valid signal de-assertion by a number of cycles to assure Rx elastic buffer and the de-skew buffer are flushed at the beginning of electrical idle.
EIWHENP0BUSY	18	0b	RW	Enable holding Tx electrical Idle on the transition L1-->L0 until all PHY status indications are received from all lanes.
RESERVED	31:19	0x0	RSV	Reserved.



8.4.2.1.30 CHKN Register - CHKN (0x00000144) PCI PHY

Field	Bit(s)	Init.	Access Type	Description
REVERSAL	0	0b	RW	Lane Reversal Mode. Previous mode - The X540 starts non-reverse mode and only if a partner can't reverse. If not, the X540 does the reverse. Current mode (default) - The X540 sends the right order (reversed or not reversed in the first TS that includes the lane number. Note that the partner might not know that a reverse took place.
CHNGSPEED	1	0b	RW	Clear changed_speed viable on L0->REC transition. Default is 0b (disabled).
EIOSDETFROMDB	2	0b	RW	Enable EIOS detection from the de-skew buffer output and not from Elastic Buffer (EB) output. The default is from EB.
RESERVED	31:3	0x0	RSV	Reserved.

8.4.2.1.31 REUTMCTR20 Register - REUTMCTR20 (0x00000160)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	7:0	0x0	RSV	Reserved.
SEL_DEEMPPHASIS	8	0b	RW	Value for sel_deemp.
RESERVED	11:9	0x0	RSV	Reserved.
RESERVED	31:12	0x0	RSV	Reserved.

8.4.2.1.32 BER Meter Symbol Pattern 0 - BERTESTSYM0 (0x00000174)

Field	Bit(s)	Init.	Access Type	Description
TEST_SYM_REG_R	29:0	0x2aaa0d55	RW	BER Meter Symbol Pattern. Bits 29:0.
RESERVED	31:30	0x0	RSV	Reserved.

8.4.2.1.33 BER Meter Symbol Pattern 1 - BERTESTSYM1 (0x00000178)

Field	Bit(s)	Init.	Access Type	Description
TEST_SYM_REG_R	9:0	0x17C	RW	BER Meter Symbol Pattern. Bits 39:30.
RESERVED	31:10	0x0	RSV	Reserved.

8.4.2.1.34 BER Config 1 - BERCFG1 (0x00000180)

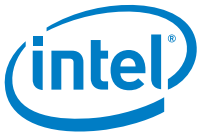
Field	Bit(s)	Init.	Access Type	Description
START_TX	7:0	0x0	RW	Start BER Transmit and Receive. Per lane [7:0].
EI_FLT	8	0b	RW	Force BER idle.
DET_IN_TEST	9	0b	RW	Indication for Production Mode BER. Count until 2^12.
RESERVED	15:10	0x0	RSV	Reserved.
START_COMMA_SEND	23:16	0x0	RW	Force comma sending on the Tx data.
RESERVED	31:24	0x0	RSV	Reserved.

8.4.2.1.35 BER Config 2 - BERCFG2 (0x00000184)

Field	Bit(s)	Init.	Access Type	Description
DET_THRESH	14:0	0x4	RW	The Threshold of BER Loops Count. Before BER enters normal mode. (length of BER lock).
WX_THRESHOLD	24:15	0x0	RW	Indication for Production Mode BER. Count until this threshold.
RESERVED	31:25	0x0	RSV	Reserved

8.4.2.1.36 PLL Lock Register - PLLLOCK (0x00000190)

Field	Bit(s)	Init.	Access Type	Description
FORCE_PLL_LOCK_IND	0	0b	RW	Force PLL Locked Indication. Bypasses the PLL calibration block.



Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:1	0x0	RSV	Reserved.

8.4.2.1.37 SWZL Register - EN (0x00000194)

Field	Bit(s)	Init.	Access Type	Description
SWZL_0	7:0	0x0	RW	Not connected.
SWZL_1	15:8	0x0	RW	Not connected.
RESERVED	31:16	0x0	RSV	Reserved.

8.4.2.1.38 Soft Reset Register - PCIE (0x0000019C)

Field	Bit(s)	Init.	Access Type	Description
CSR_LANE_RST	7:0	0x0	RW	Generates soft reset to lanes from CSR.
CSR_PLL_RST	8	0b	RW	Generates soft reset to PLL from CSR.
CSR_PLL_PD	10:9	0x0	RW	Generates soft reset to PLL from CSR.
RESERVED	31:11	0x0	RSV	Reserved.

8.4.2.1.39 Far-end Loopback Register - FELB (0x000001A8)

Field	Bit(s)	Init.	Access Type	Description
LOOPB	7:0	0x0	RW	Operate Far-end Loopback. Connects Rx to Tx. Tx transmits whatever received in Rx. Per lane control.
RESERVED	31:8	0x0	RSV	Reserved.

8.4.2.1.40 BER Config 3 - BERCFG3 (0x000001CC)

Field	Bit(s)	Init.	Access Type	Description
LIN_TIME	7:0	0xFF	RW	

Field	Bit(s)	Init.	Access Type	Description
EXP_TIME	12:8	0x0	RW	
BER_14_SYMBOLS	13	0b	RW	
RESERVED	15:14	0x0	RSV	Reserved.
TIMER_COMPLETE	23:16	0x0	RW	
SUCCESS_COMPLETE	31:24	0x0	RW	

8.4.2.1.41 PCICMD20 Register - PCICMD20 (0x000001E8)

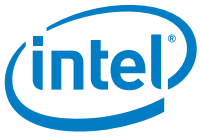
Field	Bit(s)	Init.	Access Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
VGA_PALETTE_SNOOP_ENABLEHACK	5	0b	RW	
RESERVED	31:6	0x0	RSV	Reserved.

8.4.2.1.42 BCTRL0 Register - BCTRL0 (0x000001EC)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	21:0	0x0	RSV	Reserved.
SRESET	22	0b	RW	
RESERVED	31:23	0x0	RSV	Reserved.

8.4.2.1.43 BIFCTL Register - BIFCTL (0x000001F0)

Field	Bit(s)	Init.	Access Type	Description
BIFCTL	2:0	0x7	RW	
STARTBIF	3	1b	RW	
RESERVED	31:4	0x0	RSV	Reserved.



8.4.2.1.44 PCIELER0 Register - PCIELER0 (0x000001F4)

Field	Bit(s)	Init.	Access Type	Description
L_20_LERENABLE	0	0b	RW	
RESERVED	31:1	0x0	RSV	Reserved

8.4.2.1.45 PCIELER1 Register - PCIELER1 (0x00000200)

Field	Bit(s)	Init.	Access Type	Description
L_21_LERENABLE	0	0b	RW	
RESERVED	31:1	0x0	RSV	Reserved

8.4.2.1.46 SERR Enable Register - PCICMD0 (0x0000021C)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	7:0	0x0	RSV	Reserved.
SERR_ENABLE	8	0b	RW	
RESERVED	31:9	0x0	RSV	Reserved.

8.4.2.1.47 Cache Line Size Register - CLSR10 (0x00000220)

Field	Bit(s)	Init.	Access Type	Description
CACHELINE_SIZE	7:0	0x0	RW	
RESERVED	31:8	0x0	RSV	Reserved

8.4.2.1.48 REUTENGLD3 Register - REUTENGLD3 (0x00000224)

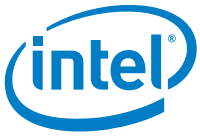
Field	Bit(s)	Init.	Access Type	Description
RXLW2_IN	4:0	0x0	RO	
RXLW_IN	9:5	0x0	RO	
RESERVED	12:10	0x0	RSV	Reserved.
RESERVED	20:13	0x17	RSV	Reserved.
RESERVED	22:21	0x0	RSV	Reserved.
LNKINACT_IN	23	0b	RO	
RESERVED	29:24	0x2	RSV	Reserved.
RESERVED	31:30	0x0	RSV	Reserved.

8.4.2.1.49 REUTENGLSCAP3 Register - REUTENGLSCAP3 (0x00000228)

Field	Bit(s)	Init.	Access Type	Description
LNKSPEEDCAP_IN	31:0	0x0	RO	

8.4.2.1.50 REUTENGLTRCON3 Register - REUTENGLTRCON3 (0x0000022C)

Field	Bit(s)	Init.	Access Type	Description
LINKCTRL	15:0	0x0	RW	
RESERVED	22:16	0x0	RSV	Reserved.
LINKAUTONEG	23	0b	RW	
RESERVED	24	0b	RSV	Reserved.
LINKSELECT	28:25	0x0	RW	
RESERVED	31:29	0x0	RSV	Reserved.



8.4.2.1.51 REUTPHTDC3 Register - REUTPHTDC3 (0x00000234)

Field	Bit(s)	Init.	Access Type	Description
TXDATALANEDIS	7:0	0x0	RW	
RESERVED	15:8	0x0	RSV	Reserved.
RESERVED	31:16	0x0	RSV	Reserved.

8.4.2.1.52 REUTPHTDS3 Register - REUTPHTDS3 (0x00000238)

Field	Bit(s)	Init.	Access Type	Description
TXLANEDETSTAT_IN	7:0	0x0	RO	
RESERVED	15:8	0x0	RSV	Reserved.
RESERVED	31:16	0x0	RSV	Reserved.

8.4.2.1.53 REUTPHRDS3 Register - REUTPHRDS3 (0x0000023C)

Field	Bit(s)	Init.	Access Type	Description
RXDATAREADY_IN	7:0	0x0	RO	
RESERVED	15:8	0x0	RSV	Reserved.
RESERVED	31:16	0x0	RSV	Reserved.

8.4.2.1.54 REUTPHPIS3 Register - REUTPHPIS3 (0x00000244)

Field	Bit(s)	Init.	Access Type	Description
LINKUP	0	0b	RW	prevent_init_speed_change. Present initiate speed change. 0b = Initiate recovery and speed change if both sides support gen2 after linkup. 1b = Don't initiate recovery, but execute the speed change if the other device initiates recovery.
RESERVED	7:1	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Access Type	Description
LTSSMSTATE_IN	12:8	0x0	RW	
RESERVED	15:13	0x0	RSV	Reserved.
RXTRACKER_IN	19:16	0x0	RW	
RESERVED	31:20	0x0	RSV	Reserved.

8.4.2.1.55 REUTPHPRE3 register - REUTPHPRE3 (0x00000248)

Field	Bit(s)	Init.	Access Type	Description
PRECFGID_IN	31:0	0x0	RO	

8.4.2.1.56 REUTPHPOST3 Register - REUTPHPOST3 (0x0000024C)

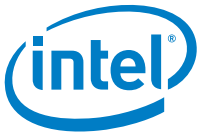
Field	Bit(s)	Init.	Access Type	Description
POSTFGID_IN	31:0	0x0	RO	

8.4.2.1.57 REUTPATCCTR4 Register - REUTPATCCTR4 (0x00000254)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	15:0	0x0	RSV	Reserved.
RXERRCNTSEL	19:16	0x0	RW	
RESERVED	31:20	0x0	RSV	Reserved.

8.4.2.1.58 REUTPMR04 Register - REUTPMR04 (0x00000274)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	23:0	0x0	RSV	Reserved.



Field	Bit(s)	Init.	Access Type	Description
GEN1FTSSENT_IN	31:24	0x0	RO	

8.4.2.1.59 REUTPMR24 Register - REUTPMR24 (0x00000278)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	23:0	0x0	RSV	Reserved.
GEN2FTSSENT_IN	31:24	0x0	RO	

8.4.2.1.60 REUTPMR33 Register - REUTPMR33 (0x0000027C)

Field	Bit(s)	Init.	Access Type	Description
L0SRXCNT_IN	3:0	0x0	RO	
RESERVED	4	0b	RSV	Reserved.
L0XTXCNT_IN	8:5	0x0	RO	
RESERVED	31:9	0x0	RSV	Reserved.

8.4.2.1.61 FPLACTRL0 Register - FPLACTRL0 (0x00000280)

Field	Bit(s)	Init.	Access Type	Description
FPLAIOEN	0	0b	RW	
ANDARYEN	1	0b	RW	
ORARYEN	2	0b	RW	
RDWRB	3	1b	RW	
RESERVED	7:4	0x0	RSV	Reserved.
PTM4ADDR	9:8	0x0	RW	
PTADDR	12:10	0x0	RW	
RESERVED	14:13	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Access Type	Description
ANDINVSEL	15	0b	RW	
BWEOR	19:16	0x0	RW	
RESERVED	23:20	0x0	RSV	Reserved.
BWEAND	27:24	0x0	RW	
RESERVED	31:28	0x0	RSV	Reserved.

8.4.2.1.62 FPLAAND0 Register - FPLAAND0 (0x00000284)

Field	Bit(s)	Init.	Access Type	Description
CSR_ANDIN	12:0	0x0	RW	
RESERVED	31:13	0x0	RSV	Reserved

8.4.2.1.63 FPLAOR0 Register - FPLAOR0 (0x00000288)

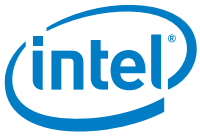
Field	Bit(s)	Init.	Access Type	Description
CSR_ANDIN	12:0	0x0	RW	
RESERVED	31:13	0x0	RSV	Reserved

8.4.2.1.64 FPLAMSEL00 Register - FPLAMSEL00 (0x0000028C)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	0	0b	RSV	Reserved.
CSR_OUTMUXSEL	31:1	0x0	RW	

8.4.2.1.65 FPLAMSEL10 Register - FPLAMSEL10 (0x00000290)

Field	Bit(s)	Init.	Access Type	Description
CSR_OUTMUXSEL	11:0	0x0	RW	



Field	Bit(s)	Init.	Access Type	Description
RESERVED	31:12	0x0	RSV	Reserved.

8.4.2.1.66 XP Outbound Error Injection Control TXN REG - OBEINJCTL0 (0x00000294)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	2:0	0x0	RSV	Reserved.
ETRANS_EN	3	0b	RW	
EBITPOS	13:4	0x0	RW	EBITPOS. Error injection bit position [9:0]. Bit field can contain more than one bit for multi-bit error injection.
ETRANS	19:14	0x0	RW	ETRANS. Symbol transfer number [5:0]. This field indicates which symbol contains the corrupted bit. A maximum depth of 64 is where the corruption could occur.
CRCERRINJ	20	0b	RW	CRCERRINJ. CRC error injection. Selects which type of injection is performed when DINJ0,1 is asserted. 0b = Data bit corruption. Based ETRANS and EBITPOS values. 1b = CRC bit corruption. Based on values in PEX[6:0]EINJMSK.
RESERVED	30:21	0x0	RSV	Reserved.
DISRETRYERR	31	0b	RW	

8.4.2.1.67 XP Private Control TXN REG - XPPRIVC0 (0x00000298)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	3:0	0x0	RSV	Reserved.
UNUSED0	7:4	0x0	RW	
POWRSVPHY	8	0b	RW	
UNUSED1	15:9	0x0	RW	
RESERVED	24:16	0x0	RSV	Reserved.
ENABLEPCIE125	25	0b	RW	
RESERVED	31:26	0x0	RSV	Reserved.

8.4.2.1.68 PXP Outbound err Injection Mask Register TXN REG - PXP0EINJMSK (0x0000029C)

Field	Bit(s)	Init.	Access Type	Description
CRCMSK	31:0	0x0	RW	CRCMSK. CRC Mask A logic 1 in any location inverts the CRC for an outbound transaction.

8.4.2.1.69 XP Global Control Register - PSCGLBCTL (0x000002B4)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	21:0	0x0	RSV	Reserved.
GEN2SEL_DELAY	24:22	0x4	RW	Determine the delay on the gen2sel from LTSSM to CAR & PLL.
RX_GATE_DURATION	28:25	0x8	RW	Determine the duration of rxclk gating starting gen2sel from LTSSM.
PLL_LNK_RATE_SELECT	29	1b	RW	Allow reducing the pcie_lclk to 125 MHz. Should be fixed at 250 MHz (value 1).
RESERVED	31:30	0x0	RSV	Reserved.

8.4.2.1.70 Bonus Config Bits - PXP0BNSBITS (0x000002D8)

Field	Bit(s)	Init.	Access Type	Description
BBITS	31:0	0x0	RW	BBITS. Bonus Bits. These are extra read/write configuration bits for potential stepping changes.

8.4.2.1.71 PXP Transmit Override / Configuration - PXP0TXO (0x000002E4)

Field	Bit(s)	Init.	Access Type	Description
TXPCMPSRCSEL	0	0b	RW	
TXNCMPSRCSEL	1	0b	RW	

Field	Bit(s)	Init.	Access Type	Description
RESERVED	3:2	0x0	RSV	Reserved.
SHNTRS_VAL	4	0b	RW	SHNTRSEN. Value of shunt resistor during squelch entry sequence. If TOE is set, this bit controls the Tx *Shunt* bit directly; otherwise, this bit has no function. 0b = Disable shunt. 1b = Enable shunt.
RESERVED	31:5	0x0	RSV	Reserved.

8.4.2.1.72 PXP De-emphasis Select Register - PXP0DEMPSEL (0x000002F4)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	29:0	0x0	RSV	Reserved.
DMPLENFOVRD	30	0b	RW	Enforce Deemphasis Override. Override the negotiated deemphasis level with the value of csr_sel_deempphasis.
RESERVED	31	0b	RSV	Reserved.

8.4.2.1.73 IBIST Port Control Register - PXP0IBPRTCTL (0x00000304)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
HVMLPBK	5	0b	RW	HVMLPBK. HVM loopback mode. This bit enables forward progress to be made with master/slave agent loopback protocol to promote self loopback in HVM component testing. 0b = Normal loopback protocol for two agents. 1b = Enable HVM loopback mode. The training state machine ignores the loopback bit that the receiver sees and does not physically connect the loopback path.
RESERVED	31:6	0x0	RSV	Reserved.

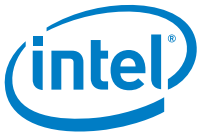
8.4.2.1.74 Lane Error Status Register - XPPGESTAT0 (0x00000308)

Field	Bit(s)	Init.	Access Type	Description
RESERVED	19:0	0x0	RSV	Reserved
CLRCOMPERR	20	0b	RW	Clear Compliance Error Count. When set, this bit clears the error count during compliance.
RESERVED	31:21	0x0	RSV	Reserved

8.4.2.1.75 RAS Capability Register - RASCAPREG (0x00000318)

Field	Bit(s)	Init.	Access Type	Description
RASCAPABILITY_IN	29:0	0x0	RW	[0]: Predictive_Fault_Capable: Set to 1 for components that support. Else 0 (reserved) [1]: X16_Reverse: Indicates whether component is capable of supporting x16 lane reversed [3:2]: X8_Capable: Bit 2 (3) indicates lanes 0-7 (8-15) can form a x8 link [5:4]: X8_Reverse_Capable: Bit 4 (5) indicates if lanes 7-0 (15-8) can form a x8 link. [9:6]: X4_Capable: Bits 6, 7, 8, and 9 indicate if a x4 link can be formed in lanes 0-3, 4-7, 8-11, and 12-15 [13:10]: X4_Reverse_Capable: Bits 10, 11, 12, and 13 indicate if a x4 link can be formed in lanes 3-0, 7-4, 11-8, 15-12 respectively. [21:14]: X2_Capable: Bits 14, 15,..., 21 indicate if a x2 link can be formed in lanes 0-1, 2-3, ..., 14-15 respectively. [29:22]: X2_Reverse_Capable: Bits 22, 23, ..., 29 indicate if a x2 link (reversed from above) can be formed in lanes 1-0, 3-2, ..., 15-14 respectively
ENPMTIMEOUT	30	0b	RW	
RESERVED	31	0b	RSV	Reserved.

8.4.2.1.76 Lane Error Status Register Low - COMERRCNT0 (0x0000031C) PCI PHY



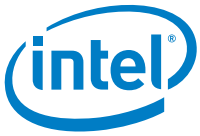
Field	Bit(s)	Init.	Access Type	Description
ERRCNT	31:0	0x0	RWS	Compliance Error Count Value.

8.4.2.1.77 Lane Error Status Register High - COMERRCNT1 (0x00000320)

Field	Bit(s)	Init.	Access Type	Description
ERRCNT	31:0	0x0	RWS	Compliance Error Count Value.

8.4.2.1.78 Cluster Physical IO Control Register - CLSPHYCTL (0x00000338)

Field	Bit(s)	Init.	Access Type	Description
SOS_INTER_VAL	10:0	0x251	RW	sos_inter_val. SOS interval. The interval for send the Skipped-Order Set is programmable. Default value = 593 * 4 ns (250 MHz core domain) = 2.37 µs.
INDEP_PULL_REC_L0S	11	1b	RW	indep_pull_rec_IOs. Independent data extraction in clock recovery logic. 0b = Do not pull data from clock recovery. 1b = Pull data from clock recovery.
RESERVED	12	0b	RSV	Reserved.
DBL_TS_SEND	13	0b	RW	dbl_ts_send. Double the number of TS set. 0b = Send the standard quantity of TS sets. 1b = Send 2x the number of TS sets. Except for polling.active.
ERR_THRESHOLD	18:14	0x0	RW	err_threshold[4:0]. Error threshold. Sets the threshold for leaky bucket error handling.
SPLIT_IN_POLLING	19	0b	RW	split_in_polling. Split the LTSSM for detect mode only in polling. 0b = Do not split. 1b = Split the LTSSM.
IDLEFRXD	20	0b	RW	idlerxd. Use exitei signal as override of receive detect. 0b = Do not use exitei. 1b = Use exitei signal to override rcv_detect.
ENDYNLNK	21	0b	RW	ddynlnken. Enable dynamic auto-negotiating. Enables the dynamic auto-negotiating with re-splitting or re-combining of the LTSSM. 0b = Disable dynamic auto-negotiation. 1b = Enable dynamic auto-negotiation.
SPLDET	22	0b	RW	spldetdis. Disable LTSSM splitting in detect state. 0b = Enable LTSSM splitting. 1b = Disable LTSSM splitting.
DISCOMP	23	0b	RW	discomp. Disable loopback skip compensation. 0b = Enable loopback skip compensation. 1b = Disable loopback skip compensation.



Field	Bit(s)	Init.	Access Type	Description
DISREC2CFGALGN	24	0b	RW	rec2cfgalgndis: Disable K-align when Entering config state 0b = Enable K-align. 1b = Disable K-align when entering config state (only in dynamic mode).
ENRECALGN	25	1b	RW	recalgnen. Enable K-align reset when entering recovery state. 0b = Disable K-align. 1b = Enable K-align when entering recovery state (only in dynamic mode).
LOADGENSUP	26	0b	RW	
LNKNUMTIGHTCHECK	27	0b	RW	Tight Link Num Check. Check the link number symbol in the receiving TS1 and TS2 ordered sets.
RESERVED	28	0b	RSV	Reserved.
CLDATAINSQ	29	1b	RW	Clear Data To Elastic Buffer In Squelch. When set, this bit clears the data to the EB when the receiver detects squelch.
DRCCHANGEL0S	30	0b	RW	
RESERVED	31	0b	RSV	Reserved.

8.4.2.1.79 Leaky Bucket Error Register - LEKBERR (0x0000033C)

Field	Bit(s)	Init.	Access Type	Description
AGGR_ERR	19:0	0xffff	RW	aggr_err[19:0]. Aggregate error for leaky bucket error handling.
LTSSM_STOPLOG	27:20	0xff	RW	ltssm_stoplog. Trigger for LTSSM state logging.
RESERVED	31:28	0x0	RSV	Reserved.

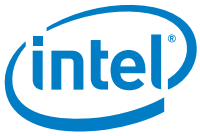
8.4.2.1.80 Port Physical IO Control Register - PRTPHYCTL0 (0x00000340)

Field	Bit(s)	Init.	Access Type	Description
UPSTREAM	0	0b	RW	upstrmen. Upstream mode enable. Select if port operates as downstream or upstream lanes. 0b = Upstream mode disable. 1b = Upstream mode enable.

Field	Bit(s)	Init.	Access Type	Description
RESERVED	15:1	0x0	RSV	Reserved.
FORCELNKINGEN2	16	0b	RW	forcelnkingen2. Force link in Gen2 mode. 0b = Do not force gen2. 1b = Force gen2 mode.
CONTDESKEW	17	0b	RW	contdekewen. Select to use continuous de-skew in L0. 0b = Disable dynamic de-skew. 1b = Enable dynamic de-skew.
CFGDESKEWSKP	18	0b	RW	cfgdekewskp. Select de-skew on SOS or TS in CFG.IDLE. 0b = Do not select de-skew. 1b = Select de-skew on SOS or TS in cfg.idle.
RCVRYDESKEWSKP	19	0b	RW	rcvrydekewskp. Select de-skew on SOS or TS in recovery. 0b = Do not select de-skew. 1b = Select de-skew on SOS or TS in recovery.
RESERVED	22:20	0x0	RSV	Reserved.
CSR_DISABLE_LINKDOWN	23	0b	RW	disable_linkdown. Disable link down to be asserted on transition to detect. 0b = Enable link down. 1b = Disable link down.
RESERVED	31:24	0x0	RSV	Reserved.

8.4.2.1.81 PXP Cluster Squelch Count - PXPSQCNT0 (0x00000344)

Field	Bit(s)	Init.	Access Type	Description
SQCNT	12:0	0x1	RW	SQCNT: Squelch Count. PCIe does not send TS1 sequences after reset until this counter expires. The default value provides 26 µs (6500 * 4 ns).
RESERVED	31:13	0x0	RSV	Reserved.

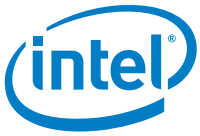


8.4.2.1.82 PXP PHY Control Register 3 - PXP0PHYCTL3 (0x00000348)

Field	Bit(s)	Init.	Access Type	Description
LAST10_MAIN_SM	31:0	0x0	RO	last10_main_sm[31:0]. This bit field logs the last 8 LTSSM (primary) states. 0000b = DETECT 0001b = POLLING/EXT 0010b = CONFIG 0011b = UP 0100b = RECOVERY 0101b = LOOPBACK 0110b = HOTRESET 0110b = PREP_DET 0111b = DISABLED

8.4.2.1.83 PXP PHY Control Register 4 - PXP0PHYCTL4 (0x0000034C)

Field	Bit(s)	Init.	Access Type	Description
LAST10_SUB_SM	31:0	0x0	ROS	last10_sub_sm[31:0]. This bit field logs the last 8 LTSSM sub-states. Detect: 0000b = X4_DET_QUIET_RST 0001b = X4_DET_QUIET_ENTER 0010b = X4_DET_QUIET 0011b = X4_DET_ACT_128US 0100b = X4_DET_ACTIVE 0101b = X4_DET_ACT2_128US 0110b = X4_DET_POL 0111b = X4_DET_EXIT_SQUELCH 1000b = X4_RST_ASSERT 1001b = X4_DET_RECOMBINE 1010b = X4_DET_TRANS_GEN1 Polling: 0000b = X4_POL_ACTIVE 0001b = X4_POL_CONFIG 0010b = X4_POL_COMP_G1 0100b = X4_POL_COMP_G2_ENTRY 0101b = X4_POL_COMP_G2_PREP 0110b = X4_POL_COMP_G2 0011b = Z_NO_EIOS 0010b = X4_POL_COMPLIANCE 0111b = X4_POL_PDP 1000b = X4_POL_COMP_G2_EXIT 1001b = X4_POL_COMP_G2_WAIT 1010b = X4_POL_COMP_G2_PREP_EXIT 1011b = X4_POL_COMP_G1_EIOS 1101b = X4_POL_COMP_G2_IDLE_WAIT 1100b = X4_POL_CHECK_COMPL Config: 0000b = X4_CFG_LNKWID_START 0001b = X4_CFG_LNKWID_ACCEPT 0010b = X4_CFG_LANENUM_WAIT 0011b = X4_CFG_LANENUM_ACCEPT 0100b = X4_CFG_COMPLETE 0101b = X4_CFG_IDLE 0110b = X4_CFG_LNKWID_START_REC 0110b = X4_CFG_RCVRCFG_OLD 0111b = X4_CFG_IDLE_OLD 1000b = X4_CFG_LWS_WAIT_FOR_TS 1001b = X4_CFG_LNKWID_START_UPCFG 1010b = X4_CFG_LNKWID_START_UPCFG_PREP_EXIT_MASTER 1011b = X4_CFG_LNKWID_START_UPCFG_PREP_EXIT_SLAVE



8.4.2.1.84 PXP PHY Control Register 5 - PXP0PHYCTL5 (0x00000350)

Field	Bit(s)	Init.	Access Type	Description
ERR_COUNT_IN	19:0	0x0	RO	LTSSM Errors Counter.
RESERVED	23:20	0x0	RSV	Reserved.
PROLOGNED_ERR_IN	31:24	0x0	RW	LTSSM Errors Counter.

8.4.2.1.85 PXP PHY Control Register 6 - PXP0PHYCTL6 (0x00000354)

Field	Bit(s)	Init.	Access Type	Description
START_LOG	0	1b	RW	start_log. Start logging the LTSSM transitions.
RESERVED	11:1	0x0	RSV	Reserved.
ERR_COUNT_IN	31:12	0x0	RW	LTSSM Errors Counter.

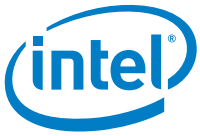
8.4.2.1.86 DLL Control Register - PXPDLLPCTRL (0x0000036C)

Field	Bit(s)	Init.	Access Type	Description
IDLE_FC_PERIOD	4:0	0x1C	RW	idle_fc_period[4:0]. This is the time in which flow control updates are scheduled, if there are no new credits to be received. If L0s is not disabled, this specifies the number of us after which the flow control packet is sent. For ports where L0s is disabled, the timer value is calculated by multiplying this number by 256 ns. The timer has an accuracy of ±16 cycles. **Note:** A value of 0x0 is illegal.
FC_C_PERIOD	9:5	0x3	RW	fc_c_period[4:0]. This specifies the latency after which the flow control packet is scheduled. The latency starts from the time at which the credit release is received by the link layer. For x4/x8/x16, the number of 250Mhz cycles is [csr_period * 16 * (gen1- 2:1)]. For x2, it is scaled by 2 and for x1, it is scaled by 4 from the x4 numbers. The timer has an accuracy of ±16 cycles. **Note:** A value of 0x0 is illegal.

Field	Bit(s)	Init.	Access Type	Description
FC_NP_PERIOD	14:10	0x3	RW	fc_np_period[4:0]. This specifies the latency after which the flow control packet is scheduled. The latency starts from the time at which the credit release is received by the link layer. For x4/x8/x16, the number of 250 MHz cycles is [csr_period * 16 * (gen1- 2:1)]. For x2, it is scaled by 2 and for x1, it is scaled by 4 from the x4 numbers. The timer has an accuracy of ±16 cycles. **Note:** A value of 0x0 is illegal.
FC_P_PERIOD	19:15	0x3	RW	fc_p_period[4:0] This specifies the latency after which the flow control packet is scheduled. The latency starts from the time at which the credit release is received by the link layer. For x4/x8/x16, the number of 250 MHz cycles is [csr_period * 16 * (gen1- 2:1)]. For x2, it is scaled by 2 and for x1, it is scaled by 4 from the x4 numbers. The timer has an accuracy of ±16 cycles. **Note:** A value of 0x0 is illegal.
ACK_PERIOD	23:20	0x3	RW	ack_period[3:0]. This specifies the latency after which a packet is ACKED. The timer starts running once it is internally received. For X4/X8/X16, the number of cycles is ack_period * 8 * (gen1 - 2 : 1). For X2, the number is scaled by 2 and for x4, the number is scaled by 4 from the X4 number. The timer has an accuracy for ±8 cycles. **Note:** A value of 0x0 is illegal.
RESERVED	31:24	0x1	RW	Reserved.

8.4.2.1.87 Retry Control Register - PXPRETRYCTRL (0x00000370)

Field	Bit(s)	Init.	Access Type	Description
REPLAY_TIMER	5:0	0x3	RW	replay_timer[5:0]. Specifies in µs the time after which a replay is initiated. Setting it to 0x0 disables the timer. The timer has an accuracy of ±1 µs.
UNUSEENTRINRTRYBUF	13:6	0x0	RW	unuseentrintrybuf[7:0]. The number of unusable entries in the retry buffer. There are 220 entries in RH. Setting it very high can cause the link to not send any TLP's.
REINIT_THRESHOLD	16:14	0x4	RW	reinit_threshold[2:0]. Specifies the number of retry attempts after which a recovery is initiated. **Note:** Logic 0 is an illegal setting.
RESERVED	31:17	0x0	RSV	Reserved.



8.4.2.1.88 Link Layer Misc Control Register - PXPLNKMISC0 (0x00000388)

Field	Bit(s)	Init.	Access Type	Description
DLLPMLANE	0	0b	RW	dllp_on_master_lane_only. When set to 1,b DLLP's start on lane 0 only. It is not sent in a cycle where TLP ends.
EDB_MODE_EN	1	1b	RW	edb_mode_en. When set to 1b, all packets that the transaction layer indicates are to be dumped have an EDB with them and the packets are resent with the *EP* bit set right after the EDB packet. When set to 0b, the packet is sent without EP.
RESERVED	2	0b	RW	Reserved.
RD_RETRY_BUF	3	0b	RW	rd_entry_buf. When set, the retry buffer can be read.
RESERVED	31:4	0x0	RSV	Reserved.

8.4.2.1.89 Credit Availability Status Register 0 - PXP0CDTAVAIL0 (0x0000038C)

Field	Bit(s)	Init.	Access Type	Description
CDT_AVAIL_IN	31:0	0x0	RO	11:0 = prd_cr_avail. 19:12 = prh_cr_avail. 31:20 = nprd_cr_avail.

8.4.2.1.90 Credit Availability Status Register 1 - PXP0CDTAVAIL1 (0x00000390)

Field	Bit(s)	Init.	Access Type	Description
CDT_AVAIL_IN	27:0	0x0	RO	11:0 = prd_cr_avail. 19:12 = prh_cr_avail. 31:20 = nprd_cr_avail.
RESERVED	31:28	0x0	RSV	Reserved.

8.4.2.1.91 Expected Bit Error Rate Register 0 - EXPBERR0 (0x00000394)

Field	Bit(s)	Init.	Access Type	Description
E_EXP_BERR	31:0	0xFFFFFFFF	RW	e_exp_berr[31:0]. Expected BER for leaky bucket error handling

8.4.2.1.92 Expected Bit Error Rate Register 1 - EXPBERR1 (0x00000398)

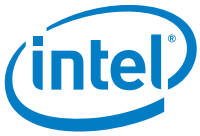
Field	Bit(s)	Init.	Access Type	Description
E_EXP_BERR	17:0	0x0	RW	e_exp_berr[49:32]. Expected BER for leaky bucket error handling.
RESERVED	31:18	0x0	RSV	Reserved.

8.4.2.1.93 Link Layer PM - LMPMB (0x000003A0)

Field	Bit(s)	Init.	Access Type	Description
WAIT_STATE_THRESH	5:0	0x1F	RW	Link pm delay before getting back to L1; after L1 exit due to packet transmission.
RESERVED	31:6	0x0	RSV	Reserved.

8.4.2.1.94 Rx PM Control Gen1 Register - RXPMCTRLGEN1 (0x000003D0)

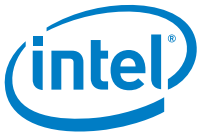
Field	Bit(s)	Init.	Access Type	Description
L0S_DIG_EXIT_TRIG	0	0b	RW	Gen1 L0s Digital Exit Trigger. Selects the digital trigger for electrical idle exit in Gen1 L0s. 0b = FTS detect. 1b = Reserved.
L0S_DIG_EXIT_THRESH	3:1	0x1	RW	Gen1 L0s Digital Exit Threshold. Threshold for the digital electrical idle exit in Gen1 L0s.



Field	Bit(s)	Init.	Access Type	Description
L0S_EI_EXIT_SEL	5:4	0x0	RW	Gen1 L0s Electrical Idle Exit Select. Selects the electrical idle exit in Gen1 L0s. 0 = Use analog squelch detector only. 1 = Use digital detector only. 2 = Use analog or digital. 3 = Use analog and digital. **Note:** if 1 or 3 selected - csr_gen1_l0s_active_lanes can't be set to zero and csr_gen1_l0s_dig_exit_thresh must be greater than zero.
L0S_ACTIVE_LANES	8:6	0x4	RW	Gen1 L0s Active Lanes. The maximum active lanes during Gen1 L0s. 000b = Shut down all lanes. 001b = Leave one lane active. 010b = Leave two lanes active (except x1 configuration). 100b = Leave all lanes active.
RESERVED	15:9	0x0	RSV	Reserved.
L1_DIG_EXIT_TRIG	16	0b	RW	Gen1 L1 Digital Exit Trigger. Selects the digital trigger for electrical idle exit in Gen1 L1. 0b = TS detect. 1b = Reserved.
L1_DIG_EXIT_THRESH	19:17	0x1	RW	Gen1 L1 Digital Exit Threshold. Threshold for the digital electrical idle exit in Gen1 L1.
L1_EI_EXIT_SEL	21:20	0x0	RW	Gen1 L1 Electrical Idle Exit Select. Selects the electrical idle exit in Gen1 L1. 0 = Use analog squelch detector only. 1 = Use digital detector only. 2 = Use analog or digital. 3 = Use analog and digital. **Note:** if 1 or 3 selected - csr_gen1_l1_active_lanes can't be set to zero and csr_gen1_l1_dig_exit_thresh must be greater than zero.
L1_ACTIVE_LANES	24:22	0x4	RW	Gen1 L1 Active Lanes. The maximum active lanes during Gen1 L1. 000b = Shut down all lanes. 001b = Leave one lane active. 010b = Leave two lanes active (except x1 configuration). 100b = Leave all lanes active.
RESERVED	31:25	0x0	RSV	Reserved.

8.4.2.1.95 Rx PM Control Gen2 Register - RXPMCTRLGEN2 (0x000003D4)

Field	Bit(s)	Init.	Access Type	Description
L0S_DIG_EXIT_TRIG	1:0	0x0	RW	Gen2 L0s Digital Exit Trigger. Selects the digital trigger for electrical idle exit in Gen2 L0s. 00b = FTS detect. 01b = EIE detect. 10b = FTS or EIE. 11b = Reserved.
L0S_DIG_EXIT_THRESH	4:2	0x1	RW	Gen2 L0s Digital Exit Threshold. Threshold for the digital electrical idle exit in Gen2 L0s.
L0S_EI_EXIT_SEL	6:5	0x0	RW	Gen2 L0s Electrical Idle Exit Select. Selects the electrical idle exit in Gen2 L0s: 0 = Use analog squelch detector only. 1 = Use digital detector only. 2 = Use analog or digital. 3 = Use analog and digital. **Note:** if 1 or 3 selected - csr_gen2_l0s_active_lanes can't be set to zero and csr_gen2_l0s_dig_exit_thresh must be greater than zero.
L0S_ACTIVE_LANES	9:7	0x4	RW	Gen2 L0s Active Lanes. The maximum active lanes during Gen2 L0s: 000b = Shut down all lanes. 001b = Leave one lane active. 010b = Leave two lanes active (except x1 configuration). 100b = Leave all lanes active.
RESERVED	15:10	0x0	RSV	Reserved.
L1_DIG_EXIT_TRIG	17:16	0x0	RW	Gen2 L1 Digital Exit Trigger. Selects the digital trigger for electrical idle exit in Gen2 L1. 00b = EIE detect. 01b = EIEOS detect. 10b = TS detect. 11b = TS or EIEOS.
L1_DIG_EXIT_THRESH	20:18	0x1	RW	Gen2 L1 Digital Exit Threshold. Threshold for the digital electrical idle exit in Gen2 L1.
L1_EI_EXIT_SEL	22:21	0x0	RW	Gen2 L1 Electrical Idle Exit Select. Selects the electrical idle exit in Gen2 L1: 0 = Use analog squelch detector only. 1 = Use digital detector only. 2 = Use analog or digital. 3 = Use analog and digital. **Note:** if 1 or 3 selected - csr_gen2_l1_active_lanes can't be set to zero and csr_gen2_l1_dig_exit_thresh must be greater than zero.



Field	Bit(s)	Init.	Access Type	Description
L1_ACTIVE_LANES	25:23	0x4	RW	Gen2 L1 Active Lanes. The maximum active lanes during Gen2 L1: 000b = Shut down all lanes. 001b = Leave one lane active. 010b = Leave two lanes active (except x1 configuration). 100b = Leave all lanes active.
RESERVED	31:26	0x0	RSV	Reserved.

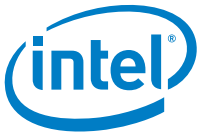
8.4.2.1.96 PIPE DFT Register - PIPE1 (0x000003D8)

Field	Bit(s)	Init.	Access Type	Description
DEEMP_OVEN	0	0b	RW	When high, pipe_Deemph signal can be overriden.
DEEMP	1	0b	RW	Override value for pipe_Deemph signal.
TXMARG_OVEN	2	0b	RW	Override enable for pipe_TxMarg signal.
TXMARG	5:3	0x0	RW	Override value for pipe_TxMarg signal.
RATE_OVEN	6	0b	RW	Override enable for pipe_rate signal.
RATE	7	0b	RW	Override value for pipe_rate signal.
TXEI_OVEN	8	0b	RW	Override enable for pipe_txelecidle signal.
TXEI	16:9	0x0	RW	Override value per lane for pipe_txelecidle signal.
RX_DETECTED	24:17	0x0	RW	Rx detection result per lane.
RESERVED	31:25	0x0	RSV	Reserved.

8.4.2.1.97 PIPE Register 2 - PIPE2 (0x000003DC)

Field	Bit(s)	Init.	Access Type	Description
LOW_SWING	0	0b	RW	Low swing value. High for low swing.
RESERVED	4:1	0x0	RSV	Reserved.
POWERDOWN_OVEN	5	0b	RW	Enables override PIPE power-down bus (DFT mode).
POWERDOWN_OV	7:6	0x0	RW	Override value for PIPE power-down bus (DFT mode).

Field	Bit(s)	Init.	Access Type	Description
DO_DETECT	15:8	0x0	RW	pipe_TxDetectRx Signal Force. When value changes from low to high, Rx detection on Tx lines are forced.
RXEI	23:16	0x0	RW	Rx Electrical Idle Status.
RESERVED	31:24	0x0	RSV	Reserved.



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

9.0 PCIe Programming Interface

9.1 PCI Compatibility

PCIe is fully compatible with existing deployed PCI software. To achieve this, PCIe hardware implementations conform to the following requirements:

- All devices are required to be supported by deployed PCI software and must be enumerable as part of a tree-through PCI device enumeration mechanisms.
- Devices must not require any resources such as address decode ranges and interrupts beyond those claimed by PCI resources for operation of software compatible and software transparent features with respect to existing deployed PCI software.
- Devices in their default operating state must confirm to PCI ordering and cache coherency rules from a software viewpoint.
- PCIe devices must conform to the PCI power management specification and must not require any register programming for PCI-compatible power management beyond those available through PCI power management capability registers. Power management is expected to conform to a standard PCI power management by existing PCI bus drivers.

PCIe devices implement all registers required by the PCI specification as well as the power management registers and capability pointers specified by the PCI power management specification. In addition, PCIe defines a PCIe capability pointer to indicate support for PCIe extensions and associated capabilities.

The X540 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
- Power management capabilities
- MSI / MSI-X capabilities
- Vital Product Data (VPD) capability
- PCIe extended capabilities:

— Advanced Error Reporting (AER)

— Serial ID

— Alternate requester ID.

— Single root IOV

9.2 Configuration Sharing Among PCI Functions

Note: Function 1 and LAN 1 are not supported by the X540 single port configuration.

The X540 contains a single physical PCIe core interface. It is designed so that each of the logical LAN devices (LAN 0, LAN 1) appears as a distinct function implementing its own PCIe device header space.

Many of the fields of the PCIe header space contain hardware default values that are either fixed or can be overridden using an NVM, but might not be independently specified for each logical LAN device. The following fields are considered to be common to both LAN functions:

Vendor ID	The Vendor ID of the X540 is specified to a single value 0x8086. The value is reflected identically for both LAN devices.
Revision	The revision number of the X540 is reflected identically for both LAN devices.
Header Type	This field indicates if a device is single function or multi-function. The value reflected in this field is reflected identically for both LAN devices, but the actual value reflected depends on LAN disable configuration. When both the X540 LAN ports are enabled, both PCIe headers return 0x80 in this field, acknowledging being part of a multi-function device. LAN 0 exists as device function 0, while LAN 1 exists as device function 1. If function 1 is disabled, then only a single-function device is indicated (this field returns a value of 0x00) and the LAN exists as device function 0.
Subsystem ID	The Subsystem ID of the X540 can be specified via an NVM, but only a single value can be specified. The value is reflected identically for both LAN devices.
Subsystem Vendor ID	The Subsystem Vendor ID of the X540 can be specified via an NVM, but only a single value can be specified. The value is reflected identically for both LAN devices.
Cap_Ptr Max Latency Min Grant	These fields reflect fixed values that are constant values reflected for both LAN devices.

The following fields are implemented as unique to each LAN functions:

Device ID	The Device ID reflected for each LAN function can be independently specified via an NVM.
Command Status	Each LAN function implements its own Command/Status registers.
Latency Timer Cache Line Size	Each LAN function implements these registers independently. The system should program these fields identically for each LAN to ensure consistent behavior and performance of each device.
Memory BAR, IO BAR Expansion ROM BAR MSIX BAR	Each LAN function implements its own Base Address registers, enabling each device to claim its own address region(s). The I/O BAR is enabled by the *IO_Sup* bit in the NVM.
Interrupt Pin	Each LAN function independently indicates which interrupt pin (INTA#...INTD#) is used by that device's MAC to signal system interrupts. The value for each LAN device can be independently specified via an NVM, but only if both LAN devices are enabled.
Class Code	Each function can have its own device class that can be: dummy function, LAN or storage as enabled by the NVM.

9.3 PCIe Register Map

Configuration registers are assigned one of the attributes described in the table that follows.

9.3.1 Register Attributes

The following table lists the register attributes used in this section.

RD/WR	Description
RO	Read-only register: Register bits are read-only and cannot be altered by software.
RW	Read-write register: Register bits are read-write and can be either set or reset.
RW1C	Read-only status, Write-1b-to-clear status register, Writing a 0b to RW1C bits has no effect.
ROS	Read-only register with sticky bits: Register bits are read-only and cannot be altered by software. Bits are 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: 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.

RD / WR	Description
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.

9.3.2 PCIe Configuration Space Summary

Table 9-1 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 9-1 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		Control Register	
	0x8	Class Code (0x020000/0x010000)			Revision ID
	0xC	Reserved	Header Type (0x0/ 0x80)	Latency Timer	Cache Line Size (0x10)
	0x10	Base Address Register 0			
	0x14	Base Address Register 1			
	0x18	Base Address Register 2			
	0x1C	Base Address Register 3			
	0x20	Base Address Register 4			
	0x24	Base Address Register 5			
	0x28	CardBus CIS pointer (0x0000)			
	0x2C	Subsystem ID		Subsystem Vendor ID	
	0x30	Expansion ROM Base Address			
	0x34	Reserved			Cap Ptr (0x40)
	0x38	Reserved			
	0x3C	Max Latency (0x00)	Min Grant (0x00)	Interrupt Pin (0x01...0x04)	Interrupt Line (0x00)
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			
Extended PCIe Configuration	0x100...0x12B	AER Capability			
	0z140...0x14B	Serial ID Capability			
	0x150...0x157	ARI Capability			
	0x160...0x19C	SR-IOV Capability			
	0x1D0...0x1D4	ACS Capability			

Table 9-2 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		Control Register	
	0x8	Class Code (0xFF0000)			Revision ID
	0xC	Reserved	Header Type (0x0/0x80)	Latency Timer	Cache Line Size (0x10)
	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 (0x01...0x04)	Interrupt Line (0x00)
Power Management Capability	0x40	Power Management Capabilities		Next Pointer (0xA0)	Capability ID (0x01)
	0x44	Data	Bridge Support Extensions	Power Management Control & Status	

Table 9-2 PCIe Configuration Registers Map (Continued)- Dummy Function

Section	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
PCIe Capability	0xA0	PCIe Capability Register (0x0002)		Next Pointer (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	
AER Capability	0x100	Next Capability Ptr. (0x140/0x150/0x1D0)	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			
Serial ID Capability	0x140	Next Capability Ptr. (0x150/0x1D0)	Version (0x1)	Serial ID Capability ID (0x0003)	
	0x144	Serial Number Register (Lower Dword)			
	0x148	Serial Number Register (Upper Dword)			

Table 9-2 PCIe Configuration Registers Map (Continued)- Dummy Function

Section	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
ARI capability	0x150	Next Capability Ptr. (0x1D0)	Version (0x1)	ARI Capability ID (0x000E)	
	0x154	ARI Control Register		ARI Capabilities	
ACS capability	0x1D0	Next Capability Ptr. (0x000)	Version (0x1)	ACS Capability ID (0x0D)	
	0x1D4	ACS Control Register (0x0)		ACS Capability Register (0x0)	

9.3.3 Mandatory PCI Configuration Registers — Except BARs

9.3.3.1 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.

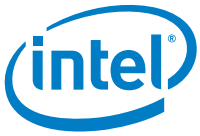
9.3.3.2 Device ID Register (0x2; RO)

This is a read-only register that identifies individual the X540 PCI functions. Both ports have the same default value equals to 0x1512, 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).

9.3.3.3 Command Register (0x4; RW)

Shaded bits are not used by this implementation and are hardwired to 0b. Each function has its own Command register. Unless explicitly specified, functionality is the same in both functions.

Bit(s)	Initial Value	Description
0	0b	I/O Access Enable.
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 – Hardwire to 0b.
4	0b	MWI Enable – Hardwire to 0b.



Bit(s)	Initial Value	Description
5	0b	Palette Snoop Enable – Hardwire to 0b.
6	0b	Parity Error Response.
7	0b	Wait Cycle Enable – Hardwired to 0b.
8	0b	SERR# Enable.
9	0b	Fast Back-to-Back Enable – Hardwire to 0b.
10	0b	Interrupt Disable. When set, devices are prevented from generating legacy interrupt messages.
15:11	0b	Reserved.

9.3.3.4 Status Register (0x6; RO)

Shaded bits are not used by this implementation and are hardwired to 0b. Each function has its own Status register. Unless explicitly specified, functionality is the same in both functions.

Bits	Initial Value	RW	Description
2:0	0b		Reserved.
3	0b	RO	Interrupt Status.[1]
4	1b	RO	New Capabilities: Indicates that a device implements extended capabilities. The X540 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 wire to 0b.
6	0b		Reserved.
7	0b		Fast Back-to-Back Capable – Hard wire to 0b.
8	0b	RW1C	Data Parity Reported.
10:9	00b		DEVSEL Timing – Hard wire 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.

9.3.3.5 Revision Register (0x8; RO)

The default revision ID of this device is 0x00. The value of the rev ID is a logic XOR between the default value and the value in the NVM PCIe Device Revision ID word. Note that LAN 0 and LAN 1 functions have the same revision ID.

9.3.3.6 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 PCIe Configuration (Offset 0x01).

- 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.

9.3.3.7 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. Loaded from the NVM. All functions are initialized to the same value.

9.3.3.8 Latency Timer (0xD; RO), Not Supported

Not used. Hard wire to 0b.

9.3.3.9 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 then this field has a value of 0x00 to indicate a single function device. If other functions are enabled then this field has a value of 0x80 to indicate a multi-function device. Table 9-3 lists the different options to set the header type field.

:

Table 9-3 Header Type Settings

Lan 0 Enable	Lan 1 Enable	Cross-Mode Enable	Dummy Function Enable	Header Type Expected Value
0	0	X	X	N/A (no function)
1	0	0	X	0x00
0	1	0	0	0x00
0	1	0	1	0x80 (dummy exist)
1	1	X	X	0x80 (dual function)

Table 9-3 Header Type Settings

Lan 0 Enable	Lan 1 Enable	Cross-Mode Enable	Dummy Function Enable	Header Type Expected Value
1	0	1	0	0x00
1	0	1	1	0x80 (dummy exist)
0	1	1	X	0x00

9.3.3.10 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.

9.3.4 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

9.3.5 Cap_Ptr Register (0x34; RO)

The *Capabilities Pointer* field (Cap_Ptr) is an 8-bit field that provides an offset in the X540's PCI configuration space for the location of the first item in the capabilities linked list. The X540 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.

9.3.5.1 Interrupt Line Register (0x3C; RW)

Read/write register programmed by software to indicate which of the system interrupt request lines the X540's interrupt pin is bound to. Refer to the PCI definition for more details. Each PCI function has its own register.

Max_Lat/Min_Gnt not used. Hard wired to 0b.

For Dummy functions this register is RO - zero.

9.3.5.2 Interrupt Pin Register (0x3D; RO)

Read-only register. LAN 0 / LAN 1[1] — A value of 0x1...0x4 indicates that this function implements a legacy interrupt on INTA#...INTD# respectively. Loaded from the NVM word offset 0x01 in the NVM PCIe Configuration Space per function. Refer to the following detailed explanation for cases in which any of the LAN port(s) are disabled.

9.3.6 Mandatory PCI Configuration Registers — BARs

9.3.6.1 Memory and IO Base Address Registers (0x10...0x27; RW)

Base Address Registers (BARs) are used to map the X540 register space of the device functions. The X540 has a memory BAR, I/O BAR and MSI-X BAR described in Table 9-4. The BARs location and sizes are described in the Table 9-4 and Table 9-5. The fields within each BAR are then described in Table 9-5.

Table 9-4 X540 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 *FLSize* and *CSRSize* fields in the BARCTRL register. Address 0 in the Flash device is mapped to address 128 K in the memory BAR. When the usable Flash size + CSR space are smaller than the memory BAR, then 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.

1. If only a single device/function of the X540 component is enabled, this value is ignored and the *Interrupt Pin* field of the enabled device reports INTA# usage.

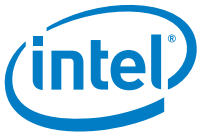


Table 9-5 Base Address Registers' Fields

Field	bits	RW	Description	
Memory and I/O Space Indication	0	RO	0b = Indicates memory space. 1b = Indicates I/O.	
Memory Type	2:1	RO	00b = Reserved10b = 64-bit BAR	
Prefetch Memory	3	R	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	The length of the RW bits and RO 0b bits depend on the mapping window sizes. Initial value of the RW fields is 0x0.	
			Mapping Window	RO bits
			Memory CSR + Flash BAR size depends on BARCTRL.FLSize and BARCTRL.CSRSize fields.	16:4 for 128 KB 17:4 for 256 KB and so on...
			MSI-X space is 16 KB.	13:4
			I/O space size is 32 bytes.	4:0

9.3.6.2 Expansion ROM Base Address Register (0x30; RW)

This register is used to define the address and size information for boot-time access to the Expansion ROM Module in the Flash memory. It is enabled by NVM words 0x24 and 0x14 for LAN 0 and LAN 1, respectively. This register returns a zero value for functions without an expansion ROM window and for dummy functions.

Field	Bit(s)	RW	Initial Value	Description
En	0	RW	0b	1b = Enables expansion ROM access. 0b = Disables expansion ROM access.
Reserved	10:1	R	0b	Always read as 0b. Writes are ignored.
Address	31:11	RW	0b	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 512 KB in powers of 2. Mapping window size is set by NVM word 0x0F.

9.3.7 PCIe Capabilities

The first entry of the PCI capabilities link list is pointed to by the Cap_Ptr register. Table 9-6 lists the capabilities supported by the X540.

Table 9-6 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 9-7 PCI Capabilities for Dummy Function

Address	Item	Next Pointer
0x40:47	PCI Power Management	0x50
0xA0:DB	PCIe Capabilities	0x00

9.3.7.1 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.

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	

9.3.7.1.1 Capability ID Register (0x40; RO)

This field equals 0x01 indicating the linked list item as being the PCI Power Management registers.

9.3.7.1.2 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.

9.3.7.1.3 Power Management Capabilities — PMC Register (0x42; RO)

This field describes the device functionality during the power management states as listed in the following table. Note that each device function has its own register.

Bits	Default	RW	Description
15:11	01001b	RO	PME_Support. This 5-bit field indicates the power states in which the function can assert PME#. Condition Functionality Values: • No AUX Pwr - PME at D0 and D3hot = 01001b • AUX Pwr - PME at D0, D3hot, and D3cold = 11001b Note: For dummy function, this field is RO - zero.
10	0b	RO	D2_Support – The X540 does not support the D2 state.
9	0b	RO	D1_Support – The X540 does not support the D1 state.
8:6	000b	RO	AUX Current – Required current defined in the Data register.
5	1b	RO	DSI – the X540 requires its device driver to be executed following a transition to the D0 uninitialized state.
4	0b	RO	Reserved.
3	0b	RO	PME_Clock – Disabled. Hard wire to 0b.
2:0	011b	RO	Version – The X540 complies with the PCI PM specification revision 1.2.

9.3.7.1.4 Power Management Control / Status Register — PMCSR (0x44; RW)

This register (shown in the following table) is used to control and monitor power management events in the device. Note that each device function has its own PMCSR.

Bits	Default	RW	Description
15	0b at power up	RW1CS	PME_Status. This bit is set to 1b when the function detects a wake-up event independent of the state of the *PME_En* bit. Writing a 1b clears this bit.
14:13	01b	RO	Data_Scale. This field indicates the scaling factor that's used when interpreting the value of the Data register. This field equals 01b (indicating 0.1 watt/units) and the Data_Select field is set to 0, 3, 4, 7, (or 8 for function 0). Otherwise, it equals 00b.

Bits	Default	RW	Description
12:9	0000b	RW	Data_Select. This 4-bit field is used to select which data is to be reported through the Data register and *Data_Scale* field. These bits are writeable only when power management is enabled via the NVM.
8	0b at power up	RWS	PME_En. If power management is enabled in the NVM, writing a 1b to this register enables Wakeup. If power management is disabled in the NVM, writing a 1b to this bit has no effect and does not set the bit to 1b.
7:4	0000b	RO	Reserved.
3	0b	RO	No_Soft_Reset. This bit is always set to 0b to indicate that the X540 performs an internal reset upon transitioning from D3hot to D0 via software control of the *PowerState* bits. Configuration context is lost when performing the soft reset. Upon transition from the D3hot to the D0 state, a full re-initialization sequence is needed to return the X540 to the D0 Initialized state.
2	0b	RO	Reserved for PCIe.
1:0	00b	RW	PowerState. This field is used to set and report the power state of a function as follows: 00b = D0. 01b = D1 (cycle ignored if written with this value). 10b = D2 (cycle ignored if written with this value). 11b = D3.

9.3.7.1.5 PMCSR_BSE Bridge Support Extensions Register (0x46; RO)

This register is not implemented in the X540; values set to 0x00.

9.3.7.1.6 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 if power management is enabled in the NVM or with a default value of 0x00. The values for the X540's functions are as follows:

Function	D0 (Consume/ Dissipate)	D3 (Consume/ Dissipate)	Common	Data_Scale/ Data_Select
	(0x0/0x4)	(0x3/0x7)	(0x8)	
0	EEP PCIe Control Word at offset 0x06	EEP PCIe Control Word at offset 0x06	Multi-function option: EEP PCIe Control Word at offset 0x06 Single-function option: 0x00	01b
1	EEP PCIe Control Word at offset 0x06	EEP PCIe Control Word at offset 0x06	0x00	01b

Note: For other Data_Select values the Data register output is reserved (0b).

9.3.7.2 MSI Capability

Note: This capability is not available for dummy functions.

This structure is required for PCIe devices.

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			

9.3.7.2.1 Capability ID Register (0x50; RO)

This field equals 0x05 indicating that the linked list item as being the MSI registers.

9.3.7.2.2 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.

9.3.7.2.3 Message Control Register (0x52; RW)

These register fields are listed in the following table. Note that there is a dedicated register (per PCI function) to separately enable its MSI.

Bits	Default	RW	Description
0	0b	RW	MSI Enable. 1b = Message Signaled Interrupts. The X540 generates an MSI for interrupt assertion instead of INTx signaling.
3:1	000b	RO	Multiple Messages Capable. The X540 indicates a single requested message per function.
6:4	000b	RO	Multiple Message Enable. The X540 returns 000b to indicate that it supports a single message per function.
7	1b	RO	64-bit Capable. A value of 1b indicates that the X540 is capable of generating 64-bit message addresses.

Bits	Default	RW	Description
8	1b[1]	RO	MSI per-vector masking. A value of 0b indicates that the X540 is not capable of per-vector masking. A value of 1b indicates that the X540 is capable of per-vector masking.
15:9	0b	RO	Reserved. Reads as 0b

1. The value is loaded from the MSI Mask bit in the NVM.

9.3.7.2.4 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.

9.3.7.2.5 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.

9.3.7.2.6 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.

9.3.7.2.7 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 X540 supports only one message, only bit 0 of these registers are implemented.

Bits	Default	RW	Description
0	0b	RW	MSI Vector 0 Mask. If set, the X540 is prohibited from sending MSI messages.
31:1	0x0	RO	Reserved

9.3.7.2.8 Pending Bits Register (0x64; RW)

Bits	Default	RW	Description
0	0b	RO	If set, the X540 has a pending MSI message.
31:1	0x0	RO	Reserved

9.3.8 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 9.3.8.2) typically contains multiple entries, each consisting of several fields: *Message Address*, *Message Upper Address*, *Message Data*, and *Vector Control*. Each entry is capable of specifying a unique vector.

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

9.3.8.1 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			

9.3.8.1.1 Capability ID Register (0x70; RO)

This field equals 0x11 indicating that the linked list item as being the MSI-X registers.

9.3.8.1.2 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.

9.3.8.1.3 Message Control Register (0x72; RW)

These register fields are listed in the following table. Note that there is a dedicated register (per PCI function).

Bits	Default	RW	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 X540 supports up to 64 different interrupt vectors per function. This field is loaded from the NVM MSI_X _N field.
13:11	0b	RO	Always returns 0b on a read. A write operation has no effect.
14	0b	RW	Function Mask. If 1b, all of the vectors associated with the function are masked, regardless of their per-vector *Mask* bit states. If 0b, each vector's *Mask* bit determines whether the vector is masked or not. Setting or clearing the MSI-X *Function Mask* bit has no effect on the state of the per-vector *Mask* bits.

Bits	Default	RW	Description
15	0b	RW	MSI-X Enable. If 1b and the MSI *Enable* bit in the MSI Message Control register is 0b, the function is permitted to use MSI-X to request service and is prohibited from using its INTx# pin. System configuration software sets this bit to enable MSI-X. A device driver is prohibited from writing this bit to mask a function's service request. If 0b, the function is prohibited from using MSI-X to request service.

9.3.8.1.4 MSI-X Table Offset Register (0x74; RW)

These register fields are listed in the following table.

Bits	Default	RW	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	0x000	RO	Table Offset. Used as an offset from the address contained in one of the function's BARs to point to the base of the MSI-X table. The lower three *Table BIR* bits are masked off (set to 0b) by software to form a 32-bit Qword-aligned offset. Note that this field is read only.

9.3.8.1.5 MSI-X Pending Bit Array — PBA Offset (0x78; RW)

This register fields are listed in the following table.

Bits	Default	RW	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.

9.3.8.2 MSI-X Table Structure

Dword3 — MSIXTVCTRL	Dword2 — MSIXTMSG	Dword1 — MSIXTUADD	Dword0 — MSIXTADD	Entry Number	BAR 3 — Offset
Vector Control	Msg Data	Msg Upper Addr	Msg Lower Addr	0	Base (0x0000)
Vector Control	Msg Data	Msg Upper Addr	Msg Lower Addr	1	Base + 1*16

Vector Control	Msg Data	Msg Upper Addr	Msg Lower Addr	2	Base + 2*16
…	…	…	…	…	
Vector Control	Msg Data	Msg Upper Addr	Msg Lower Addr	63	Base + 63*16
Vector Control	Msg Data	Msg Upper Addr	Msg Lower Addr	64	Base + 64*16
…	…	…	…	…	
Vector Control	Msg Data	Msg Upper Addr	Msg Lower Addr	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.

9.3.8.2.1 MSI–X Message Address Low — MSIXTADD (BAR3: 0x0 + 0x10*n, n=0...255; RW)

Bits	Default	Type	Description
1:0	0x00	RW	Message Address. For proper Dword alignment, software must always write zeroes 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.
31:2	0x00	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.

9.3.8.2.2 MSI–X Message Address High — MSIXTUADD (BAR3: 0x4 + 0x10*n, n=0...255; RW)

Bits	Default	Type	Description
31: 0	0x00	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.

9.3.8.2.3 MSI–X Message Data — MSIXTMSG (BAR3: 0x8 + 0x10*n, n=0...255; RW)

Bits	Default	Type	Description
31:0	0x00	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.

9.3.8.2.4 MSI–X Vector Control — MSIXTVCTRL (BAR3: 0xC + 0x10*n, n=0...255; RW)

Bits	Default	Type	Description
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.
31:1	0x00	RW	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.

9.3.8.3 MSI-X PBA Structure (BAR3: 0x2000 + 4*n, n=0...7; RW)

Field	Bit(s)	Init Val.	Description
PENBIT	31:0	0x0	MSI-X Pending Bits. Each bit is set to 1b when the appropriate interrupt request is set and cleared to 0b when the appropriate interrupt request is cleared. Bit 'i' in register 'N' is associated to MSI-X vector 32 * 'N' + 'i', 'N' = 0...3.

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.

9.3.9 CSR Access Via Configuration Address Space

Note: These registers are not available for dummy functions.

9.3.9.1 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)	R/W	Initial Value	Description
30:0	R/W[1]	0x0	Internal Register or Internal Memory location Address. 0x00000-0x1FFFF – Internal registers and memories. 0x20000-0x7FFFFFFF – Undefined.
31	R/W	0b	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 X540. Otherwise, accesses to the IODATA register are don't-cares (writes are discarded silently, reads return arbitrary results).

1. In the event that the *IO_by_cfg* bit in the PCIe Init Configuration 2 EEPROM word 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.

9.3.9.2 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)	R/W	Initial Value	Description
31:0	R/W[1]	0x0	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.

9.3.10 VPD Registers

Note: This capability is not available for dummy functions.

The X540 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.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xE0	VPD Address		Next Pointer (0x00)	Capability ID (0x03)
0xE4	VPD Data			

9.3.10.1 Capability ID Register (0xE0; RO)

This field equals 0x3 indicating the linked list item as being the VPD registers.

9.3.10.2 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.

9.3.10.3 VPD Address Register (0xE2; RW)

Word-aligned byte address of the VPD area in the NVM to be accessed. The register is read/write, and the initial value at power-up is indeterminate.

Bits	Default	Rd/Wr	Description
14:0	X	RW	Address: Dword-aligned byte address of the VPD area in the NVM to be accessed. The register is read/write, and the initial value at power-up is indeterminate. The two LSBs are RO as zero.
15	0b	RW	F: A flag used to indicate when the transfer of data between the VPD Data register and the storage component completes. The Flag register is written when the VPD Address register is written. 0b = Read. Set by hardware when data is valid. 1b = Write. Cleared by hardware when data is written to the NVM. The VPD address and data should not be modified before the action is done.

9.3.10.4 VPD Data Register (0xE4; RW)

VPD read/write data.

Bits	Default	Rd / Wr	Description
31:0	X	RW	VPD Data: VPD data can be read or written through this register. The LS byte of this register (at offset 4 in this capability structure) corresponds to the byte of VPD at the address specified by the VPD Address register. The data read from or written to this register uses the normal PCI byte transfer capabilities. Four bytes are always transferred between this register and the VPD storage component. Reading or writing data outside of the VPD space in the storage component is not allowed. In a write access, the data should be set before the address and the flag is set.

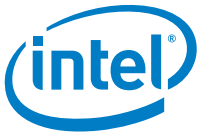
9.3.11 PCIe Configuration Registers

The X540 implements the PCIe capability structure linked to the legacy PCI capability list for endpoint devices as follows:

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xA0	PCI Express Capability Register (0x0002)		Next Pointer (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	

9.3.11.1 Capability ID Register (0xA0; RO)

This field equals 0x10 indicating that the linked list item as being the PCIe Capabilities registers.



9.3.11.2 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.

9.3.11.3 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.

Bits	Default	RW	Description
3:0	0010b	RO	Capability Version. Indicates the PCIe capability structure version. The X540 supports PCIe version 2 (also loaded from the PCIe *Capability Version* bit in the NVM).
7:4	0000b	RO	Device/Port Type. Indicates the type of PCIe functions. All functions are native PCI functions with a value of 0000b.
8	0b	RO	Slot Implemented. The X540 does not implement slot options. Therefore, this field is hard wired to 0b.
13:9	00000b	RO	Interrupt Message Number. The X540 does not implement multiple MSI per function. As a result, this field is hard wired to 0x0.
15:14	00b	RO	Reserved.

9.3.11.4 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.

Bits	Rd/Wr	Default	Description
2:0	RO	010b	Max Payload Size Supported. This field indicates the maximum payload that The X540 can support for TLPs. It is loaded from the NVM with a default value of 512 bytes.
4:3	RO	00b	Phantom Function Supported. Not supported by the X540.
5	RO	0b	Extended Tag Field Supported. Maximum supported size of the *Tag* field. The X540 supports a 5-bit *Tag* field for all functions.
8:6	RO	011b	Endpoint L0s Acceptable Latency. This field indicates the acceptable latency that the X540 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.

Bits	Rd / Wr	Default	Description
11:9	RO	110b	Endpoint L1 Acceptable Latency. This field indicates the acceptable latency that the X540 can withstand due to the transition from L1 state to the L0 state. A value of 110b equals 32 μs-64 μs. All functions share the same value loaded from the NVM.
12	RO	0b	Attention Button Present. Hard wired in the X540 to 0b for all functions.
13	RO	0b	Attention Indicator Present. Hard wired in the X540 to 0b for all functions.
14	RO	0b	Power Indicator Present. Hard wired in the X540 to 0b for all functions.
15	RO	1b	Role Based Error Reporting. Hard wired in the X540 to 1b for all functions.
17:16	RO	000b	Reserved 0b.
25:18	RO	0x00	Slot Power Limit Value. Used in upstream ports only. Hard wired in the X540 to 0x00 for all functions.
27:26	RO	00b	Slot Power Limit Scale. Used in upstream ports only. Hard wired in the X540 to 0b for all functions.
28	RO	1b	Function Level Reset Capability – A value of 1b indicates the Function supports the optional Function Level Reset (FLR) mechanism.
31:29	RO	0000b	Reserved.

9.3.11.5 Device Control Register (0xA8; RW)

This register controls the PCIe specific parameters. Note that there is a dedicated register per each function.

Bits	RW	Default	Description
0	RW	0b	Correctable Error Reporting Enable. Enable error report.
1	RW	0b	Non-Fatal Error Reporting Enable. Enable error report.
2	RW	0b	Fatal Error Reporting Enable. Enable error report.
3	RW	0b	Unsupported Request Reporting Enable. Enable error report.
4	RW	1b	Enable Relaxed Ordering. If this bit is set, the X540 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	RW	000b (128 bytes)	Max Payload Size. This field sets the maximum TLP payload size for the X540 functions. As a receiver, the X540 must handle TLPs as large as the set value. As a transmitter, the X540 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	RW	0b	Extended Tag field Enable. Not implemented in the X540.

Bits	RW	Default	Description
9	RW	0b	Phantom Functions Enable. Not implemented in the X540.
10	RWS	0b	Auxiliary Power PM Enable. When set, enables the X540 to draw AUX power independent of PME AUX power. The X540 is a multi-function device, therefore allowed to draw AUX power if at least one of the functions has this bit set.
11	RW	1b	Enable No Snoop. Snoop is gated by *No-Snoop* bits in the GCR register in the CSR space.
14:12	RW	010b	Max Read Request Size. This field sets maximum read request size for the X540 as a requester. 000b = 128 bytes. 001b = 256 bytes. 010b = 512 bytes. 011b = 1024 bytes. 100b = 2048 bytes. 101b = 4096 bytes (unsupported by the X540). 110b = Reserved. 111b = Reserved.
15	RW	0b	Initiate FLR – A write of 1b initiates FLR to the function. The value read by software from this bit is always 0b.

9.3.11.6 Device Status Register (0xAA; RW1C)

This register provides information about PCIe device specific parameters. Note that there is a dedicated register per each function.

Bits	RW	Default	Description
0	RW1C	0b	Correctable Detected. Indicates status of correctable error detection.
1	RW1C	0b	Non-Fatal Error Detected. Indicates status of non-fatal error detection.
2	RW1C	0b	Fatal Error Detected. Indicates status of fatal error detection.
3	RW1C	0b	Unsupported Request Detected. Indicates that the X540 received an unsupported request. This field is identical in all functions. The X540 can't distinguish which function causes the error.
4	RO	0b	Aux Power Detected. If Aux Power is detected, this field is set to 1b. It is a strapping signal from the periphery and is identical for all functions. Resets on LAN Power Good and PE_RST_N only.
5	RO	0b	Transaction Pending. Indicates whether the X540 has ANY transactions pending. (transactions include completions for any outstanding non-posted request for all used traffic classes).
15:6	RO	0x00	Reserved.

9.3.11.7 Link Capabilities Register (0xAC; RO)

This register identifies PCIe link-specific capabilities. This is a read-only register identical to all functions.

Bits	RW	Default	Description
3:0	RO	0010b	Supported Link Speeds. This field indicates the supported Link speed(s) of the associated link port. Defined encodings are: 0001b = 2.5 GbE link speed supported. 0010b = 5 GbE and 2.5 GbE link speeds supported.
9:4	RO	0x08	Max Link Width. Indicates the maximum link width. The X540 supports a x1, x2, 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. Defined encoding: 000000b = Reserved. 000001b = x1. 000010b = x2. 000100b = x4. 001000b = x8.
11:10	RO	11b	Active State Link PM Support. Indicates the level of the active state of power management supported in the X540. 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 Act_Stat_PM_Sup field in the NVM PCIe Init Configuration 3 offset 0x3.
14:12	RO	001b (64-128 ns)	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	RO	111b	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	RO	0	Clock Power Management

Bits	RW	Default	Description
19	RO	0	Surprise Down Error Reporting Capable. Hard wired to 0b.
20	RO	0	Data Link Layer Link Active Reporting Capable.
21	RO	0	Link Bandwidth Notification Capability. Hard wired to 0b.
22	RO	1b	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 whether to run ASPM compliance tests.
23	RO	0b	Reserved
31:24	HwInit	0x0	Port Number. The PCIe port number for the given PCIe link. This field is set in the link training phase.

9.3.11.8 Link Control Register (0xB0; RO)

This register controls PCIe link specific parameters. There is a dedicated register per each function.

Bits	RW	Default	Description
1:0	RW	00b	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 X540. 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	RO	0b	Reserved
3	RW	0b	Read Completion Boundary.
4	RO	0b	Link Disable. Reserved for endpoint devices. Hard wired to 0b.
5	RO	0b	Retrain Clock. Not applicable for endpoint devices. Hard wire to 0b.

Bits	RW	Default	Description
6	RW	0b	Common Clock Configuration. When set, indicates that the X540 and the component at the other end of the link are operating with a common reference clock. A value of 0b indicates that they are operating with an asynchronous clock. This parameter affects the L0s exit latencies. In ARI mode, the common clock configuration is determined solely by the field in function 0 (even when it is a dummy function) while it is meaningless in the other function(s).
7	RW	0b	Extended Sync. When set, this bit forces an extended Tx of the FTS ordered set in FTS and an extra TS1 at the exit from L0s prior to entering L0.
8	RO	0b	Reserved.
9	RW	0b	Reserved. Returns the value that was written.
10	RO	0b	Link Bandwidth Management Interrupt Enable. Not supported in the X540. Hard wired to 0.
11	RO	0b	Link Autonomous Bandwidth Interrupt Enable.Not supported in the X540. Hard wired to 0.
15:12	RO	0x0	Reserved

9.3.11.9 Link Status Register (0xB2; RO)

This register provides information about PCIe Link specific parameters. This is a read only register identical to all functions.

Bits	RW	Default	Description
3:0	RO	0001b	Current Link Speed. This field indicates the negotiated link speed of the given PCIe link. Defined encodings are: 0001b = 2.5 GbE PCIe link. 0010b = 5 GbE PCIe link. All other encodings are reserved.
9:4	RO	000001b	Negotiated Link Width. Indicates the negotiated width of the link. Relevant encodings for the X540 are: 000001b = x1. 000010b = X2. 000100b = x4. 001000b = x8.
10	RO	0b	Undefined.
11	RO	0b	Link Training. Indicates that link training is in progress. This field is not applicable and is reserved for endpoint devices, and is hard wired to 0b.

Bits	RW	Default	Description
12	HwInit	1b	Slot Clock Configuration. When set, indicates that the X540 uses the physical reference clock that the platform provides at the connector. This bit must be cleared if the X540 uses an independent clock. The *Slot Clock Configuration* bit is loaded from the *Slot_Clock_Cfg* NVM bit.
13	RO	0b	Data Link Layer Link Active. Not supported in the X540. Hard wire to 0b.
14	RO	0b	Link Bandwidth Management Status. Not supported in the X540. Hard wire to 0b.
15	RO	0b	Link Autonomous Bandwidth Status. This bit is not applicable and is reserved for endpoints.

The following registers are supported only if the capability version is two and above.

9.3.11.10 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.

Bits	RW	Default	Description
3:0	RO	1111b	Completion Timeout Ranges Supported. This field indicates the X540'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 X540 supports. • 0000b = Completion timeout programming not supported. The X540 must implement a timeout value in the range of 50 µs to 50 ms. • 0001b = Range A. • 0010b = Range B. • 0011b = Ranges A and B. • 0110b = Ranges B and C. • 0111b = Ranges A, B and C. • 1110b = Ranges B, C and D. • 1111b = Ranges A, B, C and D. • All other values are reserved.
4	RO	1b	Completion Timeout Disable Supported 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	RO	0b	ARI Forwarding Supported. Applicable only to Switch Downstream. Ports and Root Ports; must be 0b for other function types.
31:6	RO	0x0000	Reserved

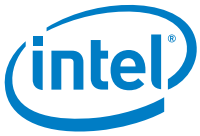
9.3.11.11 Device Control 2 Register (0xC8; RW)

This register controls the PCIe specific parameters. Note that there is a dedicated register per each function.

Bits	RW	Default	Description
3:0	RW	0x0	Completion Timeout Value. For devices that support completion timeout programmability, this field enables system software to modify the completion timeout value. Defined encodings: • 0000b = Default range: 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	RW	0b	Completion Timeout Disable. When set to 1b, this bit disables the completion timeout mechanism. Software is permitted to set or clear this bit at any time. When set, the completion timeout detection mechanism is disabled. If there are outstanding requests when the bit is cleared, it is permitted but not required for hardware to apply the completion timeout mechanism to the outstanding requests. If this is done, it is permitted to base the start time for each request on either the time this bit was cleared or the time each request was issued. Note: For dummy function, this field is RO - zero.
5	RO	0b	ARI Forwarding Enable. Applicable only to switch devices.
15:6	RO	0b	Reserved.

9.3.11.12 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.



Bits	RW	Default	Description
3:0	RWS	0010b	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. Defined encodings are: 0001b = 2.5 GbE target link speed. 0010b = 5 GbE target link speed. All other encodings are reserved. 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 X540 (as reported in the *Supported Link Speeds* field of the Link Capabilities register).
4	RWS	0b	Enter Compliance. Software is permitted to force a link to enter compliance mode at the speed indicated in the *Target Link Speed* field by setting this bit to 1b in both components on a link and then initiating a hot reset on the link. The default value of this field following a fundamental reset is 0b.
5	RWS	0b	Hardware Autonomous Speed Disable. When set to 1b, this bit disables hardware from changing the link speed for reasons other than attempting to correct unreliable link operation by reducing link speed.
6	RO	0b	Selectable De-Emphasis. This bit is not applicable and reserved for endpoints.
9:7	RWS	000b	Transmit Margin. This field controls the value of the non de emphasized voltage level at the Transmitter pins. Encodings: 000b = Normal operating range. 001b = 800-1200 mV for full swing and 400-700 mV for half-swing. 010b = (n-1) — Values must be monotonic with a non-zero slope. The value of n must be greater than 3 and less than 7. At least two of these must be below the normal operating range of n: 200-400 mV for full-swing and 100-200 mV for half-swing. 111b= (n) reserved.
10	RWS	0b	Enter Modified Compliance. When this bit is set to 1b, the device transmits modified compliance pattern if the LTSSM enters Polling.Compliance state.
11	RWS	0b	Compliance SOS. When set to 1b, the LTSSM is required to send SOS periodically in between the (modified) compliance patterns.

9.3.11.13 Link Status 2 Register (0xD2; RW)

Bits	RW	Default	Description
0	RO	0b	Current De-emphasis Level. When the link is operating at 5 GT/s speed, this bit reflects the level of de-emphasis. it is undefined when the Link is operating at 2.5 GT/s speed Encodings: 1b -3.5 dB. 0b -6 dB.

9.4 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 X540 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 X540 supports the following PCIe extended capabilities:

Table 9-8 Extended Capabilities list

Capability	Offset	Next Header
Advanced Error Reporting Capability	0x100	0x140/0x150/0x160/0x1D0[1]
Serial Number	0x140	0x150//0x160/0x1D0[1]
Alternative RID Interpretation (ARI)	0x150	0x160/0x1D0[1]
IOV support[2]	0x160	0x1D0
Access Control Services	0x1D0	0x000

1. Depends on NVM settings enabling the serial numbers and ARI/IOV structures. Serial Number capability is not exposed in case the first 4 bytes of the serial number read from NVM are all 0s.
2. In a dummy function, the IOV structure is not exposed.

9.4.1 Advanced Error Reporting Capability (AER)

The PCIe advanced error reporting capability is an optional extended capability to support advanced error reporting. The tables that follow list the PCIe advanced error reporting extended capability structure for PCIe devices.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x100	Next Capability Ptr. (0x140/0x150/0x160/ 0x1D0[1])	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			

1. Depends on NVM settings enabling the serial number and ARI/IOV structures.

9.4.1.1 Advanced Error Reporting Enhanced Capability Header Register (0x100; RO)

Bit Location	Attribute	Default Value	Description
15:0	RO	0x0001	Extended Capability ID. PCIe extended capability ID indicating advanced error reporting capability.
19:16	RO	0x2	Version Number. PCIe advanced error reporting extended capability version number.
31:20	RO	0x0140/ 0x0150/ 0x0160/ 0x01D0/ 0x0000	Next Capability Pointer. Next PCIe extended capability pointer. See Table 9-8 and Table 9-7 for possible values of the next capability pointer.

9.4.1.2 Uncorrectable Error Status Register (0x104; RW1CS)

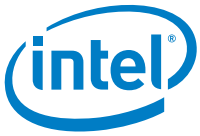
The Uncorrectable Error Status register reports error status of individual uncorrectable error sources on a PCIe device. An individual error status bit that is set to 1b indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by LAN_PWR_GOOD.

Bit Location	Attribute	Default Value	Description
3:0	RO	0b	Reserved
4	RW1CS	0b	Data Link Protocol Error Status.
11:5	RO	0b	Reserved
12	RW1CS	0b	Poisoned TLP Status.
13	RW1CS	0b	Flow Control Protocol Error Status.
14	RW1CS	0b	Completion Timeout Status.
15	RW1CS	0b	Completer Abort Status.
16	RW1CS	0b	Unexpected Completion Status.
17	RW1CS	0b	Receiver Overflow Status.
18	RW1CS	0b	Malformed TLP Status.
19	RW1CS	0b	ECRC Error Status.
20	RW1CS	0b	Unsupported Request Error Status.
21	RO	0b	ACS Violation Status.
31:22	RO	0b	Reserved

9.4.1.3 Uncorrectable Error Mask Register (0x108; RWS)

The Uncorrectable Error Mask register controls reporting of individual uncorrectable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. Note that there is a mask bit per bit of the Uncorrectable Error Status register.

Bit Location	Attribute	Default Value	Description
3:0	RO	0b	Reserved
4	RWS	0b	Data Link Protocol Error Mask.
11:5	RO	0b	Reserved
12	RWS	0b	Poisoned TLP Mask.
13	RWS	0b	Flow Control Protocol Error Mask.
14	RWS	0b	Completion Timeout Mask.



Bit Location	Attribute	Default Value	Description
15	RWS	0b	Completer Abort Mask.
16	RWS	0b	Unexpected Completion Mask.
17	RWS	0b	Receiver Overflow Mask.
18	RWS	0b	Malformed TLP Mask.
19	RWS	0b	ECRC Error Mask.
20	RWS	0b	Unsupported Request Error Mask.
21	RO	0b	ACS Violation Mask.
31:22	RO	0b	Reserved

9.4.1.4 Uncorrectable Error Severity Register (0x10C; RWS)

The Uncorrectable Error Severity register controls whether an individual uncorrectable error is reported as a fatal error. An uncorrectable error is reported as fatal when the corresponding error bit in the severity register is set. If the bit is cleared, the corresponding error is considered non-fatal.

Bit Location	Attribute	Default Value	Description
3:0	RO	0b	Reserved
4	RWS	1b	Data Link Protocol Error Severity.
11:5	RO	0000001b	Reserved
12	RWS	0b	Poisoned TLP Severity.
13	RWS	1b	Flow Control Protocol Error Severity.
14	RWS	0b	Completion Timeout Severity.
15	RWS	0b	Completer Abort Severity.
16	RWS	0b	Unexpected Completion Severity.
17	RWS	1b	Receiver Overflow Severity.
18	RWS	1b	Malformed TLP Severity.
19	RWS	0b	ECRC Error Severity.
20	RWS	1b	Unsupported Request Error Severity.
21	RO	0b	ACS Violation Severity.

Bit Location	Attribute	Default Value	Description
22	RO	1b	Uncorrectable Internal Error Severity.
31:23	RO	000000000b	Reserved

9.4.1.5 Correctable Error Status Register (0x110; RW1CS)

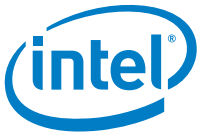
The Correctable Error Status register reports error status of individual correctable error sources on a PCIe device. When an individual error status bit is set to 1b it indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by LAN_PWR_GOOD.

Bit Location	Attribute	Default Value	Description
0	RW1CS	0b	Receiver Error Status.
5:1	RO	0b	Reserved
6	RW1CS	0b	Bad TLP Status.
7	RW1CS	0b	Bad DLLP Status.
8	RW1CS	0b	REPLAY_NUM Rollover Status.
11:9	RO	0b	Reserved
12	RW1CS	0b	Replay Timer Timeout Status.
13	RW1CS	0b	Advisory Non-Fatal Error Status.
15:14	RO	0b	Reserved

9.4.1.6 Correctable Error Mask Register (0x114; RWS)

The Correctable Error Mask register controls reporting of individual correctable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. There is a mask bit per bit in the Correctable Error Status register.

Bit Location	Attribute	Default Value	Description
0	RWS	0b	Receiver Error Mask.
5:1	RO	0b	Reserved



Bit Location	Attribute	Default Value	Description
6	RWS	0b	Bad TLP Mask.
7	RWS	0b	Bad DLLP Mask.
8	RWS	0b	REPLAY_NUM Rollover Mask.
11:9	RO	0b	Reserved
12	RWS	0b	Replay Timer Timeout Mask.
13	RWS	1b	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	RO	0b	Reserved

9.4.1.7 Advanced Error Capabilities and Control Register (0x118; RO)

Bit Location	Attribute	Default Value	Description
4:0	ROS	0b	Vector pointing to the first recorded error in the Uncorrectable Error Status register. This is a read-only field that identifies the bit position of the first uncorrectable error reported in the Uncorrectable Error Status register.
5	RO	0b	ECRC Generation Capable. If set, this bit indicates that the function is capable of generating ECRC. This bit is loaded from NVM.
6	RWS	0b	ECRC Generation Enable. When set, ECRC generation is enabled.
7	RO	1b	ECRC Check Capable. If set, this bit indicates that the function is capable of checking ECRC. This bit is loaded from NVM.
8	RWS	0b	ECRC Check Enable.When set Set, ECRC checking is enabled.

9.4.1.8 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 Location	Attribute	Default Value	Description
127:0	ROS	0b	Header of the packet in error (TLP or DLLP).

9.4.2 Serial Number

The PCIe device serial number capability is an optional extended capability that can be implemented by any PCIe device. The device serial number is a read-only 64-bit value that is unique for a given PCIe device.

All multi-function devices that implement this capability must implement it for function 0; other functions that implement this capability must return the same device serial number value as that reported by function 0.

When bytes 3:0 of the PCIe Serial Number Ethernet MAC Address NVM CSR are set to zero (Section 6.4.4.15 and Section 6.4.4.16), the PCIe serial number capability is disabled.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x140	Next Capability Ptr. (0x150/0x160/0X1D0)	Version (0x1)	Serial ID Capability ID (0x0003)	
0x144	Serial Number Register (Lower Dword)			
0x148	Serial Number Register (Upper Dword)			

9.4.2.1 Device Serial Number Enhanced Capability Header Register (0x140; RO)

Bit(s)	Default value	Attributes	Description
15:0	0x0003	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	0x150/ 0x160/ 0X1D0	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. The value of this field is 0x150 to point to the ARI capability structure. If ARI/IOV and Serial ID are disabled in NVM this field is zero. See Table 9-8. In a dummy function the value of this field is 0x150 to point to the ARI capability structure. See Table 9-7.

9.4.2.2 Serial Number Registers (0x144:0x148; RO)

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit Extended Unique Identifier (EUI-64*). The register at offset 0x144 holds the lower 32 bits and the register at offset 0x148 holds the higher 32 bits. The following figure details

the allocation of register fields in the Serial Number register. The table that follows provides the respective bit definitions.

Bit(s) Location	Attributes	Description
63:0	RO	PCIe Device Serial Number. This field contains the IEEE defined 64-bit EUI-64*. This identifier includes a 24-bit company ID value assigned by IEEE registration authority and a 40-bit extension identifier assigned by the manufacturer.

The serial number uses the Ethernet MAC address according to the following definition:

Field	Company ID			Extension Identifier				
Order	Addr+0	Addr+1	Addr+2	Addr+3	Addr+4	Addr+5	Addr+6	Addr+7
Most Significant Byte						Least Significant Byte		
Most Significant Bit						Least Significant Bit		

The serial number can be constructed from the 48-bit Ethernet MAC address in the following form:

Field	Company ID			MAC Label		Extension identifier		
Order	Addr+0	Addr+1	Addr+2	Addr+3	Addr+4	Addr+5	Addr+6	Addr+7
Most Significant Bytes						Least Significant Byte		
Most Significant Bit						Least Significant Bit		

In this case, the MAC label is 0xFFFF.

For example, assume that the company ID is (Intel) 00-A0-C9 and the extension identifier is 23-45-67. 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 Serial Number Ethernet MAC Address NVM field (not the same field that is loaded from NVM into the RAL and RAH registers).

Note: The official document that defines EUI-64* is: http://standards.ieee.org/regauth/oui/tutorials/EUI64.html

9.4.3 Alternate Routing ID Interpretation (ARI) Capability Structure

In order to allow more than eight functions per endpoint without requesting an internal switch, as is usually needed in virtualization scenarios, the PCI-SIG defines a new capability that allows a different interpretation of the *Bus*, *Device*, and *Function* fields. The ARI capability structure is as follows:

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x150	Next Capability Ptr. (0x160/0x1D0)	Version (0x1)	ARI Capability ID (0x000E)	
0x154	ARI Control Register		ARI Capabilities	

9.4.3.1 PCIe ARI Header Register (0x150; RO)

Field	Bit(s)	Initial Value	Access	Description
ID	15:0	0x000E	RO	PCIe Extended Capability ID. PCIe extended capability ID for the alternative RID interpretation.
Version	19:16	1b	RO	Capability Version. This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 0x1 for this version of the specification.
Next Capability Ptr.	31:20	0x160/ 0x1D0	RO	Next Capability Offset.This field contains the offset to the next PCIe extended capability structure. The value of the 0x160 points to the IOV structure. The value od 0x1D0 points to the ACS structure if IOV is disabled.

9.4.3.2 PCIe ARI Capabilities and Control Register (0x154; RO)

Field	Bit(s)	Initial Value	Access	Description
Reserved	0	0b	RO	Not supported in the X540.
Reserved	1	0b	RO	Not supported in the X540.
Reserved	7:2	0b	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.

Field	Bit(s)	Initial Value	Access	Description
Reserved	16	0b	RO	Not supported in the X540.
Reserved	17	0b	RO	Not supported in the X540.
Reserved	19:18	00b	RO	Reserved
Reserved	22:20	0b	RO	Not supported in the X540.
Reserved	31:23	0b	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.

9.4.4 IOV Capability Structure

Note: This capability structure is not exposed in a dummy function.

This is the new structure used to support the IOV capabilities reporting and control. The following tables shows the possible implementations of this structure in the X540.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x150	Next Capability Ptr. (0x160)	Version (0x1)	ARI Capability ID (0x000E)	
0x154	ARI Control Register		ARI Capabilities	
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)			
0x18C	VF BAR2 (RO)			
0x190	VF BAR3 — Low (RW)			
0x194	VF BAR3- High (RW)			
0x198	VF BAR5 (RO)			
0x19C	VF Migration State Array Offset (RO)			

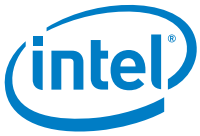
9.4.4.1 PCIe SR-IOV Header Register (0x160; RO)

Field	Bit(s)	Initial Value	Access	Description
ID	15:0	0x0010	RO	PCIe Extended Capability ID. PCIe extended capability ID for the SR-IOV capability.
Version	19:16	0x1	RO	Capability Version. This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 0x1 for this version of the specification.
Next pointer	31:20	0x0	RO	Next Capability Offset. This field contains the offset to the next PCIe extended capability structure or 0x000 if no other items exist in the linked list of capabilities.

9.4.4.2 PCIe SR-IOV Capabilities Register (0x164; RO)

Field	Bit(s)	Initial Value	Access	Description
Reserved	0	0b	RO	Not supported in the X540.
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	20:2	0x0	RO	Reserved
Reserved	31:21	0x0	RO	Not supported in the X540.

1. Set on first function where SR-IOV is enabled (see IOV enabled bit in IOV Control Word 1) and Read Only Zero in the other PF.



9.4.4.3 PCIe SR-IOV Control/Status Register (0x168; RW)

Field	Bit(s)	Initial Value	Access	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	1	0b	RO	Not supported in the X540.
Reserved	2	0b	RO	Not supported in the X540.
VF MSE	3	0b	RW	Memory Space Enable for Virtual Functions. VF MSE controls memory space enable for all VFs associated with this PF as with the Memory Space Enable bit in a functions PCI command register. The default value for this bit is 0b. When VF Enable is 1, virtual function memory space access is permitted only when VF MSE is Set. VFs shall follow the same error reporting rules as defined in the base specification if an attempt is made to access a virtual functions memory space when VF Enable is 1 and VF MSE is zero. Implementation Note: Virtual functions memory space cannot be accessed when VF Enable is zero. Thus, VF MSE is "don't care" when VF Enable is zero, however, software may choose to set VF MSE after programming the VF BARn registers, prior to setting VF Enable to 1.
ARI Capable Hierarchy	4	0b	RW (first function where SR-IOV is enabled) ROS (other function)[1]	ARI Capable Hierarchy. The X540 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 9.4.4.2) 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.

Field	Bit(s)	Initial Value	Access	Description
Reserved	15:5	0x0	RO	Reserved
Reserved	16	0b	RO	Not implemented in the X540.
Reserved	31:17	0b	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.

9.4.4.4 PCIe SR-IOV Max/Total VFs Register (0x16C; RO)

Field	Bit(s)	Initial Value	Access	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 X540 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 IOV Control Word 1 in the NVM.

9.4.4.5 PCIe SR-IOV Num VFs Register (0x170; RW)

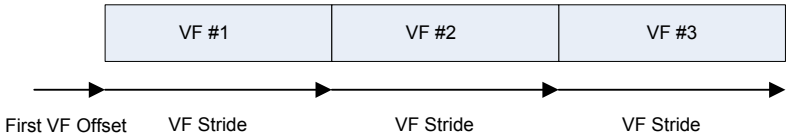
Field	Bit(s)	Initial Value	Access	Description
NumVFs	15:0	0x0	RW	Num VFs defines the number of VFs software has assigned to the PF. Software sets NumVFs to any value between one and the TotalVFs as part of the process of creating VFs. NumVFs VFs must be visible in the PCIe fabric after both NumVFs is set to a valid value 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 X540 there are no constraints.
Reserved	31:24	0	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.

9.4.4.6 PCIe SR-IOV VF RID Mapping Register (0x174; RO)

Field	Bit(s)	Initial Value	Access	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.
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 7.10.2.6.1.



9.4.4.7 PCIe SR-IOV VF Device ID Register (0x178; RO)

Field	Bit(s)	Initial Value	Access	Description
DEVID	31:16	0x1515	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

9.4.4.8 PCIe SR-IOV Supported Page Size Register (0x17C; RO)

Field	Bit(s)	Initial Value	Access	Description
Supported page Size	31:0	0x553	RO	For PFs that supports the stride-based BAR mechanism, this field defines the supported page sizes. This PF supports a page size of 2^(n+12) if bit n is set. For example, if bit 0 is Set, the Endpoint (EP) supports 4KB page sizes. Endpoints are required to support 4 KB, 8 KB, 64 KB, 256 KB, 1 MB and 4 MB page sizes. All other page sizes are optional.

9.4.4.9 PCIe SR-IOV System Page Size Register (0x180; RW)

Field	Bit(s)	Initial Value	Access	Description
Page size	31:0	0x1	RW	This field defines the page size the system uses to map the 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.

9.4.4.10 PCIe SR-IOV BAR 0 — Low Register (0x184; RW)

Field	Bit(s)	Initial Value	Access	Description
Mem	0	0b	RO	0b indicates memory space.
Mem Type	2:1	10b	RO	Indicates the address space size. 10b = 64-bit. This bit is loaded from the IOV Control word in the NVM.

Field	Bit(s)	Initial Value	Access	Description
Prefetch Mem	3	0b*	RO	0b = Non-prefetchable space. 1b = Prefetchable space. This bit is loaded from the IOV Control word in the NVM.
Memory Address Space	31:4	0x0	RW	Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size. The size is a maximum between 16 KB and the page size.

9.4.4.11 PCIe SR-IOV BAR 0 — High Register (0x188; RW)

Field	Bit(s)	Initial Value	Access	Description
BAR0 — MSB	31:0	0x0	RW	MSB part of BAR0.

9.4.4.12 PCIe SR-IOV BAR 2 Register (0x18C; RO)

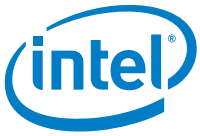
Field	Bit(s)	Initial Value	Access	Description
BAR2	31:0	0x0	RO	This BAR is not used.

9.4.4.13 PCIe SR-IOV BAR 3 — Low Register (0x190; RW)

Field	Bit(s)	Initial Value	Access	Description
Mem	0	0b	RO	0b indicates memory space.
Mem Type	2:1	10b	RO	Indicates the address space size. 10b = 64-bit. This bit is loaded from the IOV Control word in the NVM.
Prefetch Mem	3	0b*	RO	0b = Non-prefetchable space 1b = Prefetchable space This bit is loaded from the IOV Control word in the NVM.
Memory Address Space	31:4	0x0	RW	Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size. The size is a maximum between 16 KB and page size.

9.4.4.14 PCIe SR-IOV BAR 3 — High Register (0x194; RW)

Field	Bit(s)	Initial Value	Access	Description
BAR3 — MSB	31:0	0x0	RW	MSB part of BAR3.



9.4.4.15 PCIe SR-IOV BAR 5 Register (0x198; RO)

Field	Bit(s)	Initial Value	Access	Description
BAR5	31:0	0x0	RO	This BAR is not used.

9.4.4.16 PCIe SR-IOV VF Migration State Array Offset Register (0x19C; RO)

Field	Bit(s)	Initial Value	Access	Description
Reserved	2:0	0x0	RO	Not implemented in the X540.
Reserved	31:0	0x0	RO	Not implemented in the X540.

9.4.5 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.

The following table lists the PCIe ACS extended capability structure for PCIe devices.

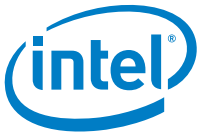
Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1D0	Next Capability Ptr. (0x000)	Version (0x1)	ACS Capability ID (0x0D)	
0x1D4	ACS Control Register (0x0)		ACS Capability Register (0x0)	

9.4.5.1 ACS CAP ID (0x1D0; RO)

Bit Location	Attribute	Default Value	Description
15:0	RO	0x0D	ACS Capability ID PCIe extended capability ID indicating ACS capability.
19:16	RO	0x1	Version Number PCIe ACS extended capability version number.
31:20	RO	0x000	Next Capability Pointer This is the last capability, so the next pointer is 0x000.

9.4.5.2 ACS Control and Capabilities (0x1D4; RO)

Bit Location	Attribute	Default Value	Description
0	RO	0b	ACS Source Validation (V) – Hardwired to zero, not supported in the X540.
1	RO	0b	ACS Translation Blocking (B) – Hardwired to zero, not supported in the X540.
2	RO	0b	ACS P2P Request Redirect (R) – Hardwired to zero, not supported in the X540.
3	RO	0b	ACS P2P Completion Redirect (C) – Hardwired to zero, not supported in the X540.
4	RO	0b	ACS Upstream Forwarding (U) – Hardwired to zero, not supported in the X540.
5	RO	0b	ACS P2P Egress Control (E) – Hardwired to zero, not supported in the X540.
6	RO	0b	ACS Direct Translated P2P (T) – Hardwired to zero, not supported in the X540.
7	Rsrv	0b	Reserved
15:8	RO	0x0	Egress Control Vector Size – Hardwired to zero, not supported in the X540.
16	RO	0b	ACS Source Validation Enable (V) – Hardwired to zero, not supported in the X540.
17	RO	0b	ACS Translation Blocking Enable (B) – Hardwired to zero, not supported in the X540.
18	RO	0b	ACS P2P Request Redirect Enable (R) – Hardwired to zero, not supported in the X540.
19	RO	0b	ACS P2P Completion Redirect Enable (C) – Hardwired to zero, not supported in the X540.
20	RO	0b	ACS Upstream Forwarding Enable (U) – Hardwired to zero, not supported in the X540.
21	RO	0b	ACS P2P Egress Control Enable (E) – Hardwired to zero, not supported in the X540.
22	RO	0b	ACS Direct Translated P2P Enable (T) – Hardwired to zero, not supported in the X540.
31:23	Rsrv	0b	Reserved



9.5 Virtual Functions Configuration Space

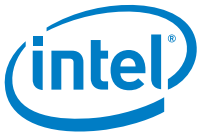
The configuration space reflected to each of the VF is a sparse version of the physical function configuration space. The following table describes the behavior of each register in the VF configuration space.

Table 9-9 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	RW	See Section 9.5.1.1.
	6	Status	Per VF	See Section 9.5.1.2.
	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 9.5.2.1.
	74	MSI-X table Address	RO — as PF	See Section 9.5.2.1.2
	78	MSI-X PBA Address	RO	See Section 9.5.2.1.3

Table 9-9 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 Table 9.5.2.2.1.
	AA	PCIe Dev Status	per VF	See Table 9.5.2.2.2.
	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 9.5.2.3.
	108	AER — Uncorr Mask	RO — 0x0	
	10C	AER — Uncorr Severity	RO — 0x0	
	110	AER — Corr Status	Per VF	
	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	0x0001000E	Next = ACS Capability.
	154	ARI — Cap/Ctrl	RO — 0X0	
ACS capability	1D0	ACS - Header	RO - 0x0001000D	Last
	1D4	ACS - Capability	RO - 0x00000000	



9.5.1 Mandatory Configuration Space

9.5.1.1 VF Command Register (0x4; RW)

Bit(s)	Initial Value	Rd/Wr	Description
0	0b	RO	IOAE: I/O Access Enable. RO as zero field.
1	0b	RO	MAE: Memory Access Enable. RO as zero field.
2	0b	RW	BME: Bus Master Enable.Disabling this bit prevents the associated VF from issuing any memory or I/O requests. Note that as MSI/MSI-X interrupt messages are in-band memory writes, disabling the bus master enable bit disables MSI/MSI-X interrupt messages as well. Requests other than memory or I/O requests are not controlled by this bit. **Note:** The state of active transactions is not specified when this bit is disabled after being enabled. The device can choose how it behaves when this condition occurs. Software cannot count on the device retaining state and resuming without loss of data when the bit is re-enabled. Transactions for a VF that has its *Bus Master Enable* set must not be blocked by transactions for VFs that have their *Bus Master Enable* cleared.
3	0b	RO	SCM: Special Cycle Enable. Hard wired to 0b
4	0b	RO	MWIE: MWI Enable. Hard wired to 0b.
5	0b	RO	PSE: Palette Snoop Enable. Hard wired to 0b.
6	0b	RO	PER: Parity Error Response. Zero for VFs.
7	0b	RO	WCE: Wait Cycle Enable. Hard wired to 0b.
8	0b	RO	SERRE: SERR# Enable. Zero for VFs.
9	0b	RO	FB2BE: Fast Back-to-Back Enable. Hard wired to 0b.
10	0b	RO	INTD: Interrupt Disable. Hard wired to 0b.
15:11	0b	RO	RSV: Reserved

9.5.1.2 VF Status Register (0x6; RW)

Bits	Initial Value	Rd/Wr	Description
2:0	0x0	RO	RSV: Reserved
3	0b	RO	IS: Interrupt Status. Hard wired to 0b.
4	1b	RO	NC: New Capabilities. Indicates that the X540 VFs implement extended capabilities. The X540 VFs implement a capabilities list, to indicate that it supports MSI-X and PCIe extensions.
5	0b	RO	66E: 66 MHz Capable. Hard wired to 0b.
6	0b	RO	RSV: Reserved
7	0b	RO	FB2BC: Fast Back-to-Back Capable. Hard wired to 0b.
8	0b	RW1C	MPERR: Data Parity Reported.
10:9	00b	RO	DEVSEL: DEVSEL Timing. Hard wired to 0b.
11	0b	RW1C	STA: Signaled Target Abort.
12	0b	RW1C	RTA: Received Target Abort.
13	0b	RW1C	RMA: Received Master Abort.
14	0b	RW1C	SSERR: Signaled System Error.
15	0b	RW1C	DSERR: Detected Parity Error.

9.5.2 PCI Capabilities

9.5.2.1 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.

9.5.2.1.1 VF MSI-X Control Register (0x72; RW)

Bits	Initial Value	Rd/Wr	Description
10:0	0x002[1]	RO	TS: Table Size.
13:11	0x0	RO	RSV: Reserved.
14	0b	RW	Mask: Function Mask.
15	0b	RW	En: MSI-X Enable.

1. Default value is read from the NVM.

9.5.2.1.2 MSI-X Table Offset Register (0x74; RW)

Bits	Default	RW	Description
31:3	0x000	RO	Table Offset. Used as an offset from the address contained in one of the function's BARs to point to the base of the MSI-X table. The lower three *Table BIR* bits are masked off (set to 0b) by software to form a 32-bit Qword-aligned offset. Note that this field is read only.
2:0	0x4	RO	Table BIR. Indicates which one of a function's BARs, beginning at 0x10 in the configuration space, is used to map the function's MSI-X table into the memory space. while BIR values: 0...5 correspond to BARs 0x10...0x 24, respectively.

9.5.2.1.3 MSI-X PBA Register (0x78; RO)

Bits	Default	Type	Description
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.
2:0	0x4	RO	PBA BIR. Indicates which one of a function's BARs, located beginning at 0x10 in configuration space, is used to map the function's MSI-X PBA into memory space. A BIR value of three indicates that the PBA is mapped in BAR 3.

9.5.2.2 PCIe Capability Registers

The device control and device status registers have some fields which are specific per VF.

9.5.2.2.1 VF Device Control Register (0xA8; RW)

Bits	Rd/Wr	Default	Description
0	RO	0b	Correctable Error Reporting Enable. Zero for VFs.
1	RO	0b	Non-Fatal Error Reporting Enable. Zero for VFs.
Bits	**Rd/Wr**	**Default**	**Description**
2	RO	0b	Fatal Error Reporting Enable. Zero for VFs.
3	RO	0b	Unsupported Request Reporting Enable. Zero for VFs.
4	RO	0b	Enable Relaxed Ordering. Zero for VFs.
7:5	RO	0b	Max Payload Size. Zero for VFs.
8	RO	0b	Not implemented in the X540.
9	RO	0b	Not implemented in the X540.
10	RO	0b	Auxiliary Power PM Enable. Zero for VFs.
11	RO	0b	Enable No Snoop. Zero for VFs.
14:12	RO	000b	Max Read Request Size. Zero for VFs.
15	RW	0b	Initiate Function Level Reset. Specific to each VF.

9.5.2.2.2 VF Device Status Register (0xAA; RW1C)

Bits	Rd/Wr	Default	Description
0	RW1C	0b	Correctable Detected. Indicates status of correctable error detection. Zero for VF.
1	RW1C	0b	Non-Fatal Error Detected. Indicates status of non-fatal error detection. Zero for VF.
2	RW1C	0b	Fatal Error Detected. Indicates status of fatal error detection. Zero for VF.
3	RW1C	0b	Unsupported Request Detected. Indicates that the X540 received an unsupported request. This field is identical in all functions. The X540 can't distinguish which function caused an error. Zero for VF.
4	RO	0b	Aux Power Detected. Zero for VFs.
5	RO	0b	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	RO	0x00	Reserved

9.5.2.3 AER Registers

The following registers in the AER capability have a different behavior in a VF function.

9.5.2.3.1 Uncorrectable Error Status Register (0x104; RW1C)

Bit Location	Attribute	Default Value	Description
3:0	RO	0x0	Reserved
4	RO	0b	Data Link Protocol Error Status
5	RO	0b	Surprise Down Error Status (Optional)
11:6	RO	0x0	Reserved
12	RW1C	0b	Poisoned TLP Status
13	RO	0b	Flow Control Protocol Error Status
14	RW1C	0b	Completion Timeout Status.
15	RW1C	0b	Completer Abort Status.
16	RW1C	0b	Unexpected Completion Status.
17	RO	0b	Receiver Overflow Status.
18	RO	0b	Malformed TLP Status.
Bit Location	**Attribute**	**Default Value**	**Description**
19	RO	0b	ECRC Error Status.
20	RW1C	0b	Unsupported Request Error Status — when caused by a function that claims a TLP.
21	RO	0b	ACS Violation Status.
31:21	RO	0x0	Reserved

9.5.2.3.2 Correctable Error Status Register (0x110; RW1C)

The Correctable Error Status register reports error status of individual correctable error sources on a PCIe device. When an individual error status bit is set to 1b it indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit.

Bit Location	Attribute	Default Value	Description
0	RW1C	0b	Receiver Error Status.
5:1	RO	0x0	Reserved

6	RW1C	0b	Bad TLP Status.
7	RW1C	0b	Bad DLLP Status.
8	RW1C	0b	REPLAY_NUM Rollover Status.
11:9	RO	0x0	Reserved
12	RW1C	0b	Replay Timer Timeout Status.
13	RW1C	0b	Advisory Non-Fatal Error Status.
31:14	RO	0b	Reserved

10.0 PHY Registers

10.1 Introduction

This chapter describes the PHY registers in the X540, which are accessible via the internal MDIO interface. Refer to Section 3.6.2.

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.
- MMD #3 contains the PCS, which handles the 10GBASE-T transmission frame coding and decoding, including the 128-DSQ and LDPC work.
- MMD #7 contains the auto negotiation function.
- MMD #29 contains the controls for the GbE and 100M PCS machinery.
- MMD# 30 contains the global control functionality for the PHY, including the embedded microcontroller.

10.1.1 PHY Register Structure

A map of these regions is shown in Table 10-1.

Table 10-1 PHY Register Map

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

5-bit Device Address (Hex)	MMD Name
1D	Clause 22 Extension
1E	Global
1F	Reserved

Any attempt to read from the reserved MMD addresses will return a value of 0x00, and any writes to these addresses will 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 PHY Register Format and Nomenclature

Abbreviation	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.
R/W	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.

Abbreviation	Type	Description
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 and Overall MMD Control
C400	Tx and Overall MMD Provisioning
C800	Tx and Overall MMD State
CC00	Tx and Overall MMD Alarms
D000	Standard Interrupt Mask
D400	Tx and Overall MMD Interrupt Mask
D800	Tx and Overall MMD Debug
DC00	Reserved
E000	Rx Control
E400	Rx Provisioning
E800	Rx State
EC00	Rx Alarms
F000	Standard Interrupt Mask

Base Offset (Hex)	Description
F400	Rx Interrupt Mask
F800	Rx Debug
FC00	Global Interrupt Flags

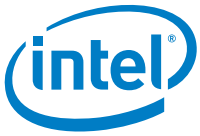
The table is split into a transmit portion, and a receive portion, with the transmit portion also containing any overall Intel specific registers for the MMD. In this table, the following definitions apply:

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 the X540 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
F:E	Reserved			RW - Reserved, do not modify.
D	Speed Selection LSB	R/W PD	1b	{D, 6}: 11b = Speed set by Bits [5:2]. 10b = 1000 Mb/s. 01b = 100 Mb/s. 00b = Not supported.
C	Reserved			RW - Reserved, do not modify.
B	Low Power	R/W PD	0b	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. 1b = Low-power mode. 0b = Normal operation.
A:7	Reserved			RW - Reserved, do not modify.
6	Speed Selection MSB	R/W PD	1b	{D, 6}: 11b = Speed set by Bits [5:2]. 10b = 1000 Mb/s. 01b = 100 Mb/s. 00b = Not supported.
5:2	10G Speed Selection [3:0]	ROS	0x0	1xxxb = Reserved. x1xxb = Reserved. xx1xb = Reserved. xxx1b = Reserved. 0000b = 10 GbE.
1	Reserved			RW - Reserved, do not modify.
0	Loopback	R/W PD	0b	This enables the PMA analog system loopback. 1b = Enable loopback mode. 0b = Normal operation.

10.2.2 PMA Standard Status 1: Address 1.1

Bit	Name	Type	Default	Description
F:8	Reserved			RW - Reserved, do not modify.
7	Fault	RO		This is the top-level fault indicator flag for the PMA (XAUI) block. This bit is set if either of the two bits 1.8.B or 1.8.A are set. 1b = Fault condition detected. 0b = No fault detected.
6:3	Reserved			RW - Reserved, do not modify.
2	PMA Receive Link Status	LL		Status of the PMA receive link. This indicates the status of the PMA receive link. This is the latched version of 1.E800.0. Note that 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. 1b = Link up. 0b = Link lost since last read.
1	Low Power Ability	RO S	1b	Indicates whether the XAUI interface supports a low-power mode. 1b = PMA supports low-power mode. 0b = no low-power mode supported.
0	Reserved			RW - 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		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		Bits 15 - 0 of the device ID.

10.2.5 PMA Standard Speed Ability: Address 1.4

Bit	Name	Type	Default	Description
F:7	Reserved			RW - Reserved, do not modify.
6	PMA 10 Mb/s Capable	ROS	0b	This is always set to 0b in the X540. 1b = PMA is 10 Mb/s capable. 0b = PMA is not 10 Mb/s capable.
5	PMA 100 Mb/s Capable	ROS	1b	This is always set to 1b in the X540. 1b = PMA is 100 Mb/s capable. 0b = PMA is not 100 Mb/s capable.
4	PMA 1 GbE Capable	ROS	1b	This is always set to 1b in the X540. 1b = PMA is 1 GbE capable. 0b = PMA is not 1 GbE capable.
3:1	Reserved			RW - Reserved, do not modify.
0	PMA 10 GbE Capable	ROS	1b	This is always set to 1b in the X540. 1b = PMA is 10 GbE capable. 0b = PMA is not 10 GbE capable.

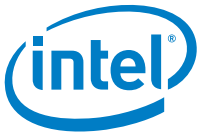
10.2.6 PMA Standard Devices in Package 1: Address 1.5

Bit	Name	Type	Default	Description
F:8	Reserved			RW - Reserved, do not modify.
7	Auto-Negotiation Present	ROS	1b	This is always set to 1b because there is Auto-negotiation in the X540. 1b = Auto-negotiation is present in the package. 0b = Auto-negotiation is not present in the package.
6	TC Present	ROS	0b	This is always set to 0b because there is no TC functionality in the X540. 1b = TC is present in the package. 0b = TC is not present in the package.
5	DTE XS Present	ROS	0b	This is always set to 0b because there is no DTE XAUI interface in the X540. 1b = DTE XS is present in the package. 0b = DTE XS is not present in the package.

Bit	Name	Type	Default	Description
4	PHY XS Present	ROS	0b	This is always set to 0b because there is no PHY XS interface in the X540. 1b = PHY XS is present in the package. 0b = PHY XS is not present in the package.
3	PCS Present	ROS	1b	This is always set to 1b because there is PCS functionality in the X540. 1b = PCS is present in the package. 0b = PCS is not present in the package.
2	WIS Present	ROS	0b	This is always set to 0b because there is no WIS functionality in the X540. 1b = WIS is present in the package. 0b = WIS is not present in the package.
1	PMA Present	ROS	1b	This is always set to 1b because there is PMA functionality in the X540. 1b = PMA is present in the package. 0b = PMA is not present.
0	Clause 22 Registers Present	ROS	0b	This is always set to 0b because there are no Clause 22 registers in the X540. 1b = Clause 22 registers are present in the package. 0b = Clause 22 registers are not present in the package.

10.2.7 PMA Standard Devices in Package 2: Address 1.6

Bit	Name	Type	Default	Description
F	Vendor Specific Device #2 Present	ROS	1b	This is always set to 1b because the X540 uses this device for the DSP PMA registers. 1b = Device #2 is present in the package. 0b = Device #2 is not present in the package.
E	Vendor Specific Device #1 Present	ROS	1b	This is always set to 1b because the X540 uses this device for the Global registers. 1b = Device #1 is present in the package. 0b = Device #1 is not present in the package.
D	Clause 22 Extension Present	ROS	1b	This is always set to 1b because the X540 uses this device for the GbE registers. 1b = Clause 22 extension is present in the package. 0b = Clause 22 extension is not present in the package.
C:0	Reserved			RW - Reserved, do not modify.



10.2.8 PMA Standard Control 2: Address 1.7

Bit	Name	Type	Default	Description
F:4	Reserved			RW - Reserved, do not modify.
3:0	PMA Device Type [3:0]	ROS	0x9	This is always set to 0x9 because the X540 is a 10GBASE-T device. 1111b = 10BASE-T PMA/PMD type. 1110b = 100BASE-TX PMA/PMD type. 1101b = 1000BASE-KX PMA/PMD type. 1100b = 1000BASE-T PMA/PMD type. 1011b = 10GBASE-KR PMA/PMD type. 1010b = 10GBASE-KX4 PMA/PMD type. 1001b = 10GBASE-T PMA type. 1000b = 10GBASE-LRM PMA/PMD type. 0111b = 10GBASE-SR PMA/PMD type. 0110b = 10GBASE-LR PMA/PMD type. 0101b = 10GBASE-ER PMA/PMD type. 0100b = 10GBASE-LX4 PMA/PMD type. 0011b = 10GBASE-SW PMA/PMD type 0010b = 10GBASE-LW PMA/PMD type. 0001b = 10GBASE-EW PMA/PMD type. 0000b = 10GBASE-CX4 PMA/PMD type.

10.2.9 PMA Standard Status 2: Address 1.8

Bit	Name	Type	Default	Description
F:E	Device Present [1:0]	ROS	0x2	This field is always set to 0x2 because the PMA is present in the X540. [F:E]: 0x3 = No device at this address. 0x2 = Device present at this address. 0x1 = No device at this address. 0x0 = No device at this address.
D	Transmit Fault Location Ability	ROS	1b	This bit indicates whether the PMA has the ability to locate faults along the transmit path. 1b = PMA has the capability to detect a fault condition on the transmit path. 0b = PMA does not have the capability to detect a fault condition on the transmit path.
C	Receive Fault Location Ability	ROS	1b	This bit indicates whether the PMA has the ability to locate faults along the receive path. 1b = PMA has the capability to detect a fault condition on the receive path. 0b = PMA does not have the capability to detect a fault condition on the receive path.

Bit	Name	Type	Default	Description
B	Transmit Fault	LH		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. 1b = Fault condition on transmit path. 0b = No fault condition on transmit path.
A	Receive Fault	LH		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. 1b = Fault condition on receive path. 0b = No fault condition on receive path.
9	Extended Abilities	RO S	1b	This field is set to 1b because the PMA in the X540 has extended abilities. 1b = PMA has extended abilities listed in register 1.11. 0b = PMA does not have extended abilities.
8	PMD Transmit Disable Ability	RO S	1b	This bit indicates whether the PMD has the capability of disabling its transmitter. This field is always set to 0x1 because the PMD in the X540 has this ability. 1b = PMD has the capability of disabling the transmitter. 0b = PMD does not have the capability of disabling the transmitter.
7	PMA 10GBASE-SR Capable	RO S	0b	This field is always set to 0b because the PMA in the X540 only supports 10GBASE-T. 1b = PMA supports 10GBASE-SR. 0b = PMA does not support 10GBASE-SR.
6	PMA 10GBASE-LR Capable	RO S	0b	This field is always set to 0b because the PMA in the X540 only supports 10GBASE-T. 1b = PMA supports 10GBASE-LR. 0b = PMA does not support 10GBASE-LR.
5	PMA 10GBASE-ER Capable	RO S	0b	This field is always set to 0b because the PMA in the X540 only supports 10GBASE-T. 1b = PMA supports 10GBASE-ER. 0b = PMA does not support 10GBASE-ER.
4	PMA 10GBASE-LX4 Capable	RO S	0b	This field is always set to 0b because as the PMA in the X540 only supports 10GBASE-T. 1b = PMA supports 10GBASE-LX4. 0b = PMA does not support 10GBASE-LX4.
3	PMA 10GBASE-SW Capable	RO S	0b	This field is always set to 0b because the PMA in the X540 only supports 10GBASE-T. 1b = PMA supports 10GBASE-SW. 0b = PMA does not support 10GBASE-SW.
2	PMA 10GBASE-LW Capable	RO S	0b	This field is always set to 0b because the PMA in the X540 only supports 10GBASE-T. 1b = PMA supports 10GBASE-LW. 0b = PMA does not support 10GBASE-LW.

Bit	Name	Type	Default	Description
1	PMA 10GBASE-EW Capable	RO S	0b	This field is always set to 0b because the PMA in the X540 only supports 10GBASE-T. 1b = PMA supports 10GBASE-EW. 0b = PMA does not support 10GBASE-EW.
0	PMA Loopback Ability	RO S	1b	This is always set to 1b because the PMA in the X540 supports loopback. 1b = PMA supports loopback. 0b = PMA does not support loopback.

10.2.10 PMD Standard Transmit Disable Control: Address 1.9

Bit	Name	Type	Default	Description
F:5	Reserved			RW - Reserved, do not modify.
4	PMD Channel 3 Transmit Disable	R/W PD	0b	When disabled, the average launch power on a pair is set to less than -53 dBm. 1b = Disable output on transmit channel 3. 0b = Normal operation.
3	PMD Channel 2 Transmit Disable	R/W PD	0b	When disabled, the average launch power on a pair is set to less than -53 dBm. 1b = Disable output on transmit channel 2. 0b = Normal operation.
2	PMD Channel 1 Transmit Disable	R/W PD	0b	When disabled, the average launch power on a pair is set to less than -53 dBm. 1b = Disable output on transmit channel 1. 0b = Normal operation.
1	PMD Channel 0 Transmit Disable	R/W PD	0b	When disabled, the average launch power on a pair is set to less than -53 dBm. 1b = Disable output on transmit channel 0. 0b = Normal operation.
0	PMD Global Transmit Disable	R/W PD	0b	When set, this bit disables (and overrides) all four channels, and sets the average launch power on all pairs to less than -53 dBm. 1b = Disable output on all channels. 0b = Normal operation.

10.2.11 PMD Standard Signal Detect: Address 1.A

Bit	Name	Type	Default	Description
F:5	Reserved			RW - Reserved, do not modify.
4	PMD Channel 3 Signal Detect	RO		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. 1b = Signal detected on receive channel 3. 0b = No signal detected.
3	PMD Channel 2 Signal Detect	RO		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. 1b = Signal detected on receive channel 2. 0b = No signal detected.
2	PMD Channel 1 Signal Detect	RO		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. 1b = Signal detected on receive channel 1. 0b = No signal detected.
1	PMD Channel 0 Signal Detect	RO		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. 1b = Signal detected on receive channel 0. 0b = No signal detected.
0	PMD Global Signal Detect	RO		This bit is marked when all required, valid Ethernet signals to create a connection are present on the line. 1b = Signals detected on all required channels. 0b = No signal detected.

10.2.12 PMD Standard 10G Extended Ability Register: Address 1.B

Bit	Name	Type	Default	Description
F:9	Reserved			RW - Reserved, do not modify.
8	PMA 10BASE-T Capable	ROS	0b	This field is always set to 0b because the PMA in the the X540 does not support 10BASE-TX. 1b = PMA capable of 10BASE-T. 0b = PMA incapable of 10BASE-T.
7	PMA 100BASE-TX Capable	ROS	1b	This field is always set to 1b because the PMA in the X540 supports 100BASE-TX. 1b = PMA capable of 100BASE-TX. 0b = PMA incapable of 100BASE-TX.

Bit	Name	Type	Default	Description
6	Reserved			RW - Reserved, do not modify.
5	PMA 1000BASE-T Capable	ROS	1b	This field is set to 1b because the PMA in the X540 supports 1000BASE-T. 1b = PMA capable of 1000BASE-T. 0b = PMA incapable of 1000BASE-T.
4:3	Reserved			RW - Reserved, do not modify
2	PMA 10GBASE-T Capable	ROS	1b	This field is set to 1b because the PMA in the X540 supports 10GBASE-T 1b = PMA capable of 10GBASE-T. 0b = PMA incapable of 10GBASE-T.
1	Reserved			RW - Reserved, do not modify.
0	PMA 10GBASE-CX4 Capable	ROS	0b	This field is always set to 0b because the PMA does not support 10GBASE-CX4. 1b = PMA capable of 10GBASE-CX4. 0b = PMA incapable of 10GBASE-CX4.

10.2.13 PMA Standard Package Identifier 1: Address 1.E

Bit	Name	Type	Default	Description
F:0	Package ID MSW [1F:10]	RO		Bits 31 - 16 of the package ID.

10.2.14 PMA Standard Package Identifier 2: Address 1.F

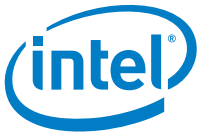
Bit	Name	Type	Default	Description
F:0	Package ID LSW [F:0]	RO		Bits 15 - 0 of the package ID.

10.2.15 PMA 10GBASE-T Status: Address 1.81

Bit	Name	Type	Default	Description
F:1	Reserved			RW - Reserved, do not modify.
0	Link Partner Information Valid	RO		When set, this bit indicates that the start-up protocol has completed. 1b = 10GBASE-T link partner information is valid. 0b = 10GBASE-T link partner information is not valid.

10.2.16 PMA 10GBASE-T Pair Swap and Polarity Status: Address 1.82

Bit	Name	Type	Default	Description
F:C	Reserved			RW - Reserved, do not modify.
B:8	Pair Polarity [3:0]	RO		When set, this bit indicates that the wires on the respective pair are reversed. 1b = Polarity of pair is reversed. 0b = Polarity of pair is normal. [3] = Pair D polarity. [2] = Pair C polarity. [1] = Pair B polarity. [0] = Pair A polarity.
7:2	Reserved			RW - Reserved, do not modify.
1:0	MDI / MD-X Connection State [1:0]	RO		These two bits indicate the current status of pair swaps at the MDI / MD-X [1:0]: 11b = No crossover. 10b = Pair A / B crossover. 01b = Pair C / D crossover. 00b = Pair A / B and C / D crossover.



10.2.17 PMA 10GBASE-T Tx Power Backoff Setting: Address 1.83

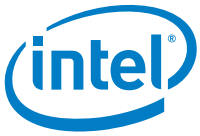
Bit	Name	Type	Default	Description
F:D	Link Partner Tx Power Backoff [2:0]	RO		The power backoff of the link partner. [F:D]: 0x7 = 14 dB. 0x6 = 12 dB. 0x5 = 10 dB. 0x4 = 8 dB. 0x3 = 6 dB. 0x2 = 4 dB. 0x1 = 2 dB. 0x0 = 0 dB.
C:A	Tx Power Backoff [2:0]	RO		The power backoff of the PMA. [C:A]: 0x7 = 14 dB. 0x6 = 12 dB. 0x5 = 10 dB. 0x4 = 8 dB. 0x3 = 6 dB. 0x2 = 4 dB. 0x1 = 2 dB. 0x0 = 0 dB.
9:1	Reserved			RW - Reserved, do not modify.
0	Reserved		0b	RW - Reserved, do not modify.

10.2.18 PMA 10GBASE-T Test Modes: Address 1.84

Bit	Name	Type	Default	Description
F:D	Test Mode Control [2:0]	R/W PD	0x0	Test mode control for the PMA as defined in Section 55.5.2 of 802.3an. [F:D]: 0x7 = Pseudo random test mode for BER Monitor. 0x6 = Transmitter droop test. 0x5 = PSD and power level test. 0x4 = Transmitter distortion test. 0x3 = Slave mode jitter test. 0x2 = Master mode jitter test. 0x1 = Master source for slave mode jitter test. 0x0 = Normal operation.
C:A	Transmitter Test Frequencies [2:0]	R/W PD	0x0	The test frequencies associated with test mode #4 in [F:D]. [C:A]: 0x7 = Reserved. 0x6 = Dual tone #5. 0x5 = Dual tone #4. 0x4 = Dual tone #3. 0x3 = Reserved. 0x2 = Dual tone #2. 0x1 = Dual tone #1. 0x0 = Reserved.
9:0	Reserved			RW - Reserved, do not modify.

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		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		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		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		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

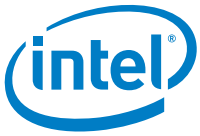
Bit	Name	Type	Default	Description
F:0	Channel A Minimum Operating Margin [F:0]	RO		Minimum operating margin (dB) of Channel A since the last link up. The excess SNR that is used by the channel, over and above the minimum SNR required to operate at a BER of 10^{-12}. It is reported with 0.1 dB of resolution 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		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		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		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		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		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

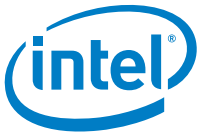
Bit	Name	Type	Default	Description
F:0	Channel C Received Signal Power [F:0]	RO		Received signal power (dBm) for Channel C. The received signal power on the channel. It is reported with 0.1 dB of resolution to an accuracy of 0.5 dB within the range of -20.0 dB to +5.5 dB. The number is in offset two's 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		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
F	Reserved			RW - 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.
7:0	Reserved			RW - Reserved, do not modify.



10.2.32 PMA 10GBASE-T Skew Delay 2: Address 1.92

Bit	Name	Type	Default	Description
F	Reserved			RW - 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.
7	Reserved			RW - Reserved, do not modify.
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.

10.2.33 PMA Receive Reserved Vendor Provisioning 1: Address 1.E400

Bit	Name	Type	Default	Description
F	External PHY Loopback	R/W PD	0b	External PHY loopback expects a loopback connector such that Pair A is connected to Pair B, and Pair C is connected to Pair D. 1b = Enable external PHY loopback. 0b = Normal operation.
E:3	Reserved Receive Provisioning 1 [B:0]	R/W PD	0x000	Reserved for future use.
2	Enable X540 Cisco Fast Retrain	R/W PD	1b	If the link is a X540 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. 1b = Enable PMA fast link retrain. 0b = Disable PMA fast link retrain.

Bit	Name	Type	Default	Description
1	Force MDI Configuration	R/W PD	0b	Normally the MDI reversal configuration is taken from the PHYn_RVSL pin, n=0,1. If the Force MDI Configuration bit is asserted, the PHYn_RVSL pin is ignored and the current provisioned value of the MDI configuration is used instead. 1b = Ignore state of PHYn_RVSL pin 0b = Set MDI Configuration based on state of PHYn_RVSL.
0	MDI Configuration	R/W PD	0b	Setting this bit determines whether MDI is reversed or not. Note that the reversal does not change pair polarity. For example, A+ maps to D+, etc. The value of this bit is set during auto-negotiation to the value of the PHYn_RVSL pin unless the *Force MDI Configuration* bit (1.E400.1) is asserted. When the *Force MDI Configuration* bit is asserted, the PHYn_RVSL pin is ignored and this bit is unchanged from its default or provisioned value. If this bit is changed manually after auto-negotiation completes, auto-negotiation must be restarted to achieve the desired MDI configuration. 1b = MDI Reversed (ABCD -> DCBA). 0b = MDI Normal (ABCD -> ABCD).

10.2.34 PMA Receive Vendor State 1: Address 1.E800

Bit	Name	Type	Default	Description
F:1	Reserved			RW - Reserved, do not modify.
0	PMA Receive Link Current Status	RO		This is the current state of 1.1.2. 1b = Rx link good.

10.2.35 PMA Receive Reserved Vendor State 1: Address 1.E810

Bit	Name	Type	Default	Description
F:0	Accumulated Fast Retrain Time[F:0]	RO		Accumulated time in milliseconds spent in fast retrain since the last auto-negotiation sequence. This is a saturating register.

10.2.36 PMA Receive Vendor State 2: Address 1.E811

Bit	Name	Type	Default	Description
F:8	Total Number Of Link Recovery Events Since Last AutoNeg [7:0]	RO		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).
7:0	Total Number Of RFI Training Link Recovery Events Since Last AutoNeg [7:0]	RO		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).

10.2.37 PMA Transmit Standard Interrupt Mask 1: Address 1.F000

Bit	Name	Type	Default	Description
F:3	Reserved			RW - Reserved, do not modify.
2	PMA Receive Link Status Mask	R/W PD	0b	Mask for Bit 1.1.2. 1b = Enable interrupt generation. 0b = Disable interrupt generation.
1:0	Reserved			RW - Reserved, do not modify.

10.2.38 PMA Transmit Standard Interrupt Mask 2: Address 1.F001

Bit	Name	Type	Default	Description
F:C	Reserved			RW - Reserved, do not modify.
B	Transmit Fault Mask	R/W PD	0b	Bit 1.8.B. 1b = Enable interrupt generation. 0b = Disable interrupt generation.

Bit	Name	Type	Default	Description
A	Receive Fault Mask	R/W PD	0b	Bit 1.8.A. 1b = Enable interrupt generation. 0b = Disable interrupt generation.
9:0	Reserved			RW - Reserved, do not modify.

10.2.39 PMA Vendor Global Interrupt Flags 1: Address 1.FC00

Bit	Name	Type	Default	Description
F:C	Reserved			RW - Reserved, do not modify.
B	Standard Alarm 1 Interrupt	RO		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.F000. 1b = Interrupt.
A	Standard Alarm 2 Interrupt	RO		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.F001. 1b = Interrupt.
9:0	Reserved			RW - Reserved, do not modify.

10.3 PCS Registers

10.3.1 PCS Standard Control 1: Address 3.0

Bit	Name	Type	Default	Description
F	Reserved			RW - Reserved, do not modify.
E	Loopback	R/W PD	0b	This enables the PCS DSQ system loopback. 1b = Enable loopback mode. 0b = Normal operation.

Bit	Name	Type	Default	Description
D	Speed Selection LSB	R/W PD	1b	{D,6}: 11b = Speed set by bits [5:2]. 10b = 1000 Mb/s. 01b = 100 Mb/s. 00b = Not supported.
C	Reserved			RW - Reserved, do not modify.
B	Low Power	R/W PD	0b	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. 1b = Low-power mode. 0b = Normal operation.
A:7	Reserved			RW - Reserved, do not modify.
6	Speed Selection MSB	R/W PD	1b	{D,6}: 11b = Speed set by Bits [5:2]. 10b = 1000 Mb/s. 01b = 100 Mb/s. 00b = Not supported.
5:2	10 GbE Speed Selection [3:0]	ROS	0x0	1xxxb = Reserved. x1xxb = Reserved. xx1xb = Reserved. xxx1b = Reserved. 0000b = 10 GbE.
1:0	Reserved			RW - Reserved, do not modify.

10.3.2 PCS Standard Status 1: Address 3.1

Bit	Name	Type	Default	Description
F:8	Reserved			RW - Reserved, do not modify.
7	Fault	RO		This is the top-level fault indicator flag for the PCS (XAUI) block. This bit is set if either of the two bits 3.8.B or 3.8.A are set. 1b = Fault condition detected. 0b = No fault detected.
6:3	Reserved			RW - Reserved, do not modify.
2	PCS Receive Link Status	LL		This indicates the status of the PCS receive link. This is a latching low version of bit 3.20.C (see PCS 10GBASE-T Status 1: Address 3.20). Status of the PCS receive link. 1b = Link up. 0b = Link lost since last read.

Bit	Name	Type	Default	Description
1	Low Power Ability	ROS	1b	Indicates whether the XAUI interface supports a low-power mode. 1b = PCS supports low-power mode. 0b = no low-power mode supported.
0	Reserved			RW - Reserved, do not modify.

10.3.3 PCS Standard Device Identifier 1: Address 3.2

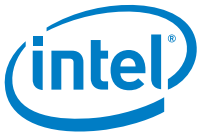
Bit	Name	Type	Default	Description
F:0	Device ID MSW [1F:10]	RO		Bits 31 - 16 of the device ID.

10.3.4 PCS Standard Device Identifier 2: Address 3.3

Bit	Name	Type	Default	Description
F:0	Device ID LSW [F:0]	RO		Bits 15 - 0 of the device ID.

10.3.5 PCS Standard Speed Ability: Address 3.4

Bit	Name	Type	Default	Description
F:1	Reserved			RW - Reserved, do not modify.
0	10 GbE Capable	ROS	1b	This is always set to 1b in the X540. 1b = PCS is 10 GbE capable. 0b = PCS is not 10 GbE capable.



10.3.6 PCS Standard Devices in Package 1: Address 3.5

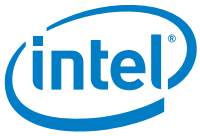
Bit	Name	Type	Default	Description
F:8	Reserved			RW - Reserved, do not modify.
7	Auto-Negotiation Present	RO S	1b	This is always set to 1b because there is auto-negotiation in the X540. 1b = Auto-negotiation is present in the package. 0b = Auto-negotiation is not present in the package.
6	TC Present	RO S	0b	This is always set to 0b because here is no TC functionality in the X540. 1b = TC is present in the package. 0b = TC is not present in the package.
5	DTE XS Present	RO S	0b	This is always set to 0b because there is no DTE XAUI interface in the X540. 1b = DTE XS is present in the package. 0b = DTE XS is not present in package.
4	PHY XS Present	RO S	0b	This is always set to 0b because there is no PHY XS interface in the X540. 1b = PHY XS is present in the package. 0b = PHY XS is not present in the package.
3	PCS Present	RO S	1b	This is always set to 1b because there is PCS functionality in the X540. 1b = PCS is present in the package. 0b = PCS is not present in the package.
2	WIS Present	RO S	0b	This is always set to 0b because there is no WIS functionality in the X540. 1b = WIS is present in the package. 0b = WIS is not present in the package.
1	PMA Present	RO S	1b	This is always set to 1b because there is PMA functionality in the X540. 1b = PMA is present in the package. 0b = PMA is not present.
0	Clause 22 Registers Present	RO S	0b	This is always set to 0b because there are no Clause 22 registers in the X540. 1b = Clause 22 registers are present in the package. 0b = Clause 22 registers are not present in the package.

10.3.7 PCS Standard Devices in Package 2: Address 3.6

Bit	Name	Type	Default	Description
F	Vendor Specific Device #2 Present	ROS	1b	This is always set to 1b because the X540 uses this device for the DSP PMA registers. 1b = Device #2 is present in the package. 0b = Device #2 is not present in the package.
E	Vendor Specific Device #1 Present	ROS	1b	This is always set to 1b because the X540 uses this device for the global control registers. 1b = Device #1 is present in the package. 0b = Device #1 is not present in the package.
D	Clause 22 Extension Present	ROS	1b	This is always set to 1b because the X540 uses this device for the GbE registers. 1b = Clause 22 Extension is present in the package. 0b = Clause 22 Extension is not present in the package.
C:0	Reserved			RW - Reserved, do not modify.

10.3.8 PCS Standard Control 2: Address 3.7

Bit	Name	Type	Default	Description
F:2	Reserved			RW - Reserved, do not modify.
1:0	PCS Device Type [1:0]	ROS	0x3	This is always set to 0x3 because the X540 only supports 10GBASE-T operation. [1:0]: 0x3 = 10GBASE-T. 0x2 = 10GBASE-W. 0x1 = 10GBASE-X. 0x0 = 10GBASE-R.



10.3.9 PCS Standard Status 2: Address 3.8

Bit	Name	Type	Default	Description
F:E	Device Present [1:0]	ROS	0x2	This field is always set to 0x2 because the PCS registers reside here in the X540. [F:E]: 0x3 = No device at this address. 0x2 = Device present at this address. 0x1 = No device at this address. 0x0 = No device at this address.
D:C	Reserved			RW - Reserved, do not modify.
B	Transmit Fault	LH		This bit indicates whether there is a fault somewhere along the transmit path. This bit is duplicated at 3.CC01.0. 1b = Fault condition on transmit path. 0b = No fault condition on transmit path.
A	Receive Fault	LH		This bit indicates whether there is a fault somewhere along the receive path. This bit is duplicated at 3.EC04.2. 1b = Fault condition on receive path. 0b = No fault condition on receive path.
9:4	Reserved			RW - Reserved, do not modify.
3	10GBASE-T capable	ROS	1b	This field is always set to 1b because the PCS in the X540 only supports 10GBASE-T. 1b = PCS supports 10GBASE-T PCS type. 0b = PCS does not support 10GBASE-T.
2	10GBASE-W capable	ROS	0b	This field is always set to 0b because the PCS in the X540 only supports 10GBASE-T. 1b = PCS supports 10GBASE-W PCS type. 0b = PCS does not support 10GBASE-W.
1	10GBASE-X capable	ROS	0b	This field is always set to 0b because the PCS in the X540 only supports 10GBASE-T. 1b = PCS supports 10GBASE-X PCS type. 0b = PCS does not support 10GBASE-X.
0	10GBASE-R capable	ROS	0b	This field is always set to 0b because the PCS in the X540 only supports 10GBASE-T. 1b = PCS supports 10GBASE-R PCS type. 0b = PCS does not support 10GBASE-R.

10.3.10 PCS Standard Package Identifier 1: Address 3.E

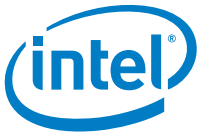
Bit	Name	Type	Default	Description
F:0	Package ID MSW [1F:10]	RO		Bits 31 - 16 of the package ID.

10.3.11 PCS Standard Package Identifier 2: Address 3.F

Bit	Name	Type	Default	Description
F:0	Package ID LSW [F:0]	RO		Bits 15 - 0 of the package ID.

10.3.12 PCS 10GBASE-T Status 1: Address 3.20

Bit	Name	Type	Default	Description
F:D	Reserved			RW - Reserved, do not modify.
C	10GBASE-T Receive Link Status	RO		When set, this bit indicates that the 10GBASE-T receive link is functioning properly. This is a non-latching version of bit 3.1.2 (see PCS Standard Status 1: Address 3.1). The receive link is up when *Block Lock* status is asserted and *High BER* is deasserted.) 1b = 10GBASE-T receive link up. 0b = 10GBASE-T receive link down.
B:2	Reserved			RW - Reserved, do not modify.
1	10GBASE-T High BER	RO		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. 1b = PCS is reporting a BER $\geq 10^{-4}$. 0b = PCS is reporting a BER $< 10^{-4}$.
0	10GBASE-T PCS Block Lock	RO		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. 1b = 10GBASE-T PCS framer is locked. 0b = 10GBASE-T PCS framer is not locked.



10.3.13 PCS 10GBASE-T Status 2: Address 3.21

Bit	Name	Type	Default	Description
F	10GBASE-T PCS Block Lock Latched	LL		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 PCS 10GBASE-T Status 1: Address 3.20). 1b = 10GBASE-T PCS framer is locked. 0b = 10GBASE-T PCS framer is not locked.
E	10GBASE-T High BER Latched	LH		When set, this bit indicates a high BER is being seen at the PCS. This is the interrupt for bit 3.20.1 (see PCS 10GBASE-T Status 1: Address 3.20”). 1b = PCS is reporting a BER $\geq 10^{-4}$. 0b = PCS is reporting a BER $< 10^{-4}$.
D:8	LDPC Errored Frame Counter [5:0]	SCT	0x00	Clear on read. This is taken from the state machine in Figure 55.14 in the 10GBASE-T specification. A saturating count of the number of times a bad LDPC frame is received.
7:0	Errored 65B Block Counter [7:0]	SCT	0x00	Clear on read. This is taken from the state machine in Figure 55.16 in the 10GBASE-T specification. A saturating count of the number of times a bad 65-byte block is received.

10.3.14 PCS Transmit Vendor Provisioning 1: Address 3.C400

Bit	Name	Type	Default	Description
F:1	Reserved			RW - Reserved, do not modify
0	PCS Tx Auxiliary Bit Value	R/W PD	0b	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.

10.3.15 PCS Transmit Reserved Vendor Provisioning 1: Address 3.C410

Bit	Name	Type	Default	Description
F:1	Reserved Transmit Provisioning 1 [F:1]	R/W PD	0x0000	Reserved for future use.
0	PCS IEEE Loopback Passthrough Disable	R/W PD	0b	When set, this bit disables the output of the PHY when IEEE loopback is set. 1b = Disable data pass through on IEEE loopback.

10.3.16 PCS Receive Vendor State 1: Address 3.E800

Bit	Name	Type	Default	Description
F:1	Reserved			RW - Reserved, do not modify.
0	PCS Rx Current Value of Auxiliary Bit	RO		The current value of the PCS Rx auxiliary bit. This value has a maskable interrupt associated with it in 3.EC00.0.

10.3.17 PCS Receive Vendor Alarms 1: Address 3.EC00

Bit	Name	Type	Default	Description
F	CRC Error	LH		This bit is set when a CRC-8 error is detected on the receive PCS frame. 1b = Rx CRC frame error.
E	LDPC Decode Failure	LH		This bit is set when the LDPC decoder fails to decode an LDPC block. 1b = LDPC decode failure.
D:9	Reserved			RW - Reserved, do not modify.
8	Invalid 65B Block	LH		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.

Bit	Name	Type	Default	Description
7:1	Reserved			RW - Reserved, do not modify.
0	Change in Auxiliary Bit	LRF		This bit is set when a change is detected in the auxiliary bit. 1b = Indicates a change in the value of the auxiliary bit.

10.3.18 PCS Receive Standard Interrupt Mask 1: Address 3.F000

Bit	Name	Type	Default	Description
F:3	Reserved			RW - Reserved, do not modify.
2	PCS Receive Link Status Mask	R/W PD	0b	Mask for bit 3.1.2. Note this bit also appears as bit 3.20.C, but only as a status bit. 1b = Enable interrupt generation. 0b = Disable interrupt generation.
1:0	Reserved			RW - Reserved, do not modify.

10.3.19 PCS Receive Standard Interrupt Mask 2: Address 3.F001

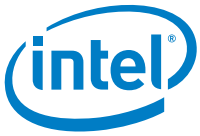
Bit	Name	Type	Default	Description
F:C	Reserved			RW - Reserved, do not modify.
B	Transmit Fault Mask	R/W PD	0b	Bit 3.8.B. 1b = Enable interrupt generation. 0b = Disable interrupt generation.
A	Receive Fault Mask	R/W PD	0b	Bit 3.8.A. 1b = Enable interrupt generation. 0b = Disable interrupt generation.
9:0	Reserved			RW - Reserved, do not modify.

10.3.20 PCS Receive Standard Interrupt Mask 3: Address 3.F002

Bit	Name	Type	Default	Description
F	10GBASE-T PCS Block Lock Latched Mask	R/W PD	0b	**CAUTION:** There is a gap in this register set, which causes a dummy word to be inserted in the C structure. 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. 1b = Enable interrupt generation. 0b = Disable interrupt generation.
E	10GBASE-T High BER Latched Mask	R/W PD	0b	When set, this bit indicates a high BER is seen at the PCS. This is the interrupt for bit 3.21.E. 1b = Enable interrupt generation. 0b = Disable interrupt generation.
D:0	Reserved			RW - Reserved, do not modify.

10.3.21 PCS Receive Vendor Interrupt Mask 1: Address 3.F400

Bit	Name	Type	Default	Description
F	CRC Error Mask	R/W PD	0b	This bit is set when a CRC-8 error is detected on the receive PCS frame. 1b = Enable interrupt generation. 0b = Disable interrupt generation.
E	LDPC Decode Failure Mask	R/W PD	0b	This bit is set when the LDPC decoder fails to decode an LDPC block. 1b = Enable interrupt generation. 0b = Disable interrupt generation.
D:9	Reserved			RW - Reserved, do not modify.
8	Invalid 65B Block Mask	R/W PD	0b	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 = Enable interrupt generation. 0b = Disable interrupt generation.
7:1	Reserved			RW - Reserved, do not modify.
0	Change in Auxiliary Bit Mask	R/W PD	0b	This bit is set when a change is detected in the auxiliary bit. 1b = Enable interrupt generation. 0b = Disable interrupt generation.



10.3.22 PCS Vendor Global Interrupt Flags 1: Address 3.FC00

Bit	Name	Type	Default	Description
F	Standard Alarm 1 Interrupt	RO		An interrupt was generated from status register PCS Standard Status 1: Address 3.1 and the corresponding mask register PCS Receive Standard Interrupt Mask 1: Address 3.F000. 1b = Interrupt in standard alarms 1.
E	Standard Alarm 2 Interrupt	RO		An interrupt was generated from status register Global Standard Status 2: Address 1E.8 and the corresponding mask register PCS Receive Standard Interrupt Mask 2: Address 3.F001. 1b = Interrupt in standard alarms 2.
D	Standard Alarm 3 Interrupt	RO		An interrupt was generated from status register PCS 10GBASE-T Status 2: Address 3.21 and the corresponding mask register PCS Receive Standard Interrupt Mask 3: Address 3.F002. 1b = Interrupt in standard alarms 3.
C:0	Reserved			RW - 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
F:E	Reserved			RW - Reserved, do not modify.
D	Extended Next Page Control	R/W PD	1b	This bit is OR'ed with bit 7.10.C. 1b = Extended next pages are enabled. 0b = Extended next pages are disabled.
C	Auto-Negotiation Enable	R/W PD	1b	When enabled, auto-negotiation determines the link speed. When disabled, the link is forced to its down state. 1b = Enable auto-negotiation. 0b = Disable auto-negotiation.
B:A	Reserved			RW - Reserved, do not modify.
9	Restart Auto-Negotiation	R/W SC	0b	1b = Restart auto-negotiation process. 0b = Normal operation.
8:0	Reserved			RW - Reserved, do not modify.

10.4.2 Auto-Negotiation Standard Status 1: Address 7.1

Bit	Name	Type	Default	Description
F:8	Reserved			RW - Reserved, do not modify.
7	Extended Next Page Status	RO		Indicates that both the local device and the link partner have indicated support for extended next page. 1b = Extended next page will be used. 0b = Extended next page will not be used.
6	Extended Next Page Received	LH		Indicates that an extended next page has been received and has been placed in registers 7.19 through 7.1B. 1b = An extended next page has been received. 0b = An extended next page has not been received.

Bit	Name	Type	Default	Description
5	Auto-Negotiation Complete	RO		This indicates the status of the auto-negotiation receive link. 1b = Auto-negotiation complete. 0b = Auto-negotiation in process.
4	Remote Fault	LH		This indicates that the remote PHY has a fault. 1b = Remote fault condition detected. 0b = No remote fault condition detected.
3	Auto-Negotiation Ability	ROS	1	Always set as 1b because the local device has auto-negotiation ability. 1b = PHY is able to perform auto-negotiation. 0b = PHY is not able to perform auto-negotiation.
2	Link Status	LL		This bit is a duplicate of the *PMA Receive Link Status* bit (bit 2) of the PMA Standard Status 1: Address 1.1 register. Note that 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. 1b = Link is up. 0b = Link lost since last read.
1	Reserved			RW - Reserved, do not modify.
0	Link Partner Auto-Negotiation Ability	RO		1b = Link partner able to perform auto-negotiation. 0b = Link partner not able to perform auto-negotiation.

10.4.3 Auto-Negotiation Standard Device Identifier 1: Address 7.2

Bit	Name	Type	Default	Description
F:0	Device ID MSW [1F:10]	RO		Bits 31:16 of the device ID.

10.4.4 Auto-Negotiation Standard Device Identifier 2: Address 7.3

Bit	Name	Type	Default	Description
F:0	Device ID LSW [F:0]	RO	0x0	Bits 15:0 of device ID.
9:4	Manufacturer's Model Number	RO	0x20	6 bits containing the manufacturer's part number
3:0	Manufacturer's Revision Number	RO		4 bits containing the manufacturer's revision number. 0x5: B0 step

10.4.5 Auto-Negotiation Standard Devices in Package 1: Address 7.5

Bit	Name	Type	Default	Description
F:8	Reserved			RW - Reserved, do not modify.
7	Auto-Negotiation Present	ROS	1b	This bit is always set to 1b because there is auto-negotiation in the X540. 1b = Auto-negotiation is present in package. 0b = Auto-negotiation is not present in package.
6	TC Present	ROS	0b	This bit is always set to 0b because there is no TC functionality in the X540. 1b = TC is present in package. 0b = TC is not present in package.
5	DTE XS Present	ROS	0b	This bit is always set to 0b because there is no DTE XAUI interface in the X540. 1b = DTE XS is present in package. 0b = DTE XS is not present in package.
4	PHY XS Present	ROS	0b	This bit is always set to 0b because there is no PHY XS interface in the X540. 1b = PHY XS is present in package. 0b = PHY XS is not present in package.
3	PCS Present	ROS	1b	This bit is always set to 1b because there is PCS functionality in the X540. 1b = PCS is present in package. 0b = PCS is not present in package.

Bit	Name	Type	Default	Description
2	WIS Present	ROS	0b	This bit is always set to 0b because there is no WIS functionality in the X540. 1b = WIS is present in package. 0b = WIS is not present in package.
1	PMA Present	ROS	1b	This bit is always set to 1b because there is PMA functionality in the X540. 1b = PMA is present in package. 0b = PMA is not present.
0	Clause 22 Registers Present	ROS	0b	This bit is always set to 0b because there are no Clause 22 registers in the X540. 1b = Clause 22 registers are present in package. 0b = Clause 22 registers are not present in package.

10.4.6 Auto-Negotiation Standard Devices in Package 2: Address 7.6

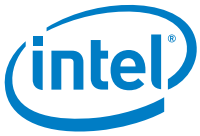
Bit	Name	Type	Default	Description
F	Vendor Specific Device #2 Present	ROS	1b	This is always set to 1b because the X540 uses this device for the DSP PMA registers. 1b = Device #2 is present in package. 0b = Device #2 is not present in package.
E	Vendor Specific Device #1 Present	ROS	1b	This is always set to 1b because the X540 uses this device for the global control registers. 1b = Device #1 is present in package. 0b = Device #1 is not present in package.
D	Clause 22 Extension Present	ROS	1b	This is always set to 1b because the X540 uses this device for the GbE registers. 1b = Clause 22 Extension is present in package. 0b = Clause 22 Extension is not present in package.
C:0	Reserved			RW - Reserved, do not modify.

10.4.7 Auto-Negotiation Standard Status 2: Address 7.8

Bit	Name	Type	Default	Description
F:E	Device Present [1:0]	ROS	0x2	This field is always set to 0x2 because auto-negotiation resides in the X540. [F:E]: 0x3 = No device at this address. 0x2 = Device present at this address. 0x1b = No device at this address. 0x0b = No device at this address.
D:0	Reserved			RW - Reserved, do not modify.

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		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		Bits 15- 0 of the package ID.

10.4.10 Auto-Negotiation Advertisement Register: Address 7.10[1]

Bit	Name	Type	Default	Description
F	Next Page Ability	R/W PD	1b	If the X540 implements next page ability and needs to engage in a next page exchange, it must set the *Next Page (NP)* bit to 1b. the X540 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.
E	Reserved			RW - Reserved, do not modify.
D	Advertisement Remote Fault	R/W	0b	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 shall be maintained as 0b.

1. This register is used in conjunction with bit 7.C400.5 to hard-provision the auto-negotiation base page that is sent.

C	Extended Next Page Ability	R/W PD	1b	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. 1b = Extended next page capable. 0b = Not capable of extended next pages.
B:5	Technology Ability Field [6:0]	R/W PD	0x08	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. **Note:** Technology NOT supported by the X540, bits must be set to 0x00. Bit 0 = 10BASE-T. Bit 1 = 10BASE-T full duplex. 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.
4:0	Selector Field [4:0]	R/W PD	0x01	This field should always be set to 0x01 because the PHY is only capable of handling 802.3 Ethernet. This defines the device compatibility: 0x00b = Reserved. 0x01b = IEEE 802.3. 0x02 = IEEE 802.9 ISLAN-16T. 0x03 = IEEE 802.5. 0x04 = IEEE 1394. 0x05:0x1F = Reserved.

10.4.11 Auto-Negotiation Link Partner Base Page Ability Register: Address 7.13

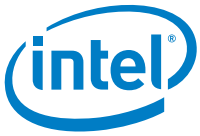
Bit	Name	Type	Default	Description
F	Link Partner Next Page Ability	RO		If next page ability is not supported, the *NP* bit must always be set to 0b. If the X540 implements next page ability and needs to engage in a next page exchange, it must set the *NP* bit to 1b. 1b = Next page ability. 0b = Next page ability not supported or not engaged.
E	Link Partner Base Page Acknowledge	RO		This bit is used by the auto-negotiation function to indicate that a device has successfully received its link partner's link code word. 1b = Acknowledge.

D	Link Partner Remote Fault	RO		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.
C	Link Partner Extended Next Page Ability	RO		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. 1b = Extended next page capable. 0b = Not capable of extended next pages.
B:5	Link Partner Technology Ability Field [6:0]	RO		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 X540 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 = 10BASE-T. Bit 1 = 10BASE-T full duplex. 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.
4:0	Link Partner Selector Field [4:0]	RO		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: 0x00b = Reserved. 0x01b = IEEE 802.3. 0x02 = IEEE 802.9 ISLAN-16T. 0x03 = IEEE 802.5. 0x04 = IEEE 1394. 0x05:0x1F = Reserved.

10.4.12 Auto-Negotiation Extended Next Page Transmit Register: Address 7.16[1]

Bit	Name	Type	Default	Description
F	Next Page	R/W	0b	Next page is used by the next page function to indicate whether or not this is the last next page to be transmitted. 1b = Additional next page follows. 0b = Last page.
E	Reserved			RW - Reserved, do not modify.
D	Message Page	R/W	0b	Message page is used by the next page function to differentiate a message page from an unformatted page. 1b = Message page. 0b = Unformatted page.
C	Acknowledge 2	R/W	0b	This field is used by the next page function to indicate that a device has the ability to comply with the message. 1b = Complies with corresponding message. 0b = Cannot comply with corresponding message.
B	Reserved			RW - Reserved, do not modify.
A:0	Message Code Field [A:0]	R/W PD	0x001	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: 0x0b = Reserved. 0x1b = Null message. 0x2:0x3 = Reserved expansion message. 0x4 = Remote fault details message. 0x5 = OUI message. 0x6 = PHY ID message. 0x7 = 100BASE-T2 message. 0x8 = 1000BASE-T message. 0x9 = 10GBASE-T message.

1. 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.



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]	R/W PD	0x0000	[F:0] = 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]	R/W PD	0x0000	[F:0] = Unformatted Code Field 2.

10.4.15 Auto-Negotiation Link Partner Extended Next Page Ability Register: Address 7.19[1]

Bit	Name	Type	Default	Description
F	Link Partner Next Page	RO		1b = Next page ability.
E	Link Partner Extended Next Page Acknowledge	RO		This field 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.
D	Link Partner Message Page	RO		1b = Message page. 0b = Unformatted page.

1. 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
C	Link Partner Acknowledge 2	RO		1b = Link partner acknowledges that they can comply with the current next page. 0b = Link partner cannot comply with the current next page.
B	Link Partner Toggle	RO		Set opposite of the corresponding bit in the previous page. Value of link partner's toggle bit.
A:0	Link Partner Message Code Field [A:0]	RO		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: 0x0b = Reserved. 0x1b = Null message. 0x2:0x3 = Reserved expansion message. 0x4 = Remote fault details message. 0x5 = OUI message. 0x6 = PHY ID message. 0x7 = 100BASE-T2 message. 0x8 = 1000BASE-T message. 0x9 = 10GBASE-T message.

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		[F:0] = 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		[F:0] = Unformatted Code Field 2 [15:0].

10.4.18 Auto-Negotiation 10GBASE-T Control Register: Address 7.20

Bit	Name	Type	Default	Description
F	Master/Slave Manual Configuration Enable	R/W PD	0b	1b = Enable master/slave manual configuration. 0b = Disable master/slave manual configuration.
E	Master/Slave Configuration	R/W PD	0b	1b = Master. 0b = Slave.
D	Port Type	R/W PD	0b	1b = Multiport device. 0b = Single port device.
C	10GBASE-T Ability	R/W PD	1b	1b = Advertise PHY as 10GBASE-T capable. 0b = Do not advertise PHY as 10GBASE-T capable.
B:3	Reserved			RW - Reserved, do not modify.

Bit	Name	Type	Default	Description
2	LD PMA Training Reset Request	R/W PD	0b	1b = Local device requests that Link Partner reset PMA training PRBS every frame. 0b = Local device requests that Link Partner run PMA training PRBS continuously.
1	Reserved			RW - Reserved, do not modify.
0	LD Loop Timing Ability	R/W PD	1b	1b = Advertise PHY as capable of loop timing 0b = Do not advertise PHY as capable of loop timing.

10.4.19 Auto-Negotiation 10GBASE-T Status Register: Address 7.21

Bit	Name	Type	Default	Description
F	Master/Slave Configuration Fault	LH		1b = Master/slave configuration fault.
E	Master/Slave Configuration Resolution	RO		1b = Local PHY resolved to master. 0b = Local PHY resolved to slave.
D	Local Receiver Status	RO		Set by the microcontroller. 1b = Local receiver operational. 0b = Local receiver not operational.
C	Remote Receiver Status	RO		Set by the microcontroller. 1b = Remote receiver operational. 0b = Remote receiver not operational.
B	Link Partner 10GBASE-T Ability	RO		This bit is only valid when the *Page Received* bit 7.1.6 is set to 1b. 1b = Link partner is 10GBASE-T capable. 0b = Link partner is not 10GBASE-T capable.

Bit	Name	Type	Default	Description
A	Link Partner Loop Timing Ability	RO		1b = Link partner is capable of loop timing. 0b = Link partner is not capable of loop timing.
9	Link Partner Training Reset Request	RO		1b = Link partner has requested that PMA PRBS training be reset every frame. 0b = Link partner has requested that PMA PRBS training run continuously.
8:0	Reserved			RW - Reserved, do not modify.

10.4.20 Auto-Negotiation Vendor Provisioning 1: Address 7.C400

Bit	Name	Type	Default	Description
F	1000BASE-T Full Duplex Ability	R/W PD	1b	1b = Advertise PHY as 1000BASE-T full duplex capable. 0b = Do not advertise PHY as 1000BASE-T full duplex capable.
E:D	Reserved			RW - Reserved, do not modify.
C:7	Reserved			RW - Reserved, do not modify.
6	Exchange PHY ID Information	R/W PD	1b	1b = Exchange PHY ID information.

5	User Provided Auto-Negotiation Data	R/W PD	0b	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 connect), the PHY then 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. 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. 0b = Construct the correct auto-negotiation words based on the register settings of 7.10, 7.20, and 7.C400.
4	Automatic Downshift Enable	R/W PD	1b	1b = Enable automatic downshift. 0b = Manual downshift.
3:0	Retry Attempts Before Downshift [3:0]	R/W PD	0x4	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. Number of retry attempts before downshift.

10.4.21 Auto-Negotiation Reserved Vendor Provisioning 1: Address 7.C410

Bit	Name	Type	Default	Description
F:E	Reserved			RW - Reserved, do not modify.
D:B	Reserved Provisioning 0 [2:0]	R/W PD	0x0	Reserved for future use.
A:8	Semi-Cross Link Attempt Period [2:0]	R/W PD	0x0	Number of failed link attempts before trying semi cross. Set to 0x0 to disable semi-cross. Set to 0x7 to always use semi-cross.
7	WoL Mode	R/W PD	0b	This bit indicates whether the X540 goes into 100BASE-TX or 1000BASE-T WoL operation. 0b = 100BASE-TX 1b = 1000BASE-T

6	WoL Enable	R/W PD	0b	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.
5:2	Extra Page Count [3:0]	R/W PD	0x4	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 X540 starts it's training. This might confuse the link partner MDI/MDI-X state machine. Sending extra pages seems to correct this problem.
1:0	MDI / MDI-X Control [1:0]	R/W PD	0x0	These bits are used to force a manual MDI or MDI-X configuration. 0x0 = Automatic MDI / MDI-X operation 0x1 = Manual MDI. 0x2 = Manual MDI-X. 0x3 = Reserved.

10.4.22 Auto-Negotiation Reserved Vendor Provisioning 2: Address 7.C411

Bit	Name	Type	Default	Description
F:C	Autonegotiation Timeout [3:0]	R/W PD	0x3	These bits control the auto-negotiation timeout watchdog use.
B	Autonegotiation Timeout Mode	R/W PD	1b	0b = Enable timeout for all auto-negotiation behavior. 1b = Enable timeout only if the following are enabled: • LPLU • Downshifting
A	Autonegotiation Timeout Status	R/W PD	0b	1b = Indicates that an auto-negotiation timeout has occurred. 0b = Indicates that an auto-negotiation timeout has not occurred.
9:5	Reserved			Reserved for future use.
4:1	Low-Power Link-Up Upshift Threshold [3:0]	R/W PD	0x2	This field sets the number of times the PHY attempts to connect at the lowest rate before giving up and moving to the next highest rate.
0	Low-Power Link-Up Upshift Enable	R/W PD	1b	Low-power link-up upshifting is the reverse of auto-negotiation downshifting. Fox example, low-power link-up attempts to connect at the lowest rate, and if this fails, moves to the next highest rate. The signals for LPLU from the MAC are in 1E.1203 - 1E.1204. 1b = Enable low-power link-up upshifting. 0b = Disable low-power link-up upshifting.

10.4.23 Auto-Negotiation Vendor Status 1: Address 7.C800

Bit	Name	Type	Default	Description
F:3	Reserved			RW - Reserved, do not modify.
2:1	Connect Rate [1:0]	RO		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: 0x3 = 10GBASE-T. 0x2 = 1000BASE-T. 0x1 = 100BASE-TX. 0x0 = Reserved
0	Connect Type	RO		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: 1b = Full duplex. 0b = Reserved

10.4.24 Auto-Negotiation Reserved Vendor Status 1: Address 7.C810

Bit	Name	Type	Default	Description
F	Reserved	RO		Reserved, not supported.
E	Device Present	RO		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. 1b = Far-end Ethernet device present. 0b = No far-end Ethernet device detected.

Bit	Name	Type	Default	Description
D:9	Connection State [4:0]	RO		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 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. 0x0C:0x1E = Reserved. 0x1F = Invalid.
8	MDI/MDI-X	RO		When auto-negotiation completes, this register indicates whether the connection was made as an MDI or MDI-X connection. 0b = MDI. 1b = MDI-X.
7	Duplicate Link Partner Auto-Negotiation Ability	RO		The link partner is capable of auto-negotiation. This is a duplicate of the bit at 07.0001.0.
6:2	Reserved Status 1 [6:2]	RO		Reserved for future use.
1	Receive PAUSE Resolution	RO		PAUSE resolution from 28B-3 1b = Receive PAUSE enabled. 0b = Receive PAUSE disabled.
0	Transmit PAUSE Resolution	RO		PAUSE resolution from 28B-3 1b = Transmit PAUSE enabled. 0b = Transmit PAUSE disabled.

10.4.25 Auto-Negotiation Reserved Vendor Status 2: Address 7.C811

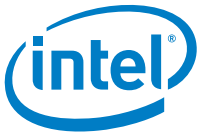
Bit	Name	Type	Default	Description
F:0	Auto-Negotiation Attempts [F:0]	RO		This is a rolling counter (at saturation reverts to zero). It is cleared at reset or after a successful connection completes. The number of auto-negotiation attempts since the last successful connection (or power-up).

10.4.26 Auto-Negotiation Transmit Vendor Alarms 1: Address 7.CC00

Bit	Name	Type	Default	Description
F:4	Reserved			RW - Reserved, do not modify.
3	Auto-Negotiation Completed For Non-Supported Rate	LH		This means that the X540 has completed auto-negotiation and was unable to agree on a rate that both could operate at. Note that this indication should be ignored in the case of master/slave resolution fault. 1b = Auto-negotiation has completed for a rate that is not supported by the X540.
2	Auto-Negotiation Completed for Supported Rate	LH		1b = Auto-negotiation has completed successfully for a rate that is supported by the X540.
1	Automatic Downshift	LH		1b = Automatic downshift has occurred.
0	Connection State Change	LH		This interrupt indicates a change in the *Connection State* [D:B] in Auto-Negotiation Reserved Vendor Status 1: Address 7.C810 register. Note that this indicates any state change versus 7.CC01.0, which indicates a connect or disconnect event. 1b = The connection state has changed.

10.4.27 Auto-Negotiation Transmit Vendor Alarms 2: Address 7.CC01

Bit	Name	Type	Default	Description
F:1	Reserved Vendor Alarms 2 [E:0]	LH		Reserved for future use.
0	Link Connect/ Disconnect	LH		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.



10.4.28 Auto-Negotiation Transmit Vendor Alarms 3: Address 7.CC02

Bit	Name	Type	Default	Description
F:0	Reserved Vendor Alarms 3 [F:0]	LH		Reserved for future use.

10.4.29 Auto-Negotiation Standard Interrupt Mask 1: Address 7.D000

Bit	Name	Type	Default	Description
F:7	Reserved			RW - Reserved, do not modify.
6	Extended Next Page Received Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
5	Reserved			RW - Reserved, do not modify.
4	Remote Fault Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
3	Reserved			RW - Reserved, do not modify.
2	Link Status Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
1:0	Reserved			RW - Reserved, do not modify.

10.4.30 Auto-Negotiation Standard Interrupt Mask 2: Address 7.D001

Bit	Name	Type	Default	Description
F	Master/Slave Configuration Fault Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
E:0	Reserved			RW - Reserved, do not modify.

10.4.31 Auto-Negotiation Transmit Vendor Interrupt Mask 1: Address 7.D400

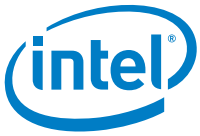
Bit	Name	Type	Default	Description
F:4	Reserved			RW - Reserved, do not modify.
3	Auto-Negotiation Completed For Non-Supported Rate Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
2	Auto-Negotiation Completed for Supported Rate Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
1	Automatic Downshift Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
0	Connection State Change Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.

10.4.32 Auto-Negotiation Transmit Vendor Interrupt Mask 2: Address 7.D401

Bit	Name	Type	Default	Description
F:0	Reserved Vendor Alarms 2 Mask [F:0]	R/W PD	0x0000	Reserved for future use.
0	Link Connect/ Disconnect Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.

10.4.33 Auto-Negotiation Transmit Vendor Interrupt Mask 3: Address 7.D402

Bit	Name	Type	Default	Description
F:0	Reserved			RW - Reserved, do not modify.



10.4.34 Auto-Negotiation Receive Vendor Provisioning 1: Address 7.E410

Bit	Name	Type	Default	Description
F:1	Reserved Receive Provisioning 1 [E:0]	R/W PD	0x0000	Reserved for future use.
0	Disable 100BASE-TX Parallel Detection	R/W PD	1b	1b = Disable 100BASE-TX parallel detection. 0b = Normal operation (100BASE-TX parallel detection enabled).

10.4.35 Auto-Negotiation Receive Link Partner Status 1: Address 7.E820

Bit	Name	Type	Default	Description
F	Link Partner 1000BASE-T Full Duplex Ability	RO		1b = Link partner is 1000BASE-T full-duplex capable. 0b = Link partner is not 1000BASE-T full-duplex capable.
E	Link Partner 1000BASE-T Half Duplex Ability	RO		1b = Link partner is 1000BASE-T half-duplex capable. 0b = Link partner is not 1000BASE-T half-duplex capable.
D	Link Partner Short-Reach	RO		1b = Link partner is operating in short-reach mode. 0b = Link partner is not operating in short-reach mode.
C:0	Reserved			RW - Reserved, do not modify.

10.4.36 Auto-Negotiation Receive Link Partner Status 4: Address 7.E823

Bit	Name	Type	Default	Description
F:8	Link Partner Firmware Major Revision Number [7:0]	RO		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. [F:8] = Link partner firmware major revision number. [7:0] = Link partner firmware minor revision number.
7:0	Link Partner Firmware Minor Revision Number [7:0]	RO		

10.4.37 Auto-Negotiation Receive Reserved Vendor Status 3: Address 7.E832

Bit	Name	Type	Default	Description
F:1	Reserved Receive Status 3 [E:0]	RO		Reserved for future use.
0	Link Partner AFR Enabled	RO		1b = The link partner has fast reframe capability enabled.

10.4.38 Auto-Negotiation Receive Vendor Alarms 1: Address 7.EC00

Bit	Name	Type	Default	Description
F:0	Reserved Receive Vendor Alarms 1 [F:0]	LH		Reserved

10.4.39 Auto-Negotiation Receive Vendor Alarms 2: Address 7.EC01

Bit	Name	Type	Default	Description
F	100BASE-TX Device Detect	LH		This is set when the presence of a 1 V peak-to-peak MLT-3 signal is found on the line. 1b = 100BASE-TX device detected.
E	Reserved	LH		Reserved, not supported.
D	Autonegotiation Protocol Error	LH		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.
C	FLP Idle Error	LH		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.
B:4	Reserved Receive Vendor Alarms 2 [7:0]	LH		Reserved
3	Next Page 3rd Received	LH		1b = 3rd next page received and placed in register 7.1B.

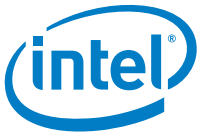
2	Next Page 2nd Received	LH		1b = 2nd next page received and placed in register 7.1A.
1	Next Page 1st Received	LH		1b = 1st next page received and placed in register 7.19.
0	Base Page Received	LH		1b = Base page received and placed in register 7.13.

10.4.40 Auto-Negotiation Receive Vendor Alarms 3: Address 7.EC02

Bit	Name	Type	Default	Description
F:3	Reserved			RW - Reserved, do not modify.
2	10BASE-T Device Detect	LL		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.
1:0	Reserved			RW - Reserved, do not modify.

10.4.41 Auto-Negotiation Receive Vendor Alarms 4: Address 7.EC03

Bit	Name	Type	Default	Description
F:1	Reserved Receive Vendor Alarms 4 [E:0]	LH		Reserved
0	100BASE-TX Parallel Detect	LH		1b = 100BASE-TX parallel event detection circuit.



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]	R/W PD	0x0000	1b = Enable interrupt generation. 0b = Disable interrupt generation.

10.4.43 Auto-Negotiation Receive Vendor Interrupt Mask 2: Address 7.F401

Bit	Name	Type	Default	Description
F	100BASE-TX Device Detect Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
E	Reserved	R/W PD	0b	Reserved, not supported.
D	Auto-negotiation Protocol Error Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
C	FLP Idle Error Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
B:4	Reserved Receive Vendor Alarms 2 Mask [7:0]	R/W PD	0x00	1b = Enable interrupt generation. 0b = Disable interrupt generation.
3	Next Page 3rd Received Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
2	Next Page 2nd Received Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
1	Next Page 1st Received Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
0	Base Page Received Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.

10.4.44 Auto-Negotiation Receive Vendor Interrupt Mask 3: Address 7.F402

Bit	Name	Type	Default	Description
F:3	Reserved			RW - Reserved, do not modify.
2	10BASE-T Device Detect Mask	R/W PD	0	1b = Enable interrupt generation. 0b = Disable interrupt generation.
1:0	Reserved			RW - Reserved, do not modify.

10.4.45 Auto-Negotiation Receive Vendor Interrupt Mask 4: Address 7.F403

Bit	Name	Type	Default	Description
F:1	Reserved Receive Vendor Alarms 4 Mask [E:0]	R/W PD	0x0000	1b = Enable interrupt generation. 0b = Disable interrupt generation.
0	100BASE-TX Parallel Detect Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.

10.4.46 Auto-Negotiation Vendor Global Interrupt Flags 1: Address 7.FC00

Bit	Name	Type	Default	Description
F	Standard Alarms 1 Interrupt	RO		An interrupt was generated from status register () and the corresponding mask register (Auto-Negotiation Standard Interrupt Mask 1: Address 7.D000). 1b = Interrupt.
E	Standard Alarms 2 Interrupt	RO		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.
D:B	Reserved			RW - Reserved, do not modify.

A	Vendor Specific Alarms 1 Interrupt	RO		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.
9	Vendor Specific Alarms 2 Interrupt	RO		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 1: Address 7.D401). 1b = Interrupt.
8	Vendor Specific Alarms 3 Interrupt	RO		An interrupt was generated from status register (Auto-Negotiation Transmit Vendor Alarms 3: Address 7.CC02) and the corresponding mask register (Auto-Negotiation Transmit Vendor Interrupt Mask 1: Address 7.D402). 1b = Interrupt.
7:4	Reserved			RW - Reserved, do not modify.
3	Vendor Specific Rx Alarms 1 Interrupt	RO		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.
2	Vendor Specific Rx Alarms 2 Interrupt	RO		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.
1	Vendor Specific Rx Alarms 3 Interrupt	RO		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.
0	Vendor Specific Rx Alarms 4 Interrupt	RO		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.

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		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		Bits 15 - 0 of the device ID.

10.5.3 GbE Standard Devices in Package 1: Address 1D.5

Bit	Name	Type	Default	Description
F:8	Reserved			RW - Reserved, do not modify.
7	Auto-Negotiation Present	ROS	1b	This is always set to 1b because there is auto-negotiation in the X540. 1b = Auto-negotiation is present in the package. 0b = Auto-negotiation is not present in the package.
6	TC Present	ROS	0b	This is always set to 0b because there is no TC functionality in the X540. 1b = TC is present in the package. 0b = TC is not present in the package.
5	DTE XS Present	ROS	0b	This is always set to 0b because there is no MAC XAUI interface in the X540. 1b = DTE XS is present in the package. 0b = DTE XS is not present in the package.

Bit	Name	Type	Default	Description
4	Control Present	ROS	1b	This is always set to 1b because there is a PHY XAUI interface in the X540. 1b = Control is present in the package. 0b = Control is not present in the package.
3	PCS Present	ROS	1b	This is always set to 1b because there is PCS functionality in the X540. 1b = PCS is present in the package. 0b = PCS is not present in the package.
2	WIS Present	ROS	0b	This is always set to 0b because there is no WIS functionality in the X540. 1b = WIS is present in the package. 0b = WIS is not present in the package.
1	PMA Present	ROS	1b	This is always set to 1b because there is PMA functionality in the X540. 1b = PMA is present in the package. 0b = PMA is not present.
0	Clause 22 Registers Present	ROS	0b	This is always set to 0b because there are no Clause 22 registers in the X540. 1b = Clause 22 registers are present in the package. 0b = Clause 22 registers are not present in the package.

10.5.4 GbE Standard Vendor Devices in Package 2: Address 1D.6

Bit	Name	Type	Default	Description
F	Vendor Specific Device #2 Present	ROS	1b	This is always set to 1b because the X540 uses this device for the DSP PMA registers. 1b = Device #2 is present in the package. 0b = Device #2 is not present in the package.
E	Vendor Specific Device #1 Present	ROS	1b	This is always set to 1b because the X540 uses this device for the global control registers. 1b = Device #1 is present in the package. 0b = Device #1 is not present in the package.
D	Clause 22 Extension Present	ROS	1b	This is always set to 1b because the X540 uses this device for the global control registers. 1b = Clause 22 Extension is present in the package. 0b = Clause 22 Extension is not present in the package.
C:0	Reserved			RW - Reserved, do not modify.

10.5.5 GbE Standard Status 2: Address 1D.8

Bit	Name	Type	Default	Description
F:E	Device Present [1:0]	ROS	0x2	This field is always set to 0x2 because the control is present in the X540. [F:E]: 0x3 = No device at this address. 0x2 = Device present at this address. 0x1 = No device at this address. 0x0 = No device at this address.
D:0	Reserved			RW - Reserved, do not modify.

10.5.6 GbE Standard Package Identifier 1: Address 1D.E

Bit	Name	Type	Default	Description
F:0	Package ID MSW [1F:10]	RO		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		Bits 15 - 0 of the package ID.

10.5.8 GbE Reserved Provisioning 2: Address 1D.C501

Bit	Name	Type	Default	Description
F:D	Test Mode [2:0]	R/W P/D	0x0	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. 101b:1111b = Reserved.
C:2	Reserved Provisioning 2 [A:0]	R/W PD	0x0000	Reserved for future use.
1:0	100BASE-TX Test Mode [1:0]	R/W	0x0	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.

10.6 Global Registers

10.6.1 Global Standard Control 1: Address 1E.0

Bit	Name	Type	Default	Description
F:C	Reserved			RW - Reserved, do not modify.
B	Low Power	R/W PD	0	A 1b written to this register causes the corresponding the X540 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. 1b = Low-power mode. 0b = Normal operation.
EA:0	Reserved			RW - 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		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		Bits 15 - 0 of the device ID.

10.6.4 Global Standard Devices in Package 1: Address 1E.5

Bit	Name	Type	Default	Description
F:8	Reserved			RW - Reserved, do not modify.
7	Auto-Negotiation Present	RO S	1b	This is always set to 1b because there is auto-negotiation in the X540. 1b = Auto-negotiation is present in the package. 0b = Auto-negotiation is not present in the package.
6	TC Present	RO S	0b	This is always set to 0b because there is no TC functionality in the X540. 1b = TC is present in the package. 0b = TC is not present in the package.
5	DTE XS Present	RO S	0b	This is always set to 0b because there is no DTE XAUI interface in the X540. 1b = DTE XS is present in the package. 0b = DTE XS is not present in the package.
4	PHY XS Present	RO S	0b	This is always set to 0b because there is no PHY XS interface in the X540. 1b = PHY XS is present in the package. 0b = PHY XS is not present in the package.

Bit	Name	Type	Default	Description
3	PCS Present	ROS	1b	This is always set to 1b because there is PCS functionality in the X540. 1b = PCS is present in the package. 0b = PCS is not present in the package.
2	WIS Present	ROS	0b	This is always set to 0b because there is no WIS functionality in the X540. 1b = WIS is present in the package. 0b = WIS is not present in the package.
1	PMA Present	ROS	1b	This is always set to 1b because there is PMA functionality in the X540. 1b = PMA is present in the package. 0b = PMA is not present.
0	Clause 22 Registers Present	ROS	0b	This is always set to 0b because there are no Clause 22 registers in the X540. 1b = Clause 22 registers are present in the package. 0b = Clause 22 registers are not present in the package.

10.6.5 Global Standard Vendor Devices in Package 2: Address 1E.6

Bit	Name	Type	Default	Description
F	Vendor Specific Device #2 Present	ROS	1	This is always set to 1b because the X540 uses this device for the DSP PMA registers. 1b = Device #2 is present in the package. 0b = Device #2 is not present in the package.
E	Vendor Specific Device #1 Present	ROS	1	This is always set to 1b because the X540 uses this device for the global control registers. 1b = Device #1 is present in the package. 0b = Device #1 is not present in the package.
D	Clause 22 Extension Present	ROS	1	This is always set to 1b because the X540 uses this device for the GbE registers. 1b = Clause 22 extension is present in the package. 0b = Clause 22 extension is not present in the package.
C:0	Reserved			RW - Reserved, do not modify.

10.6.6 Global Standard Status 2: Address 1E.8

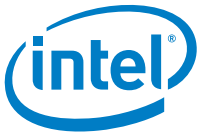
Bit	Name	Type	Default	Description
F:E	Device Present [1:0]	ROS	0x2	This field is always set to 0x2 because the global MMD resides here in the X540. [F:E]: 0x3 = No device at this address. 0x2 = Device present at this address. 0x1 = No device at this address. 0x0 = No device at this address.
D:0	Reserved			RW - Reserved, do not modify.

10.6.7 Global Standard Package Identifier 1: Address 1E.E

Bit	Name	Type	Default	Description
F:0	Package ID MSW [1F:10]	RO		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		Bits 15 - 0 of the package ID.



10.6.9 Global Firmware ID: Address 1E.20

Bit	Name	Type	Default	Description
F:8	Firmware Major Revision Number [7:0]	RO		[F:8] = Major revision number. The lower six bits of major and minor firmware revision are exchanged in auto-negotiation when the PHYID message is sent.
7:0	Firmware Minor Revision Number [7:0]	RO		[7:0] = Minor 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 Chip Identification: Address 1E.21

Bit	Name	Type	Default	Description
F:0	Chip Identification [F:0]	RO		Hardware Chip ID. This value is a hard-coded chip ID.

10.6.11 Global Chip Revision: Address 1E.22

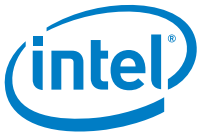
Bit	Name	Type	Default	Description
F:0	Chip Revision [F:0]	RO		Hardware Chip Revision. This value is a hard-coded chip revision.

10.6.12 Global Diagnostic Provisioning: Address 1E.C400

Bit	Name	Type	Default	Description
F	Enable Diagnostics	R/W PD	0b	1b = the X540 performs diagnostics on power up.
E:0	Reserved			RW - Reserved, do not modify.

10.6.13 Global General Provisioning 3: Address 1E.C442

Bit	Name	Type	Default	Description
F:1	Reserved			RW - Reserved, do not modify
0	Daisy Chain Reset	R/W	0b	Toggling this bit from 0b to 1b reloads the IRAM and DRAM and resets the microprocessor. The microprocessor is in microprocessor run stall during the reload process. After the reload process, the microprocessor run stall is deasserted and the microprocessor reset is asserted. Note that before setting this bit, the *Soft Reset* bit needs to be deasserted.



10.6.14 Global Reserved Provisioning 1: Address 1E.C470

Bit	Name	Type	Default	Description
F	Diagnostics Select	R/W PD	0b	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 X540 is reset. Conversly, the only way to exit this routine once it starts is to issue a PMA reset. Bit F settings: • 1b = Provide extended MDI diagnostics information. • 0b = Provide normal cable diagnostics. Bit E:D settings: • 0x0 = TDR data. • 0x1 = RFI channel PSD. • 0x2 = Noise PSD while the local Tx is off. • 0x3 = Noise PSD while the local Tx is on.
E:D	Extended MDI Diagnostics Select [1:0]	R/W PD	0x0	
C:8	Reserved			RW - Reserved, do not modify.
7	Trigger Diagnostic Interrupt	R/W SC	0b	This bit is used by system diagnostic routines to generate an alarm to test interrupt circuitry. This register is self clearing. 1b = Trigger an alarm.
6:5	Reserved			RW - Reserved, do not modify.

Bit	Name	Type	Default	Description
4	Initiate Cable Diagnostics	R/W SC	0b	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.
3:0	Reserved			RW - Reserved, do not modify.

10.6.15 Global Reserved Provisioning 3: Address 1E.C472

Bit	Name	Type	Default	Description
F	Enable LVDD Power Supply Tuning	R/W PD	0b	This bit controls whether the PHY attempts to tune the external LVDD power supply via the PMBus. 1b = Enable external LVDD power supply tuning. 0b = Disable external LVDD power supply tuning.
E	Enable VDD Power Supply Tuning	R/W PD	0b	This bit controls whether the PHY attempts to tune the external VDD power supply via the PMBus. 1b = Enable external VDD power supply tuning. 0b = Disable external VDD power supply tuning.
D:8	Reserved			RW - Reserved, do not modify
7	Tunable External LVDD Power Supply Present	R/W PD	0b	This bit must be set if tuning of external LVDD power supply is desired (see bit F). 1b = Tunable external LVDD power supply present 0b = No tunable external LVDD power supply present
6	Tunable External VDD Power Supply Present	R/W PD	0b	This bit must be set if tuning of external VDD power supply is desired (see bit E). 1b = Tunable external VDD power supply present 0b = No tunable external VDD power supply present
5:1				
0	Enable 5th Channel RFI Cancellation	R/WP PU	1b	Note: The value of this bit at the time of auto-negotiation sets the local PHY behavior until the next time auto-negotiation occurs.

10.6.16 Global Reserved Provisioning 5: Address 1E.C474

Bit	Name	Type	Default	Description
F:8	Reserved			RW - Reserved, do not modify.
7:1	Training SNR [6:0]	R/W PD	0x00	SNR during 10G training on the worst channel. SNR is in steps of 0.2dB. The SNR margin that is seen by the worst channel, over and above the minimum SNR required to operate at a BER of 10^{-12}. It is reported with 0.2 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 0x4000.
0	Flash Boot-Load Kick-Start	R/W PD	0x0	Writing a 1b to this bit triggers the Flash boot load kick start, which causes the Flash boot state machine to read from the Flash without reloading the microprocessor IRAM/DRAM or resetting the microprocessor.

10.6.17 Global Reserved Provisioning 6: Address 1E.C475

Bit	Name	Type	Default	Description
D:C	Reserved			RW - Reserved, do not modify.
B	CFR LP Disable Timer	R/W PD	0b	1b = Link partner requires cfr_disable timer 0b = Link partner does not require cfr_disable timer
A	CFR LP Extended Maxwait	R/W PD	0b	1b = Link partner requires extended max wait 0b = Link partner does not require extended maxwait
9	CFR LP THP	R/W PD	0b	1b = Link partner requires local PHY to enable THP 0b = Link partner does not require local PHY to enable THP
8	CFR LP Support	R/W PD	0b	1b = Link partner supports Cisco Fast Retrain 0b = Link partner does support Cisco Fast Retrain
7	CFR Disable Timer	R/W PD	0b	1b = Local PHY requires cfr_disable timer 0b = Local PHY does not require cfr_disable timer

Bit	Name	Type	Default	Description
6	CFR Extended Maxwait	R/W PD	1b	1b = Local PHY requires extended maxwait 0b = Local PHY does not require extended maxwait
5	CFR THP	R/W PD	0b	1b = Local PHY requires local PHY to enable THP 0b = Local PHY does not require local PHY to enable THP
4	CFR Support	R/W PD	1b	1b = Local PHY supports Cisco Fast Retrain 0b = Local PHY does support Cisco Fast Retrain
3:0	Reserved			RW - Reserved, do not modify.

10.6.18 Global Cable Diagnostic Status 1: Address 1E.C800

Bit	Name	Type	Default	Description
F	Reserved			RW - Reserved, do not modify.
E:C	Pair A Status [2:0]	RO		This register summarizes the worst case impairment on pair A. [F:D]: 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 011b = Connected to Pair B. 010b = Connected to Pair C. 001b = Connected to Pair D. 000b = OK.
B	Reserved			RW - Reserved, do not modify.
A:8	Pair B Status [2:0]	RO		This register summarizes the worst case impairment on pair B. [C:A]: 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 011b = Connected to Pair C. 010b = Connected to Pair D. 001b = Connected to Pair A. 000b = OK.
7	Reserved			RW - Reserved, do not modify.

Bit	Name	Type	Default	Description
6:4	Pair C Status [2:0]	RO		This register summarizes the worst case impairment on pair C. [9:7] 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 011b = Connected to Pair D. 010b = Connected to Pair A. 001b = Connected to Pair B. 000b = OK.
3	Reserved			RW - Reserved, do not modify.
2:0	Pair D Status [2:0]	RO		This register summarizes the worst case impairment on pair D. [6:4] 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 011b = Connected to Pair A. 010b = Connected to Pair B. 001b = Connected to Pair C. 000b = OK.

10.6.19 Global Cable Diagnostic Status 2: Address 1E.C801

Bit	Name	Type	Default	Description
F:8	Pair A Reflection #1 [7:0]	RO		The distance in meters (accurate to ±1 m) of the first 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. A value of zero indicates that this reflection does not exist or was not computed.
7:0	Pair A Reflection #2 [7:0]	RO		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. A value of zero indicates that this reflection does not exist or was not computed.

10.6.20 Global Cable Diagnostic Status 3: Address 1E.C802

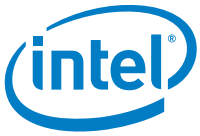
Bit	Name	Type	Default	Description
F:0	Impulse Response MSW [F:0]	RO		The MSW of the memory location that contains the start of the impulse response data for the extended diagnostic type in 1E.C470.E:D. See 1E.C470 for more information.

10.6.21 Global Cable Diagnostic Status 4: Address 1E.C803

Bit	Name	Type	Default	Description
F:8	Pair B Reflection #1 [7:0]	RO		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. A value of zero indicates that this reflection does not exist or was not computed.
7:0	Pair B Reflection #2 [7:0]	RO		

10.6.22 Global Cable Diagnostic Status 5: Address 1E.C804

Bit	Name	Type	Default	Description
F:0	Impulse Response LSW [F:0]	RO		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 for more information.



10.6.23 Global Cable Diagnostic Status 6: Address 1E.C805

Bit	Name	Type	Default	Description
F:8	Pair C Reflection #1 [7:0]	RO		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. A value of zero indicates that this reflection does not exist or was not computed.
7:0	Pair C Reflection #2 [7:0]	RO		

10.6.24 Global Cable Diagnostic Status 7: Address 1E.C806

Bit	Name	Type	Default	Description
F:0	Reserved1 [F:0]	RO		Reserved.

10.6.25 Global Cable Diagnostic Status 8: Address 1E.C807

Bit	Name	Type	Default	Description
F:8	Pair D Reflection #1 [7:0]	RO		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. A value of zero indicates that this reflection does not exist or was not computed.
7:0	Pair D Reflection #2 [7:0]	RO		

10.6.26 Global Cable Diagnostic Status 9: Address 1E.C808

Bit	Name	Type	Default	Description
F:0	Reserved2 [F:0]	RO		Reserved.

10.6.27 Global Fault Message: Address 1E.C850

Bit	Name	Type	Default	Description
F:0	Message [F:0]	RO		Error code describing fault. Code 0x8001: Firmware is not compatible with the chip architecture. This fault occurs when firmware is compiled for a different core that is loaded. Code 0x8002: VCO calibration failed. This occurs when the main PLLs on the chip fail to lock (no trigger is possible). Code 0x8003: XAUI calibration failed. This occurs when the XAUI PLLs fail to lock (no trigger is possible). Code 0x8004: Failed to set operating voltages via PMBus. This only occurs when the processor has control over the power supply voltage via an attached PMBus device and there is a protocol error on the I^2C bus (no trigger is possible). Code 0x8005: Unexpected device ID. This occurs if the device ID programmed into the internal E-Fuse registers in not valid (no trigger is possible). Code 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. Code 0x8007: Detected a bit error in static memory. To trigger, corrupt one of the static regions. Code 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. Code 0xC002 Instruction Fetch Error. Internal physical address or a data error during instruction fetch (no trigger is possible). Code 0xC003 Load Store Error. Internal physical address or data error during load store operation (no trigger is possible). Code 0xC004 Privileged Instruction. Attempt to execute a privileged operation without sufficient privilege (no trigger is possible). Code 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). Code 0xC006 Instruction fetch from prohibited space (no trigger is possible). Code 0xC007 Data load from prohibited space (no trigger is possible). Code 0xC008 Data store into prohibited space (no trigger is possible).

10.6.28 Global Primary Status: Address 1E.C851

Bit	Name	Type	Default	Description
F:1	Reserved			Value always 0b, writes are ignored.
0	Primary Status	RO		1b = PHY is the primary PHY. 0b = PHY is the secondary PHY.

10.6.29 Global Cable Diagnostic Impedance 1: Address 1E.C880

Bit	Name	Type	Default	Description
F	Reserved 1	RO		Reserved
E:C	Pair A Reflection #1 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.
B	Reserved 2	RO		Reserved
A:8	Pair A Reflection #2 [2:0]	RO		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. 011b = Open circuit (> 300 Ω). 010b = High mismatch (> 115 Ω). 001b = Low mismatch (< 85 Ω). 000b = Short circuit (< 30 Ω).
7	Reserved 3	RO		Reserved
6:4	Pair A Reflection #3 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.

Bit	Name	Type	Default	Description
3	Reserved 4	RO		Reserved
2:0	Pair A Reflection #4 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.

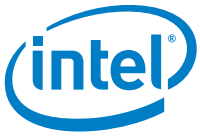
10.6.30 Global Cable Diagnostic Impedance 2: Address 1E.C881

Bit	Name	Type	Default	Description
F	Reserved 5	RO		Reserved
E:C	Pair B Reflection #1 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.
B	Reserved 6	RO		Reserved
A:8	Pair B Reflection #2 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.
7	Reserved 7	RO		Reserved
6:4	Pair B Reflection #3 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.

Bit	Name	Type	Default	Description
3	Reserved 8	RO		Reserved
2:0	Pair B Reflection #4 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.

10.6.31 Global Cable Diagnostic Impedance 3: Address 1E.C882

Bit	Name	Type	Default	Description
F	Reserved 9	RO		Reserved
E:C	Pair C Reflection #1 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.
B	Reserved 10	RO		Reserved
A:8	Pair C Reflection #2 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.
7	Reserved 11	RO		Reserved
6:4	Pair C Reflection #3 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.
3	Reserved 12	RO		Reserved
2:0	Pair C Reflection #4 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.



10.6.32 Global Cable Diagnostic Impedance 4: Address 1E.C883

Bit	Name	Type	Default	Description
F	Reserved 13	RO		Reserved
E:C	Pair D Reflection #1 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.
B	Reserved 14	RO		Reserved
A:8	Pair D Reflection #2 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.
7	Reserved 15	RO		Reserved
6:4	Pair D Reflection #3 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.
3	Reserved 16	RO		Reserved
2:0	Pair D Reflection #4 [2:0]	RO		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. 111b = Open circuit (> 300 Ω). 110b = High mismatch (> 115 Ω). 101b = Low mismatch (< 85 Ω). 100b = Short circuit (< 30 Ω). 0xxb = No information available.

10.6.33 Global Status: Address 1E.C884

Bit	Name	Type	Default	Description
F:8	Reserved Status 0 [7:0]	RO		Reserved
7:0	Cable Length [7:0]	RO		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.

10.6.34 Global Reserved Status 1: Address 1E.C885

Bit	Name	Type	Default	Description
F:A	Nearly Seconds MSW [5:0]	RO		Bits 21:16 of the 22-bit nearly seconds uptime counter. The nearly seconds counter is incremented every 1024 ms.
9:8	NVM Status [1:0]	ROS PD	0x0	This register indicates the status of the last NVM operation. Status of NVM: 0x0 = NVM not enabled. 0x1 = Last NVM operation succeeded. 0x2 = Last NVM operation failed. 0x3 = Reserved.
7:0	ROM Revision [7:0]	ROS PD	0x00	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.

10.6.35 Global Reserved Status 2: Address 1E.C886

Bit	Name	Type	Default	Description
F:0	Nearly Seconds LSW [F:0]	RO		Bit 15:0 of the 22-bit nearly seconds uptime counter. The nearly seconds counter is incremented every 1024 ms.

10.6.36 Global Alarms 1: Address 1E.CC00

Bit	Name	Type	Default	Description
FA:7	Reserved			RW - Reserved, do not modify.
6	Reset completed	LH		This bit is set by the µP completing it's initialization sequence. 1b = PHY init sequence completed.
5	Reserved			RW - Reserved, do not modify.
4	Device Fault	LH		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. 1b = Fault.
3	Reserved Alarm A	LH		Reserved.
2	Reserved Alarm B	LH		Reserved.
1	Reserved Alarm C	LH		Reserved.
0	Reserved Alarm D	LH		Reserved.

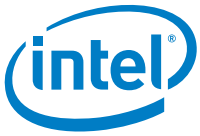
10.6.37 Global Alarms 2: Address 1E.CC01

Bit	Name	Type	Default	Description
F:D	Reserved			RW - Reserved, do not modify.
C:1	Reserved Alarms [B:0]	LH		Reserved
C:6	Reserved Alarms [6:0]	LH		Reserved
5	MAC Reset Request Handled	LH		1b = PHY handled the MAC reset request.
4	MAC PHY Disable Request Handled	LH		1b = PHY handled the MAC disable request.
3	MAC PHY Low Power Request Handled	LH		1b = PHY handled the MAC low power request.
2	MAC PHY Normal State Request Handled	LH		1b = PHY handled the MAC normal state request.

Bit	Name	Type	Default	Description
1	MAC PHY Clear Reset Request Handled	LH		1b = PHY handled the MAC clear reset request.
0	Diagnostic Alarm	LH		A diagnostic alarm used to test system alarm circuitry. 1b = Alarm triggered by a write to 1E.C470.7.

10.6.38 Global Alarms 3: Address 1E.CC02

Bit	Name	Type	Default	Description
F	NVM Operation Complete	LH		NVM interface is ready interrupt for registers 1b = NVM operation is complete
E	Mailbox Operation: Complete	LH		Mailbox interface is ready interrupt for registers 1b = Mailbox operation is complete
D:B	Reserved			RW - Reserved, do not modify.
A	uP DRAM Parity Error	LH		1b = Parity error detected in the uP DRAM
9:8	uP IRAM Parity Error [1:0]	LH		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
7:6	Reserved			RW - Reserved, do not modify.
5	Tx Enable State Change	LRF		1b = TX_EN pin has changed state
4:3	Reserved			RW - Reserved, do not modify.
2	MDIO MMD Error	LH		1 = Invalid MMD address detected
1	MDIO Timeout Error	LH		1 = MDIO timeout detected
0	Watchdog Timer Alarm	LH		1 = Watchdog timer alarm



10.6.39 Global Interrupt Mask 1: Address 1E.D400

Bit	Name	Type	Default	Description
A:7	Reserved			RW - Reserved, do not modify.
6	Reset completed Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
5	Reserved			RW - Reserved, do not modify.
4	Device Fault Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
3	Reserved Alarm A Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
2	Reserved Alarm B Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
1	Reserved Alarm C Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
0	Reserved Alarm D Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.

10.6.40 Global Interrupt Mask 2: Address 1E.D401

Bit	Name	Type	Default	Description
F:D	Reserved			RW - Reserved, do not modify.
C:1	Reserved Alarms [B:0]	LH		Reserved.
C:6	Reserved Alarms [6:0]	LH		Reserved.
5	MAC Reset Request Handled Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
4	MAC PHY Disable Request Handled Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
3	MAC PHY Low Power Request Handled Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.

Bit	Name	Type	Default	Description
2	MAC PHY Normal State Request Handled Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
1	MAC PHY Clear Reset Request Handled Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
0	Diagnostic Alarm Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.

10.6.41 Global LA SI Interrupt Mask 3: Address 1E.D402

Bit	Name	Type	Default	Description
F	NVM Operation Complete Mask	R/W PD	0b	NVM interface is ready interrupt for registers 1b = Enable interrupt generation 0b = Disable interrupt generation
E	Mailbox Operation: Complete Mask	R/W PD	0b	Mailbox interface is ready interrupt for registers 1b = Enable interrupt generation 0b = Disable interrupt generation
D:B	Reserved			RW - Reserved, do not modify.
A	uP DRAM Parity Error Mask	R/W PD	0b	1b = Enable interrupt generation 0b = Disable interrupt generation
9:8	uP IRAM Parity Error [1:0] Mask	R/W PD	00b	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 = Enable interrupt generation 0b = Disable interrupt generation
7:6	Reserved			RW - Reserved, do not modify.
5	Tx Enable State Change Mask	R/W PD	0b	1b = Enable interrupt generation 0b = Disable interrupt generation
4:3	Reserved			RW - Reserved, do not modify.
2	MDIO MMD Error Mask	R/W PD	0b	1b = Enable interrupt generation 0b = Disable interrupt generation

Bit	Name	Type	Default	Description
1	MDIO Timeout Error Mask	R/W PD	0b	1b = Enable interrupt generation 0b = Disable interrupt generation
0	Watchdog Timer Alarm Mask	R/W PD	1b	1b = Enable interrupt generation 0b = Disable interrupt generation

10.6.42 Global Chip-Wide Standard Interrupt Flags: Address 1E.FC00

Bit	Name	Type	Default	Description
F	PMA Standard Alarm 1 Interrupt	RO		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 LASI Interrupt Mask 1: Address 1.F000. 1b = Interrupt in PMA standard alarms 1.
E	PMA Standard Alarm 2 Interrupt	RO		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 LASI Interrupt Mask 2: Address 1.F001. 1b = Interrupt in PMA standard alarms 2.
D	PCS Standard Alarm 1 Interrupt	RO		An interrupt was generated from status register PCS Standard Status 1: Address 3.1 and the corresponding mask register PCS Receive Standard LASI Interrupt Mask 1: Address 3.F000. 1b = Interrupt in PCS standard alarms 1.
C	PCS Standard Alarm 2 Interrupt	RO		An interrupt was generated from status register PCS Standard Status 2: Address 3.8 and the corresponding mask register PCS Receive Standard LASI Interrupt Mask 2: Address 3.F001. 1b = Interrupt in PCS standard alarms 2.
B	PCS Standard Alarm 3 Interrupt	RO		An interrupt was generated from status register PCS 10GBASE-T Status 2: Address 3.21 and the corresponding mask register PCS Receive Standard LASI Interrupt Mask 3: Address 3.F002. 1b = Interrupt in PCS standard alarms 3.
A	PHY XS Standard Alarms 1 Interrupt	RO		An interrupt was generated from the status register PHY XS Standard Status 1: Address 4.1 and the corresponding mask register PHY XS Transmit (XAUI Rx) Standard LASI Interrupt Mask 1: Address 4.D000. 1b = Interrupt in PHY XS standard alarms 1.
9	PHY XS Standard Alarms 2 Interrupt	RO		An interrupt was generated from the status register PHY XS Standard Status 2: Address 4.8 and the corresponding mask register PHY XS Transmit (XAUI Rx) Standard LASI Interrupt Mask 2: Address 4.D001. 1b = Interrupt in PHY XS standard alarms 2.
8	Auto-Negotiation Standard Alarms 1 Interrupt	RO		An interrupt was generated from status register Auto-Negotiation Standard Status 1: Address 7.1 and the corresponding mask register Auto-Negotiation Standard LASI Interrupt Mask 1: Address 7.D000. 1b = Interrupt in auto-negotiation standard alarms 1.

Bit	Name	Type	Default	Description
7	Auto-Negotiation Standard Alarms 2 Interrupt	RO		An interrupt was generated from status register Auto-Negotiation 10GBASE-T Status Register: Address 7.21 and the corresponding mask register Auto-Negotiation Standard LASI Interrupt Mask 2: Address 7.D001. 1b = Interrupt in auto-negotiation standard alarms 2.
6	GbE Standard Alarms Interrupt	RO		An interrupt was generated from the TGE core. 1b = Interrupt in GbE standard alarms.
5:1	Reserved			RW - Reserved, do not modify.
0	All Vendor Alarms Interrupt	RO		An interrupt was generated from status register Global LASI Vendor Interrupt Flags: Address 1E.FC01 and the corresponding mask register. Global Interrupt PHY Vendor Mask: Address 1E.FF01. 1b = Interrupt in all vendor alarms.

10.6.43 Global Chip-Wide Vendor Interrupt Flags: Address 1E.FC01

Bit	Name	Type	Default	Description
F	PMA Vendor Alarm Interrupt	RO		A PMA alarm was generated. (see PMA Vendor Global LASI Interrupt Flags 1: Address 1.FC00). 1b = Interrupt in PMA vendor specific alarm.
E	PCS Vendor Alarm Interrupt	RO		A PCS alarm was generated. (See PCS Vendor Global LASI Interrupt Flags 1: Address 3.FC00). 1b = Interrupt in PCS vendor specific alarm.
D	PHY XS Vendor Alarm Interrupt	RO		A PHY XS alarm was generated (see "PHY XS Vendor Global LASI Interrupt Flags 1: Address 4.FC00" on page 135.) 1b = Interrupt in PHY XS vendor specific alarm.
C	Auto-Negotiation Vendor Alarm Interrupt	RO		An auto-negotiation alarm was generated (see Auto-Negotiation Vendor Global LASI Interrupt Flags 1: Address 7.FC00). 1b = Interrupt in auto-negotiation vendor specific alarm.
B	GbE Vendor Alarm Interrupt	RO		A GbE alarm was generated (see GbE PHY Vendor Global LASI Interrupt Flags 1: Address 1D.FC00). 1b = Interrupt in GbE vendor specific alarm.
A:3	Reserved			RW - Reserved, do not modify.
2	Global Alarms 1 Interrupt	RO		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.

Bit	Name	Type	Default	Description
1	Global Alarms 2 Interrupt	RO		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.
0	Global Alarms 3 Interrupt	RO		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.

10.6.44 Global Interrupt Chip-Wide Standard Mask: Address 1E.FF00

Bit	Name	Type	Default	Description
F	PMA Standard Alarm 1 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
E	PMA Standard Alarm 2 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
D	PCS Standard Alarm 1 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
C	PCS Standard Alarm 2 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
B	PCS Standard Alarm 3 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
A	PHY XS Standard Alarms 1 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
9	PHY XS Standard Alarms 2 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
8	Auto-Negotiation Standard Alarms 1 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
7	Auto-Negotiation Standard Alarms 2 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.

Bit	Name	Type	Default	Description
6	Gbe Standard Alarms Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
5:1	Reserved			RW - Reserved, do not modify.
0	All Vendor Alarms Interrupt Mask	R/W PD	1b	1b = Enable interrupt generation. 0b = Disable interrupt generation.

10.6.45 Global Interrupt Chip-Wide Vendor Mask: Address 1E.FF01

Bit	Name	Type	Default	Description
F	PMA Vendor Alarm Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
E	PCS Vendor Alarm Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
D	PHY XS Vendor Alarm Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
C	Auto-Negotiation Vendor Alarm Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
B	GbE Vendor Alarm Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
A:3	Reserved			RW - Reserved, do not modify.
2	Global Alarms 1 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
1	Global Alarms 2 Interrupt Mask	R/W PD	0b	1b = Enable interrupt generation. 0b = Disable interrupt generation.
0	Global Alarms 3 Interrupt Mask	R/W PD	1b	1b = Enable interrupt generation. 0b = Disable interrupt generation.

11.0 Manageability

11.1 Platform Configurations

This section describes the hardware configurations for platform management. It describes the partitioning of platform manageability among system components and the functionality provided by the X540 in each of the platform configurations.

The X540 supports pass-through manageability to an on-board BMC. The link between the X540 and the BMC is either SMBus or NC-SI.

Note: BMC is synonymous with Manageability Controller (MC). Both are used interchangeably throughout this section.

11.1.1 On-Board BMC Configurations

Figure 11-1 (left option) depicts an SMBus-only connection between the X540 and the BMC. The SMBus is used for all communication between the X540 and the BMC (pass-through traffic, configuration, and status). The protocol details for this configuration follow the SMBus commands. Figure 11-1 (right option) depicts an NC-SI-only connection between the X540 and the BMC. The NC-SI is used for all communication between the X540 and the BMC (pass-through traffic, configuration, and status). The protocol details for this configuration follow the NC-SI protocol.

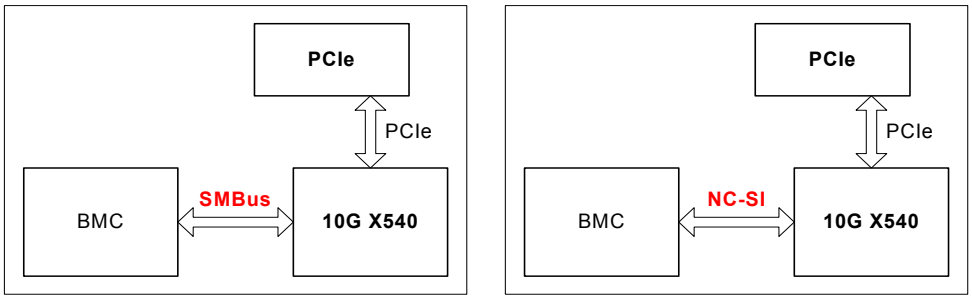


Figure 11-1 X540 to BMC Connectivity Through SMBus Link or NC-SI Link

Refer to the sections that follow for a description of the traffic types that use the NC-SI and/or SMBus interfaces.

11.1.2 X540 NIC

BMC connection to a NIC is not expected.

11.2 Pass-Through (PT) Functionality

The X540 supports traffic pass-through to an external BMC. The Pass-through traffic is carried through an NC-SI interface or SMBus (legacy devices) based on the redirection sideband interface setting in the NVM (loaded on power up). The usable bandwidth for either direction is up to 100 Mb/s in NC-SI mode and up to 400Kb/s in SMBus mode. Supplemental description on SMBus and NC-SI interfaces can be found in Section 3.2 and in Section 3.3. The following list describes usage models for the pass-through traffic:

- BMC management traffic
- Keyboard or mouse traffic for KVM (low data rate)
- Video traffic for KVM (low average rate of 150 Kb/s to 200 Kb/s) — transmit only
- USB 2.0 redirect (up to 50 Mb/s)
- IDE redirect for remote CD/floppy (rate — priority 1 — CDx7 = 1.05 Mb/s. Priority 2 — CDx24 = 64 Mb/s
- Serial Over LAN (SoL) — 300 Kb/s

11.2.1 NC-SI Mode

The X540 supports all the mandatory features of the NC-SI spec rev 1.0.0.

11.2.1.1 Supported Features

Table 11-1 lists the commands supported by the X540.

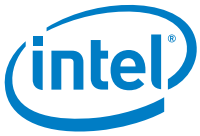


Table 11-1 Supported NC-SI / MCTP Commands

Command	Supported Over NC-SI	Supported Over MCTP
Clear Initial State	Yes	Yes
Get Version ID	Yes	Yes
Get Parameters	Yes	Yes
Get Controller Packet Statistics[1]	Yes, partially	Yes, partially
Get Link Status	Yes	Yes
Enable Channel	Yes	Yes[2]
Disable Channel	Yes	Yes[2]
Reset Channel	Yes	Yes[2]
Enable VLAN	Yes (filtering only by the VLAN ID. No filtering by the user priority)	No[2]
Disable VLAN	Yes	No[2]
Enable BCast	Yes	No[2]
Disable BCast	Yes	No[2]
Set MAC Address	Yes	No[2]
Get NC-SI Statistics	Yes, partially	Yes, partially
Set NC-SI Flow-Control	No	No
Set Link Command	Yes	Yes
Enable Global multicast Filter	Yes	No[2]
Disable Global multicast Filter	Yes	No[2]
Get Capabilities	Yes	Yes
Set VLAN Filters	Yes	No[2]
AEN Enable	Yes	Yes

Table 11-1 Supported NC-SI / MCTP Commands

Command	Supported Over NC-SI	Supported Over MCTP
Get Pass-Through Statistics	Yes, partially	No[2]
Select Package	Yes	Yes
Deselect Package	Yes	Yes
Enable Channel Network TX	Yes	No
Disable Channel Network TX	Yes	No
OEM Command	Yes	Yes

1. Only statistics 2 through 8 and 13 through 16 are supported.
2. In MCTP over SMBus mode, only control commands are supported and not pass-through traffic. As a result, 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).

Table 11-2 lists the NC-SI features supported by the X540:

Table 11-2 Optional NC-SI Features Support

Feature	Supported	Details
AENs	Yes.	Note: The driver state AEN might be emitted up to 15 seconds after actual driver change.
Get NC-SI statistics command	Yes, partially	Supports the following counters: 1-4, 7
Get NC-SI pass-through statistics command	Yes, partially	Supports the following counters: 2 Supports the following counters only when the operating system is down: 1, 6, 7
Get Controller Packet Statistics command	Yes, partially	Supports the following counters: 2 to 8 and 13 to 16[1].
VLAN modes	Yes, partially	Supports only modes 1,3
Buffering capabilities	Yes	8K
Ethernet MAC address filters	Yes	Supports 2 Ethernet MAC addresses as mixed per port
Channel count	Yes	Supports 2 channels
VLAN filters	Yes	Supports 8 VLAN filters per port
Broadcast filters	Yes	Supports the following filters: • ARP • DHCP • NetBIOS

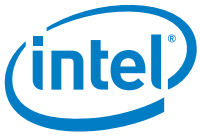


Table 11-2 Optional NC-SI Features Support

Feature	Supported	Details
Multicast filters	Yes	Supports the following filters (supported only when all three are enabled): • IPv6 neighbor advertisement • IPv6 router advertisement • DHCPv6 relay and server multicast
NC-SI flow control command	No	
Hardware arbitration	Yes	

1. As described in the Get Controller Packet Statistics Counter Numbers table in the NC-SI specification.

11.2.1.2 Set Link Command Error Codes

The MC can use the NC-SI Set Link command to control the external interface link settings. This command enables the MC to set the auto-negotiation, link speed, duplex, and other parameters.

The following flow is performed by the device firmware to complete the command to the MC:

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 10G on a 1G device will succeed 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 will accept a command with both FD and HD set), and also for HD being requested with speeds of 1G and higher as long as a speed below 1G is also requested (and is supported in HD). The device will simply ignore the unsupported parameters.

   b. Force Mode Parameter Validation:
      - If more than one link speed is being forced, then return a Command Specific Response (0x9) with a Set Link Speed Conflict Reason (0x5).
      - If 1G and above is requested with HD, then return a General Reason Code for a failed command (0x1) with Parameter Is Invalid, Unsupported, or Out-of-Range Reason (0x2).
4. Media Type Compatibility Check: If current media type is not compatible for the requested link parameters, return a Command Specific Response (0x9) with Set Link Media Conflict Reason (0x2).

5. Power State Compatibility Check: If current power state does not allow for the requested link parameters, return a Command Specific Response (0x9) with Set Link Power Mode Conflict Reason (0x4).

Note: If for some reason the device cannot perform the flow required for the command, return a Command Specific Response (0x9) with Link Command Failed-Hardware Access Error (0x6).

11.2.1.3 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, proxy or wakeup). 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 All PHYs in D3 N bit.

11.2.2 SMBus Pass-Through (PT) Functionality

When operating in SMBus mode, the X540 provides the following manageability services to the BMC on top of the pass-through traffic functionality:

- ARP handling — The X540 can be programmed to auto-ARP replying for ARP request packets and sending gratuitous ARP to reduce the traffic over the SMBus.
- 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 (Section 6.5.3) and flex TCO NVM structure (Section 6.5.4.11).

11.2.2.1 Pass-Through (PT) Modes

PT configuration depends on how the LAN ports are configured. Same as in the LAN channels, there is no logical connection between the two devices. The fail-over between the two LAN ports are done by the external BMC (by sending/receiving packets through different devices). The status reports to the BMC, ARP handling, DHCP and other pass-through functionality are unique for each port.

In pass-through mode most of the manageability traffic is handled by the BMC. However, portion of the network traffic can be offloaded and by the X540 as described in the following sub-sections. This configuration can be done by issuing configuration

commands over the SMBus channel or the X540 can load it from its NVM at power up (or both).

11.2.2.2 ARP Handling

Independent of the management interface, the X540 can be programmed by the BMC to provide ARP services. The X540 supports auto-ARP replying for ARP request packets and sending Gratuitous ARP. Auto-ARP is done in both ports.Each channel uses its own IP and Ethernet MAC address (either the operating system Ethernet MAC address or independent addresses). The following ARP parameters are loaded from the NVM on power up or configured through the management interface:

- ARP auto-reply enabled
- ARP IP address (to filter ARP packets)
- ARP Ethernet MAC Addresses (for ARP response)

When an ARP request packet is received on the wire and ARP auto-reply is enabled, the X540 checks the targeted IP address (after the packet has passed L2 checks and ARP checks). If the targeted IP matches the X540 IP configuration, then it replies with an ARP response. The X540 responds to the ARP request targeted to the ARP IP address with its ARP Ethernet MAC address. In a case where there is no match, the X540 silently discards the packets. If the X540 is not configured to do auto-ARP response, it forwards the ARP packets to the BMC. See Section 16.1 for ARP request and ARP response packet formats.

When the external BMC uses the same IP and MAC of the operating system, the ARP operation should be coordinated with the operating system operation. In this mode, the external BMC has the responsibility and ARP auto-reply should be disabled.

Note: When configured in NC-SI mode, the X540 does not provide ARP services. All ARP handling is done by the BMC.

11.3 Manageability Receive Filtering

11.3.1 Overview and General Structure

For completeness, this section summarizes the MAC and VLAN filters described in Section 7.1.1.1 and Section 7.1.1.2. In addition, this section describes the manageability receive packet filtering flow. The description applies to any of the X540 LAN ports. Receive packet filtering can have one of the following routing results:

- Discard packets (packets that do not pass the host nor manageability filtering)
- Send packets to host memory (default hardware setting)
- Send packets to the external BMC (two modes):
  - Receive All — All received packets are routed to the BMC in this mode. It is enabled by setting the RCV_TCO_EN bit (which enables packets to be routed to the BMC) and RCV_ALL bit (which routes all packets to the BMC) in the MANC register.

  - — Receive Filtering — In this mode only some of the packet types are directed to the manageability block. The BMC should set the RCV_TCO_EN bit together with the required packet types bits in the manageability filtering registers. Note that the RCV_ALL bit must be cleared).
- Send packets to both the external BMC and host memory:
  - — The BMC can enable this mode by setting the EN_MNG2HOST bit in the MANC register and enable specific packet types in the MANC2H register.

The BMC controls its packet filtering by programming the receive manageability filters listed in the following table. These registers are not write-accessible by the host (protecting the BMC from erroneous/malicious host software).

Register	Functionality	Init	Loaded from the NVM
MANC	General configuration of the manageability filters	Power up and firmware initialization	Section 6.5.3.17
MANC2H	Enables routing of manageability packets to host	Power up and firmware initialization	Section 6.5.3.21
MFVAL	Valid entries in the manageability filter registers	Power up and firmware initialization	Section 6.5.3.15
MDEF[7:0], MDEF_EXT[7:0]	Configuration of manageability decision filters	Power up and firmware initialization	Section 6.5.3.23
MMAH[3:0], MMAL[3:0]	4 unicast MAC manageability addresses	Power up	Section 6.5.3.9
MAVTV[7:0]	8 VLAN tag values	Power up	Section 6.5.3.14
MFUTP[7:0]	16 destination port values	Power up	Section 6.5.3.13
FTFT_FILTER	4 flex TCO filters	Power up	Section 6.5.5
MIPAF	IP address for manageability filtering	Power up	Section 6.5.3.2
METF	L2 EtherType values	Power up	Section 6.5.3.28

Manageability filtering follows these steps and are detailed in the following sections:

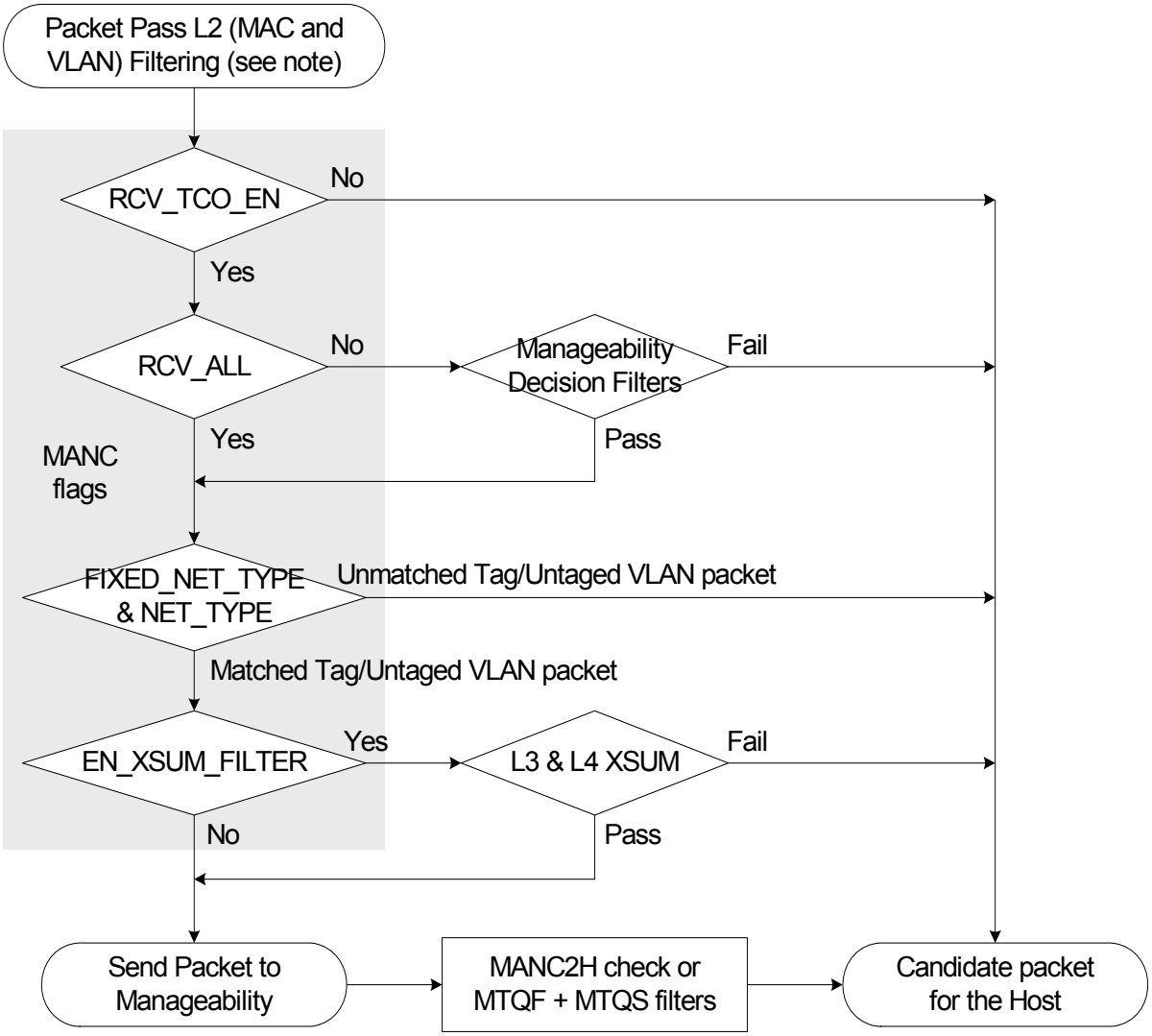
1. L2 Ethernet MAC Address and VLAN filtering
2. L3/L4 manageability filters — Port, IP, flex filters (packets must also match the above L2 filtering).

Filtering exceptions:

- Fragmented packets can be routed to manageability but not parsed beyond the IP header.
- Packets with L2 errors (CRC, alignment, etc.) are never forwarded to manageability.
- Jumbo packets above 2 KB are not expected to be received by the manageability data path.

Note: If the manageability unit uses dedicated Ethernet MAC address/VLAN tag, it should not use further L3/L4 filters on top of it. Otherwise, packets that match the L2 filters but fail the L3/L4 filters are routed to the host.

The complete filtering flow is described in the following flow diagram:



Note: L2 MAC address and VLAN filtering are described in Section 7.1.1.1 and Section 7.1.1.2.

11.3.2 L2 EtherType Filters

Packets are compared against the EtherType filters programmed in the METF.EType (up to 4 filters) and the result is incorporated to the decision filters.

Each of the manageability EtherType filters can be configured as pass ("positive") or reject ("negative") polarity. When negative polarity filters are used, all negative filters should be included in all enabled decision filters.

Examples for usages of the L2 EtherType filters are:

- Block routing of packets with the NC-SI EtherType from being routed to the BMC. The NC-SI EtherType is used for communications between the BMC on the NC-SI link and the X540. Packets coming from the network are not expected to carry this EtherType and such packets are blocked to prevent attacks on the BMC.
- Determine the destination of 802.1X control packets. The 802.1X protocol is executed at different times in either the BMC or by the host. The L2 EtherType filters are used to route these packets to the proper agent.
- In order to get meaningful filtering of Ethertype packets, negative filters should be in the AND section. If more than one positive Ethertype filter is needed, then they should be set in the OR section. A single positive Ethertype filter can be enabled both in the AND or in the OR section.

11.3.3 VLAN Filtering - Single and Double VLAN Cases

the X540 supports eight VLAN filters per port defined by the MAVTV[n] and controlled by the MANC register as described in the text that follows.

- When MANC.NET_TYPE = 1 and MANC.FIXED_NET_TYPE = 1b (pass only VLAN tagged packets)
  - A packet without any VLAN or a single VLAN header is not routed to manageability
  - A packet with 2 VLANs is a candidate for manageability
- When MANC.NET_TYPE = 0b and MANC.FIXED_NET_TYPE = 1b (pass only un-tagged packets)
  - A packet without any VLAN or a single VLAN header is a candidate for manageability
  - A packet with 2 VLANs is not routed to manageability
- When MANC.FIXED_NET_TYPE = 0b (both tagged and untagged packets are candidates for manageability)
  - A packet with no VLAN header skips successfully to the next filtering level
  - A packet with a single VLAN or 2 VLANs are filtered by its VLAN header as described in Section 7.1.1.2

11.3.4 L3 and L4 Filters

11.3.4.1 ARP Filtering

The X540 supports filtering of both ARP request packets (initiated externally) and ARP responses (to requests initiated by the BMC or the X540).

11.3.4.2 Neighbor Discovery Filtering

The X540 supports filtering of neighbor discovery packets. Neighbor discovery filters use the IPV6 destination address filters defined in the MIPAF registers (such as match to any of the enabled IPv6 addresses).

11.3.4.3 Port 0x298/0x26F Filtering

The X540 supports filtering by fixed destination ports numbers: 0x26F and 0x298.

11.3.4.4 Flex Port Filtering

The X540 implements 16 flex destination port filters. The X540 directs packets whose L4 destination port matches the value of the respective word in the MFUTP registers. The BMC must ensure that only valid entries are enabled in the decision filters that follow.

11.3.4.5 Flex TCO Filters

See Section 11.3.5 below.

11.3.4.6 IP Address Filtering

The X540 supports filtering by IP address through dedicated IPv4 and IPv6 address filters to manageability. Two modes are possible, depending on the value of the MANC.EN_IPv4_FILTER bit:

- EN_IPv4_FILTER = 0b: The X540 provides four IPv6 address filters.
- EN_IPv4_FILTER = 1b: The X540 provides three IPv6 address filters and four IPv4 address filters.
- The MFVAL register indicates which of the IP address filters are valid (contains a valid entry and should be used for comparison).

11.3.4.7 Checksum Filter

If bit MANC.EN_XSUM_FILTER is set, the X540 directs packets to the BMC only if they match all other filters previously described as well as pass L3/L4 checksum (if it exists).

11.3.5 Flexible 128 Bytes Filter (TCO Filters)

11.3.5.1 Overview

The flexible 128 filters are a set of filters designed to enable dynamic filtering of received packets. These filters are part of the manageability receive filters. The filters do not make a decision on the packet's destination. They participate in the decision mechanism for each received packet (Section 11.3.6).

Each filter enables a flexible testing of the first 128 bytes of the packet against a given value. The filter also enables testing of specific bytes by defining a byte-wise mask on the filter.

The X540 provides four flex TCO filters. Each filter looks for a pattern match within the 1st 128 bytes of the packet. The BMC must ensure that only valid entries are enabled in the decision filters.

Note: The flex filters are temporarily disabled when read or written by the host. Any packet received during a read or write operation is dropped. Filter operation resumes once the read or write access completes.

11.3.5.2 Structure

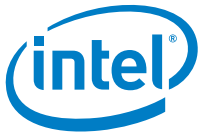
Each filter is composed of the following fields:

1. Flexible Filter Length: This field indicates the number of bytes in the packet header that should be inspected. This field also indicates the minimal length of packets in order to be inspected by the filter. A packet below that length is not inspected by the filter. Valid values for this field are: 8*n, where n=1…8.
2. Data: This is a set of up to 128 bytes comprising the values that the header bytes of each packet are tested against.
3. Mask: This is a set of 128 bits corresponding to the 128 data bytes that indicate for each corresponding byte if is tested against its corresponding byte.

Overall, each filter tests the first 128 bytes (or less) of a packet, where not necessarily all bytes must be tested.

Structure of the Flexible TCO 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

The following table depicts the addresses used for filter 0.

Field	Dword	Address	Bit(s)	Initial Value
Filter 0 DW0	0	0x09400	31:0	X
Filter 0 DW1	1	0x09404	31:0	X
Filter 0 Mask[7:0]	2	0x09408	7:0	X
Reserved	3	0x0940C		X
Filter 0 DW2	4	0x09410	31:0	X
…				
Filter 0 DW30	60	0x094F0	31:0	X
Filter 0 DW31	61	0x094F4	31:0	X
Field	**Dword**	**Address**	**Bit(s)**	**Initial Value**
Filter 0 Mask[127:120]	62	0x094F8	7:0	X
Length	63	0x094FC	6:0	X

11.3.5.2.1 Programming

Programming each filter is done using the following two commands (NC-SI or SMBus) in a sequential manner:

1. Filter Mask and Length. This command configures the following fields.
   a. Mask: A set of 16 bytes containing the 128 bits of the mask. Bit 0 of the first byte corresponds to the first byte on the wire.
   b. Length: A 1-byte field indicating the length.
2. Filter Data.

   The filter data is divided into groups of bytes. as follows:

Group	Test Bytes
0x0	0-29
0x1	30-59
0x2	60-89
0x3	90-119
0x4	120-127

Each group of bytes needs to be configured using a separate command, where the group number is given as a parameter.

The command has the following parameters:

a. Group number. A 1-byte field indicating the current group addressed.

b. Data bytes. Up to 30 bytes of test-bytes for the current group.

11.3.6 Manageability Decision Filters

The manageability decision filters are a set of eight filters with the same structure (MDEF[7:0] and MDEF_EXT[7:0]). The filtering rule for each decision filter is programmed by the BMC and defines which of the L2, VLAN, and manageability filters participate in the decision (host software can't modify their setting). A packet that passes at least one set of decision filters is directed to manageability and possibly to the host as well. The inputs to each decision filter are:

- Packet passed a valid management L2 unicast 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 discovery 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.

The structure of each of the decision filters is shown in Figure 11-2 . A boxed "x.y" number indicates that the input is conditioned on a mask bit "y" defined in register index "x", while x=0 denotes MDEF and x=1 denotes MDEF_EXT. The decision filter rules are as follows:

- Any bit set in the MDEF and MDEF_EXT registers enables its corresponding filter. Any filter that is not enabled in the MDEF and MDEF_EXT registers is ignored. If all bits in the MDEF and MDEF_EXT registers of a specific decision filter are cleared, it is disabled and ignored.

- All enabled AND filters must pass for the decision filter to match.
- If at least one OR filter is enabled, then at least one of the enabled OR filters must pass for the decision filter to match.

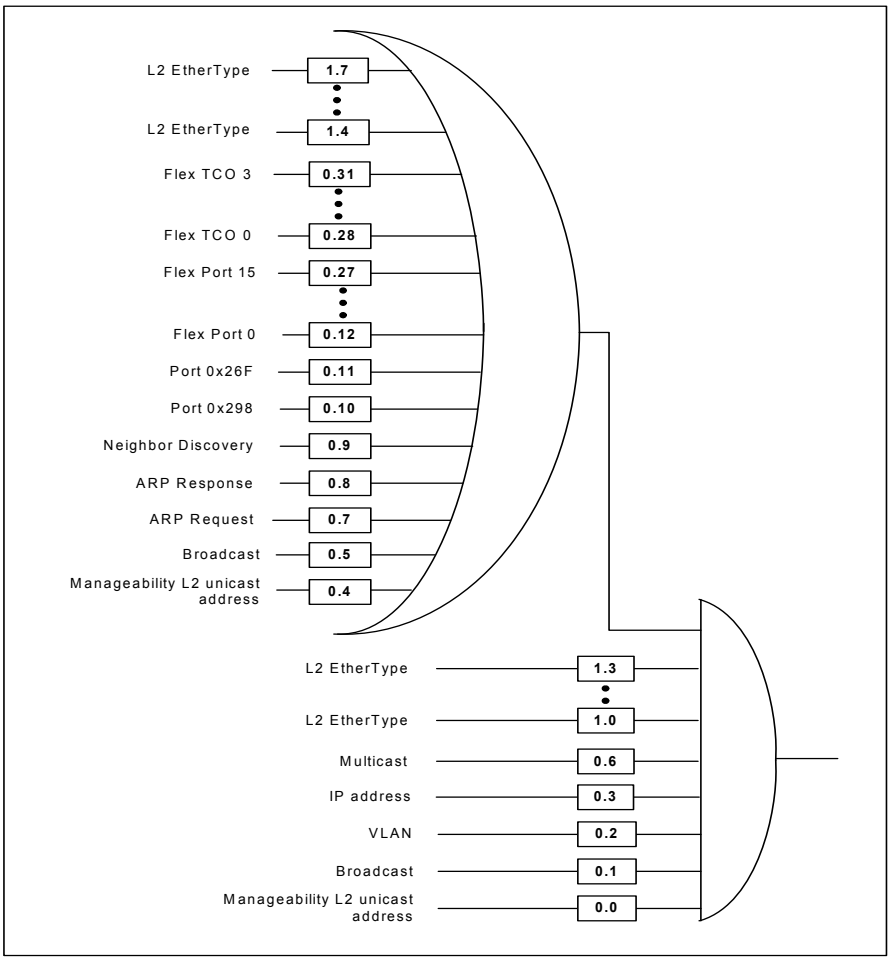


Figure 11-2 Manageability Decision Filters

11.3.7 Possible configurations

This section describes possible ways of using the management filters. Actual usage may vary.

Dedicated MAC packet filtering

- Select one of the eight rules for broadcast filtering
- Set bit 0 of the decision rule to enforce Ethernet MAC address filtering
- Set other bits to qualify which packets are allowed to pass through. For example,

  - Set bit 2 to qualify with manageability VLAN
  - Set bit 3 to qualify with a match to an IP address
  - Set any L3/L4 bits (30:7) to qualify with any of a set of L3/L4 filters

Broadcast packet filtering

- Select one of the eight rules for broadcast filtering
- Set bit 1 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 2 to qualify with manageability VLAN
  - Set bit 3 to qualify with a match to an IP address
  - Set any L3/L4 bits (30:7) to qualify with any of a set of L3/L4 filters

VLAN packet filtering

- Select one of the eight rules for VLAN filtering
- Set bit 2 of the decision rule to enforce VLAN filtering
- Set other bits to qualify which VLAN packets are allowed to pass through. For example:
  - Set any L3/L4 bits (30:7) to qualify with any of a set of L3/L4 filters

IPv6 filtering is done via the following IPv6-specific filters

- IP unicast filtering — requires filtering for link local address and a global address. Filtering setup might depend on whether an Ethernet MAC address is shared with the host or dedicated to manageability:
  - Dedicated Ethernet MAC address such as dynamic address allocation with DHCP does not support multiple IP addresses for one Ethernet MAC address). In this case, filtering can be done at L2 using two dedicated unicast MAC filters.
  - Shared Ethernet MAC Address such as static address allocation sharing addresses with the host). In this case, filtering needs to be done at L3, requiring two IPv6 address filters, one per address.
- A Neighbor discovery filter — The X540 supports IPv6 neighbor discovery protocol. Since the protocol relies on multicast packets, the X540 supports filtering of these packets. IPv6 multicast addresses are translated into corresponding Ethernet multicast addresses of the form 33-33-xx-xx-xx-xx, where the last 32 bits of the address are taken from the last 32 bits of the IPv6 multicast address. Therefore, two direct MAC filters can be used to filter IPv6 solicited-node multicast packets as well as IPv6 all node multicast packets.

Receive filtering with shared IP — CPMP

When the BMC shares the MAC and IP address with the host, receive filtering is based mainly on identifying specific flows through port allocation. The following setting can be used:

Select one of the eight rules:

- Set a manageability dedicated MAC filter to the host Ethernet MAC address and set bit 0 in the MNG_ FILTER_RULE register

- If VLAN is used for management, load one of more management VLAN filters and set bit 2 in the MNG_ FILTER_RULE register
- ARP filter / neighbor discovery filter is enabled when the BMC is responsible to handle the ARP protocol. Set bit 7 or bit 8 in the MNG_ FILTER_RULE register for this functionality
- Program flex port filters with the port values for management flows such as DHCP, HTTP, HTTPS, SMWG, SoL/IDER/KVM, WS-MAN, Telnet, USB redirection, SSH, DNS, and more. Set the respective bits 26:11 in the MNG_ FILTER_RULE register.
- An IP address filter can be loaded as well by setting bit 3 in the MNG_ FILTER_RULE register
- Management flex filters are programmed to correspond to remaining flows such as DNS update response packets. Set appropriate bits 30:27 in the MNG_ FILTER_RULE register

11.4 MACsec and Manageability

For details on MACsec and the role of manageability in it, see Section 7.8.

Pass-through mode is supported in a MACsec environment in one of the following modes of operations:

- Management traffic not protected by MACsec — The management traffic from and to the BMC is carried over a separate Ethernet MAC address and/or a separate VLAN and the network switch is configured to enable such traffic to pass unprotected.
- Management traffic is protected by MACsec — The X540 supports a single secure channel for both host and BMC. At a given time, the host and BMC can be active or inactive. When only BMC is active, it acts as the KaY controlling the secured channel. The host can act as the KaY when it is functional and after it acquires control over MACsec. In this case, the BMC uses the secured channel set by the host. Even when operating in this mode, the BMC can transmit packets on the clear (as required for 802.1x control packets). The BMC must disable MACsec operation before sending such packets and re-enable MACsec operation afterwards. The messages that control MACsec operation are described in Section 11.7.1.15.

The X540 provides the following functionality that enables management traffic over the same secure channel with the host:

- Handover of MACsec ownership between the BMC and the host. Several transitions in ownership are possible:
  - Power-on — The X540 powers up with MACsec **not** being owned by the BMC. If the BMC is configured for MACsec, it takes ownership over MACsec as follows. If the BMC is not configured for MACsec, the host takes ownership when it boots. If MACsec is not owned by the BMC, the host is not required for any handshake with the BMC as there are cases where the BMC is not connected to the X540. If there is a race between the BMC and the host, the BMC wins over MACsec, and the host is then interrupted so that the MACsec resources are not accessible.
  - Handover of MACsec responsibility from BMC to host — The host can initiate a transfer of ownership from the BMC (such as on operating system boot).

  - — Handover of MACsec responsibility from host to BMC — The host can initiate a transfer of ownership to the BMC (such as on entry to low power state). This is done through the host slave command interface.
  - — Forced handover of MACsec responsibility from host to BMC — The BMC can acquire ownership of MACsec on its own, for example when the host fails to acquire a secure channel. See Section 11.4.1 for the different transition sequences.
- Configuration of MACsec resources by the BMC — When the BMC owns the secure channel, it configures MACsec operation through the SMBus or NC-SI vendor-specific commands (see Section 11.7.1.15 and Section 11.7.2.1.7).
- Alerts — The X540 initiates an SMBus or NC-SI alert to the BMC on several MACsec events as follows (refer to alerts message format in Section 11.7.1.16, Section 11.7.2.2.3, and Section 11.7.2.2.8).
  - — Packet arrived with a MACsec error (no SA match, replay detection, or a bad MACsec signature).
  - — Key-exchange event — relevant on TX when the packet number counter reaches the exhaustion threshold as described in Section 7.8.5.1.
  - — Host request for MACsec ownership.
  - — Host request to relinquish MACsec ownership.
- Interrupt causes — The X540 issues a management interrupt to the host on the following MACsec events:
  - — Acknowledge of handover of MACsec responsibility from BMC to host.
  - — Forced handover of MACsec responsibility from host to BMC.

  The host might identify the ownership status by reading the *Operating System Status* field in the LSWFW register.

11.4.1 Handover of MACsec Responsibility Between BMC and Host

11.4.1.1 KaY Ownership Release by the Host

The following procedure is used by the host in order to release ownership of the MACsec capability. This procedure is usually done before an ordered shutdown of the host.

- The host should stop accessing the MACsec registers and set the *Release MACsec* bit in the LSWFW register.
- Setting the *Release MACsec* bit causes an interrupt to the firmware that is forwarded to the BMC.
- The BMC then takes ownership as described in Section 11.4.1.2.
- The host can then wait for an interrupt from the firmware indicating that the BMC took the KaY ownership.

11.4.1.2 KaY Ownership Takeover by BMC

As previously mentioned, the BMC can acquire ownership over MACsec either by ownership relinquish by the host or without any negotiation (such as on power-up and on a forced transition when the host failed to bring up a MACsec connection). The BMC acquires ownership of MACsec by taking the following actions:

- Locking access to MACsec resources to the host by setting the *Lock MACsec Logic* bit in the LSWFW register.
- Blocking host packets' transmission from the wire by setting the *Block Host Traffic* bit in the LSWFW register.
- Set the *OS Status* field in the LSWFW register to 1b indicating a BMC takeover of the MACsec logic.
- Issue a manageability event interrupt to the host.

11.4.1.3 KaY Ownership Request by the Host

The following procedure is used by the host in order to request ownership of the MACsec capability:

- The host should read the LSWFW.OS status field to check if the KaY is currently owned by the BMC.
- if KaY is owned by the BMC, then the host should set the *Request MACsec* bit in the LSWFW register prior to assuming responsibility over MACsec connection.
- Setting the *Request MACsec* bit causes an interrupt to the firmware that is forwarded to the BMC.
- The host should then wait for an interrupt from the firmware indicating that the BMC released the KaY ownership.
- Following the manageability interrupt, the host should check the *OS Status* and Lock *MACsec Logic* fields in the LSWFW register to make sure the BMC released the KaY ownership.

11.4.1.4 KaY Ownership Release by BMC

In order to release ownership of MACsec, the BMC should take the following actions:

- Disconnect the MACsec connection with the switch (such as EAP logoff).
- Clear the *Lock MACsec Logic* bit in the LSWFS register enabling the host setting of the MACsec registers.
- Clear the *OS Status* bit to 0b in the LSWFS register indicate a MACsec release.
- Issue a manageability event interrupt to the host.
- Poll the connection state to check if the MACsec channel was set by the host.

If the BMC decides to deny the release request, it silently ignores the request.

11.4.1.5 Control Registers

The complete set of manageability registers are described in the Management Filters Registers and Manageability Host Interface Registers sections. The following configuration fields are dedicated for manageability control over MACsec:

LSWFW Field	LSWFW Field Functionality
Block Host Traffic	Enables or disables host transmit traffic for this PCI function from going to the wire. Default is to enable.
OS Status	Set by firmware to indicate the status of the MACsec ownership: 0b = MACsec owned by host (default). 1b = MACsec owned by BMC.
MACsec Request	Bit used by host to request KaY ownership.
MACsec Release	Bit used by host to release KaY ownership.
Lock MACsec Logic	Serves two purposes. It indicates who owns MACsec (default value is host ownership). Second, it enables or disables host accesses to the MACsec registers. Default is to enable. The following registers are blocked: LSECTXCAP; LSECRXCAP; LSECTXCTRL; LSECRXCTRL; LSECTXSCL; LSECTXSCH; LSECTXSA; LSECTXPN0; LSECTXPN1; LSECTXKEY0 (4 registers); LSECTXKEY1 (4 registers); LSECRXSCL; LSECRXSCH; LSECRXSA (0 and 1); LSECRXSAPN (0 and 1); LSECRXKEY (4 registers / SA); LSECTXUT; LSECTXPKTE; LSECTXPKTP; LSECTXOCTE; LSECTXOCTP; LSECRXUTnS; LSECRXUTyS; LSECRXOCTE; LSECRXOCTP; LSECRXBAD; LSECRXNOSCInS; LSECRXNOSCIyS; LSECRXNOSCI; LSECRXDELAY; LSECRXLATE; LSECRXOK; LSECRXINVCK; LSECRXINVST; LSECRXNSAST; LSECRXNSA

11.5 OS2BMC Traffic

11.5.1 Overview

In current systems, 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 X540 enables the host and the local BMC communication via the regular pass-through interface, and thus enable 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 X540 examines these packets and it then decides if they should be forwarded to the BMC. On the inverse path, when the BMC sends a packet through the pass-through interface, the X540 checks if it should be forwarded to the network, the host, or both. Figure 11-3 describes the flow for OS2BMC traffic.

The OS2BMC flow can be enabled using the *OS2BMC enable* field for the relevant port in the OS2BMC configuration structure of the NVM.

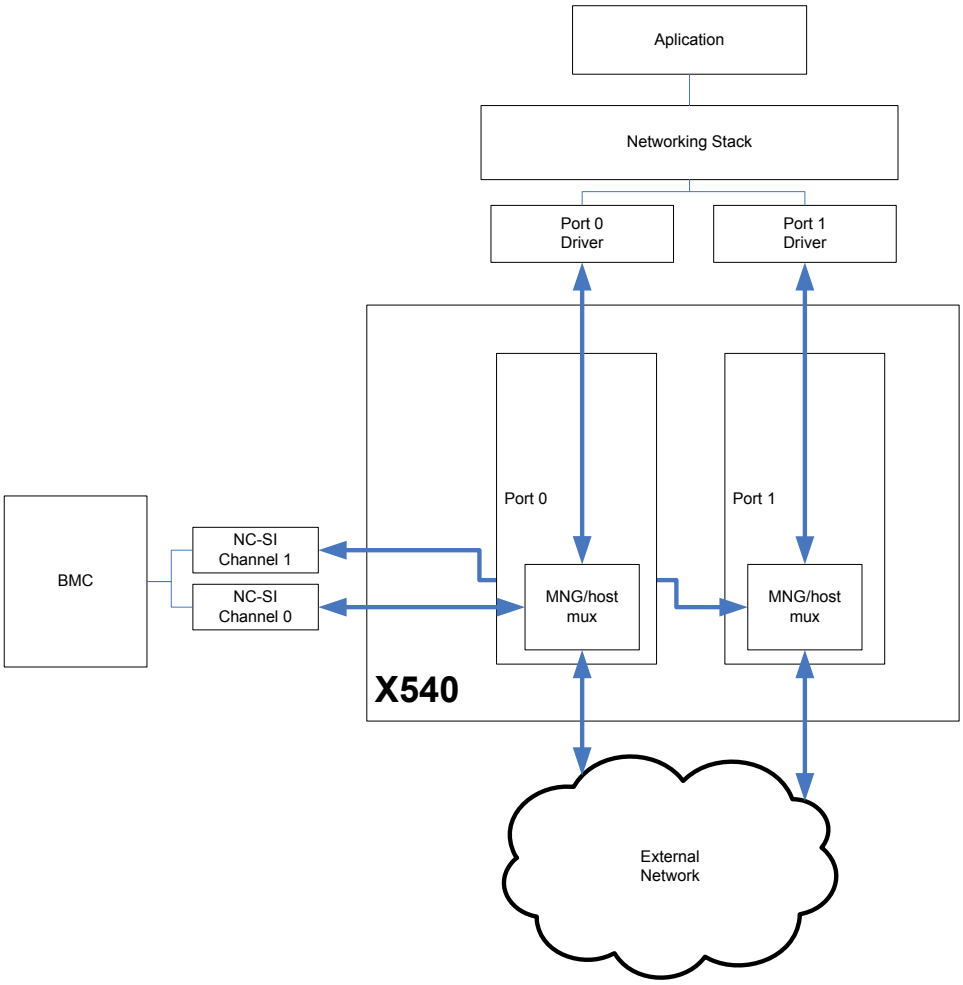


Figure 11-3 OS2BMC Diagram

Note: This flow assumes that the BMC does not share a MAC address with the host.

Note: The OS2BMC flow is enabled only for ports enabled by the NC-SI Enable Channel command or via the *OS to BMC Enable* field for the relevant port in the OS-to-BMC configuration structure of the NVM.

OS2BMC traffic must comply with NC-SI specifications and is therefore limited to maximum sized frames of 1536 bytes (in both directions).

Note: When the BMC uses IPSec for the flow directed to the host, it should make sure the X540 does not offload the IPSec flow by exposing the IP address used via the *Set IP address* NC-SI OEM command.

11.5.2 Filtering

11.5.2.1 OS2BMC Filtering

When OS2BMC traffic is enabled, the filters used for network-to-BMC traffic are also used for OS2BMC traffic. Traffic considered as exclusive to the BMC (relevant bit in MNG2HOST is cleared) is also considered as exclusive to the BMC when sent from the host and not forwarded to the network.

11.5.2.2 Handling of OS2BMC Packets

All the regular transmit offloads are also available for OS2BMC packets, apart from IPSec offload, which cannot be provided. See Section 7.12.2.2 for details.

11.5.2.3 BMC2OS Filtering

When OS2BMC is enabled, as with regular BMC transmit traffic, the port (operating system 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 7.1.1). According to the results of the filtering the packet can be forwarded to the operating system, 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.

The packet will be forwarded only according to the destination MAC address and VLAN tag.

When working in non VMDq modes (MRQC.Multiple Receive Queues Enable field != 011b), unicast packets that matches one of the exact filters (RAH/RAL) are sent only to the host. Other packets that passes the L2 host filtering will be sent to both the host and the network. Packets that do not pass the L2 filtering will be sent only to the network.

When working in VMDq mode (MRQC.Multiple Receive Queues Enable field =011b), if a packet passes L2 filtering, the forwarding decisions are based on the Tx switching algorithm described in Section 7.10.3.4. A packet that do not passes L2 filtering is sent to the network.

11.5.2.4 Queuing of Packets Received from the BMC

The traffic of the BMC to the host can be queued according to the following mechanisms:

1. VMDq — In this mode, the packets are queued according to the destination MAC address and VLAN. This mode is enabled by setting the *MRQC.Multiple Receive Queues Enable* field to 011b.

2. If the previous modes is not used, the packets are forwarded to queue 0.

11.5.2.5 Offloads of Packets Received from the BMC

Packets received from the BMC and forwarded to the operating system 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. Packets 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 RSS type and RSS hash are not calculated for BMC packets and are always set to zero.
- The header of BMC packets is never split.
- A fragmented BMC packet is not detected by the hardware.
- The BMC packets are not detected as time sync packets. The *RDESC.STATUS.TS* will always be clear for these packets.
- The L3 and L4 checksum are not performed on these packets. The *L4I*, *IPCS*, *UDPCS*, and *UDPV* fields will always be cleared 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.

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 can cause corruption to the packet received by the operating system.

- 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.

11.5.3 Blocking of Network-to-BMC Flow

In some systems the BMC might have its own private connection to the network and might use the X540 port only for the OS2BMC traffic. In this case, the BMC to network flow should be blocked while enabling the OS2BMC 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.

Note: 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.

Note: 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 might be enabled for receive, but all receive filters should be cleared.

11.5.4 OS2BMC and Flow Control

Traffic between the host and manageability is affected by two factors:

1. External link or priority flow control events received from the network.
2. Internal link or priority flow control events due to internal congestion on loopback path in VT switch.

Table 11-3 and Table 11-4 lists the effect of external/internal link and priority flow control events on the different traffic flows. In case of external/internal priority flow control (DCB mode only), only traffic associated to the paused traffic class is affected. Tx MNG traffic is associated to a specific traffic class according to setting made in MNGTXMAP.MAP field.

Table 11-3 Impact of External Flow Control

To/From	LAN	Host	MNG
LAN	N/A	No	No
Host	Yes	Yes[2]	Yes
MNG	Yes	Yes	N/A

Table 11-4 Impact of Internal Flow Control

To/From	LAN	HOST	MNG
LAN	N/A	No	No
HOST	Yes[1]	Yes[2]	Yes
MNG	Yes[1]	Yes	N/A

1. Only if first packet in the concerned Tx packet buffer is destined to VT switch and/or is a BC or MC packet.
2. VM-to-VM traffic via internal VT switch.

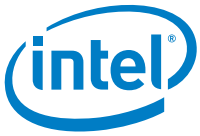
Notes: Host-to-MNG traffic is not regulated by an internal flow control mechanism. If the internal host-to-MNG buffer encounters congestion conditions, host-to-MNG packets are dropped.

Internal congestion on loopback path in VT switch always generates internal link or priority flow control, according to DCB mode (disabled or enabled, respectively), and regardless to the drop or no-drop policy of the link or of the traffic class over the LAN port.

11.5.5 Statistics

Packets sent from the operating system to the BMC should be counted by all statistical counters as packets sent by the operating system. If they are sent to both the network and to the BMC, then 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, then they are counted both as received packets and as packet transmitted to the network.



In addition the X540 supports the following statistical counters that measure just the BMC2OS and OS2BMC traffic:

- O2BGPTC - OS2BMC packets received by BMC
- O2BSPC - OS2BMC packets transmitted by the operating system
- B2OSPC - BMC2OS packets sent by BMC
- B2OGPRC - BMC2OS packets received by the operating system (counting each replication).

The driver can use these statistics to count packets dropped by the X540 during the transfer between the operating system and the BMC.

11.5.6 OS2BMC Enablement

The X540 supports one software model for OS2BMC traffic, the unified network model, where the OS2BMC traffic is shared with the regular traffic. In this model, there is no need for a special configuration of the operating system networking stack or the BMC stack, but if the link is down, then the OS2BMC communication is stopped.

Note: Some packets may still pass through the OS2BMC channel after a link down event, but the first packet to the LAN will block any traffic to the BMC also. In addition, as the OS networking stack and the BMC are notified of the link down condition, they will stop to send traffic.

In order to enable OS2BMC either:

- Enable OS2BMC in the port traffic type field in the traffic type parameters NVM word for the relevant port.
- Send an *EnableOS2BMC Flow* NC-SI OEM command

Note: When OS2BMC is enabled, the operating system must avoid sending packets longer than 9.5 KB to BMC.

If the OS2BMC enablement configuration needs to be changed on the fly (not as part of the system initialization procedure), designers should disable TCO receive via the Receive Enable command, set RCV_EN to 0b, and then wait 200 ms before changing the OS2BMC configuration. In addition, the BMC should not transmit during this period. OS2BMC enablement refers to commands described in

11.6 MCTP

11.6.1 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.

11.6.1.1 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 or to allow the BMC to control the network controller. 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.

Note: The X540's MCTP protocol implementation is based on an early draft of the DSP0261 Standard and it includes a *Payload Type* field that was removed in the final release of the standard.

11.6.1.2 MCTP Usage Model

The X540 supports NC-SI over MCTP protocol over SMBus. A X540 NIC can connect through MCTP to a BMC as described in Figure 11-1. The MCTP interface will be used by the BMC to control the NIC and not for pass-through traffic.

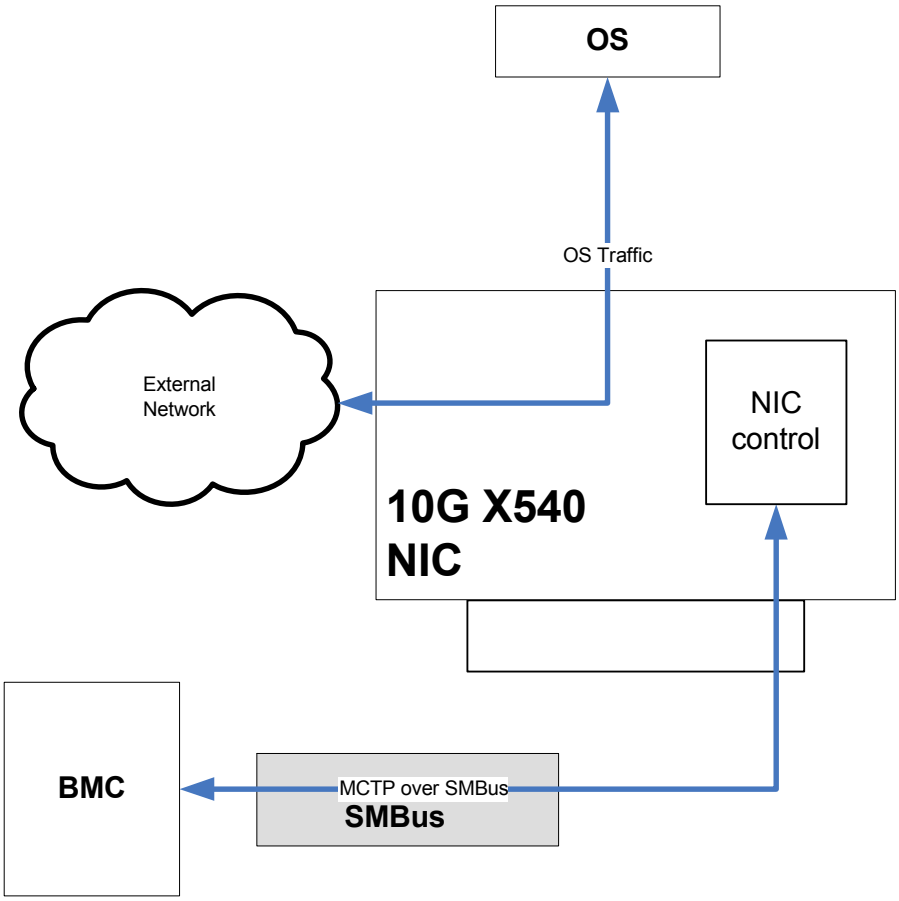


Figure 11-1. MCTP Connections of the X540

11.6.2 NC-SI to MCTP Mapping

The two network ports of the X540 (mapped to two NC-SI channels) are mapped to a single MCTP endpoint on SMBus.

The channels are not used for pass-through traffic and are only used to define which port is currently being accessed for control or status update.

The topology used for MCTP connection is described in Figure 11-2.

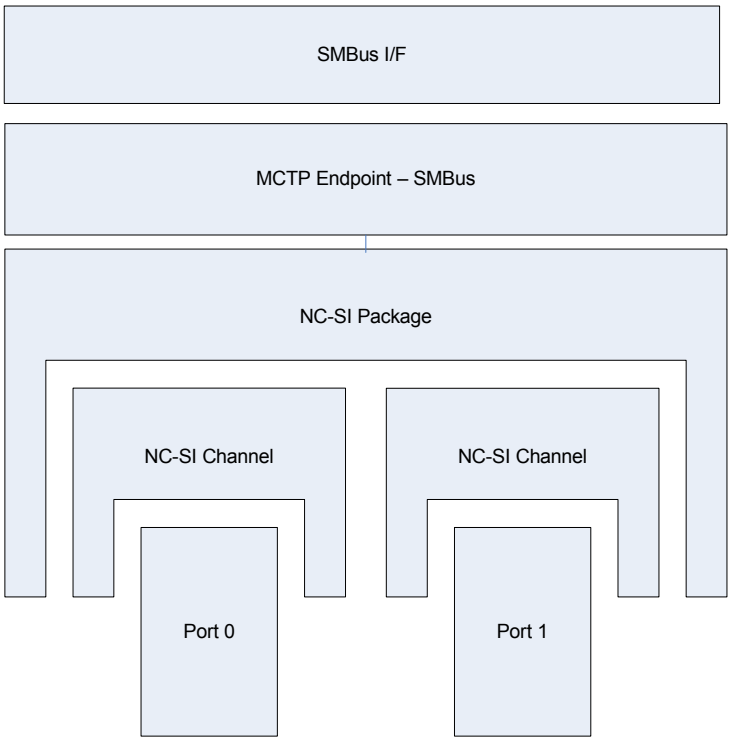


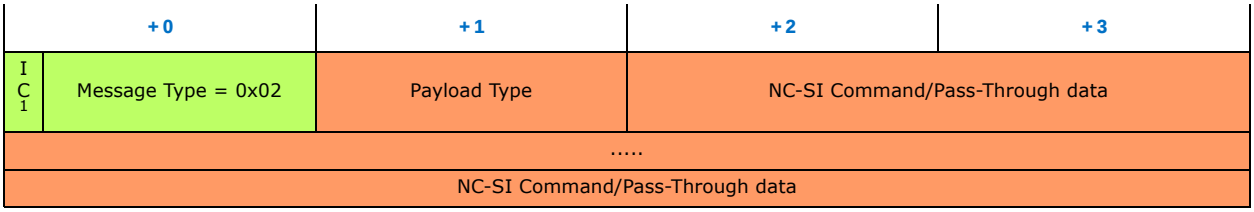
Figure 11-2. MCTP Endpoints Topology

11.6.3 MCTP over SMBus

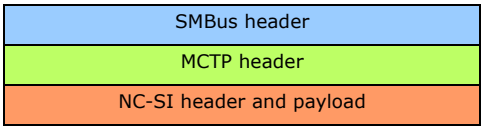
The message format used for NC-SI over MCTP over SMBus is as follows:

11.6.3.1 SMBus Discovery Process

+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							
MCTP Reserved				Header version = 1				Destination endpoint ID								Source endpoint ID								SOM		EOM		SEQ#		TO	Tag



1. IC = 0



The X540 follows the discovery process described in section 5.5 of the MCTP SMBus/I2C Transport Binding Specification (DSP0237). It indicates support for ASF in the SMBus getUID command (see Section 11.7.2.3.5). It will respond to any SMBus command using the MCTP command code - so that the bus owner knows the X540 supports MCTP.

11.6.4 NC-SI over MCTP

The X540 support for NC-SI over MCTP is similar to the support for NC-SI over RMII with the following exceptions:

1. A set of new NC-SI OEM commands used to support a shared IP or shared MAC usage model are added. In order to support this mode, a flow is added that allows snooping of transmit traffic to the pass-through channel.
2. The format of the packets is modified to account for the new transport layer as described in the sections that follow.

11.6.4.1 NC-SI Packets Format

NC-SI over MCTP defines six type of packets. The X540 supports three of them:

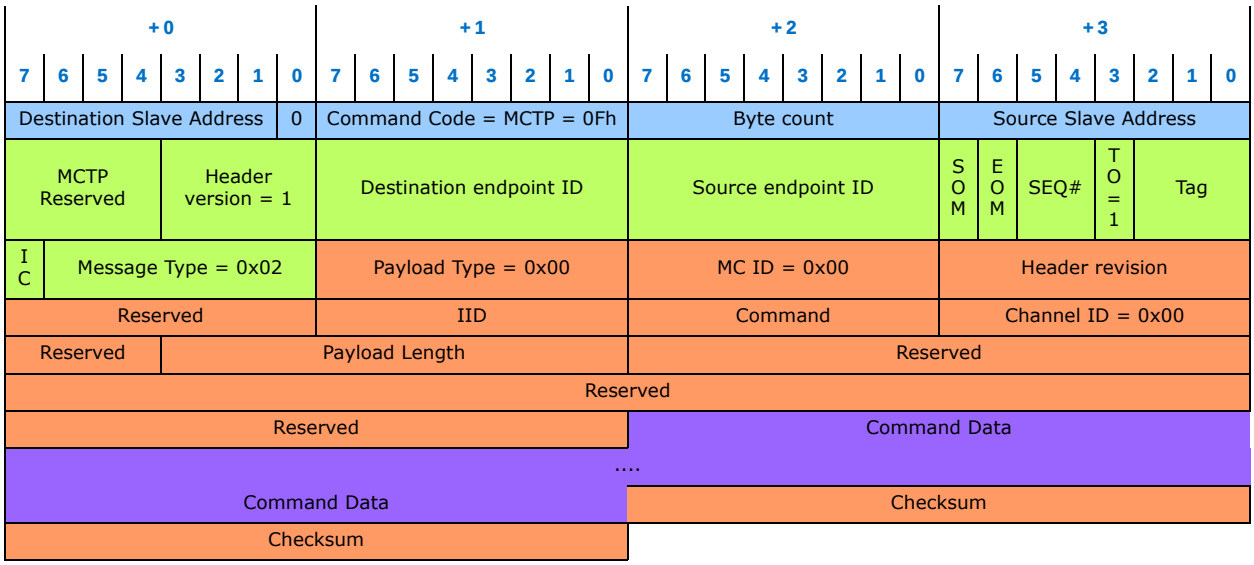
Table 11-1. NC-SI Over MCTP Packet Types

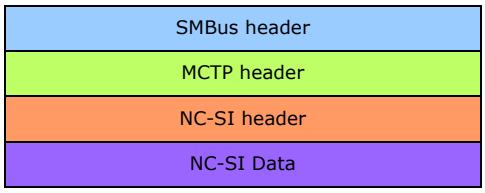
Packet Type	Payload Type	Supported	Section
Command	0x00	Yes	11.6.4.1.1
Response	0x01	Yes	11.6.4.1.2
Pass-through traffic	0x02	No	
Asynchronous Event Notification (AEN)	0x03	Yes	11.6.4.1.3
Outbound traffic	0x04	No	
Outbound acknowledge	0x05	No[1]	

1. Reliable delivery mode is not supported.

11.6.4.1.1 Command Packets

The format used for Command packets is as follows:

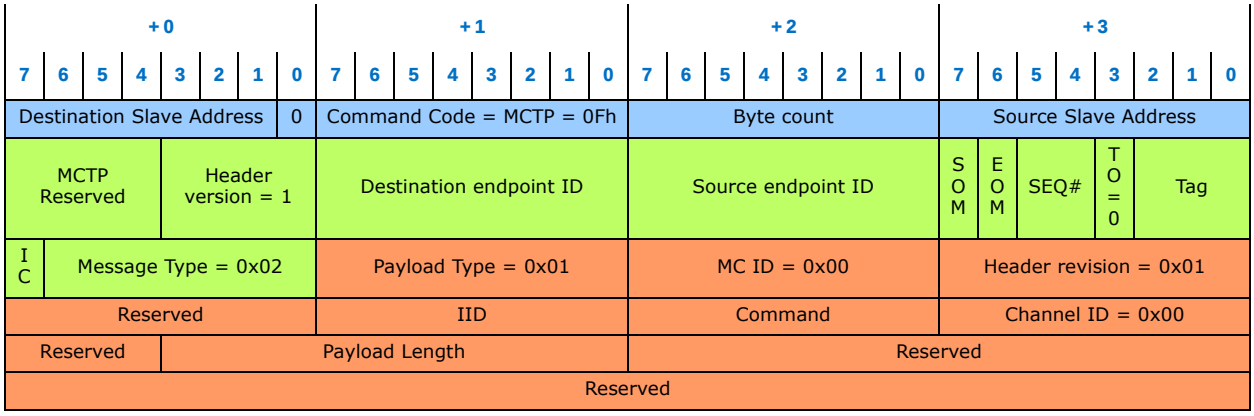


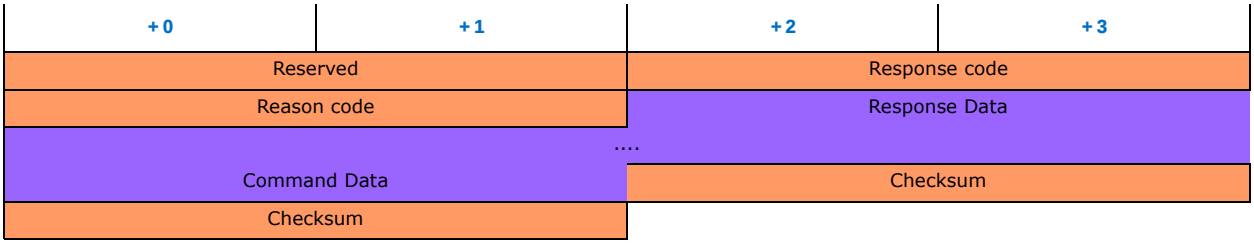


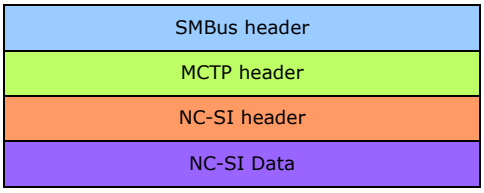
Note that the MAC header and MAC FCS present when working over NC-SI are not part of the packet in MCTP mode.

11.6.4.1.2 Response Packets

The format used for Response packets is as follows:



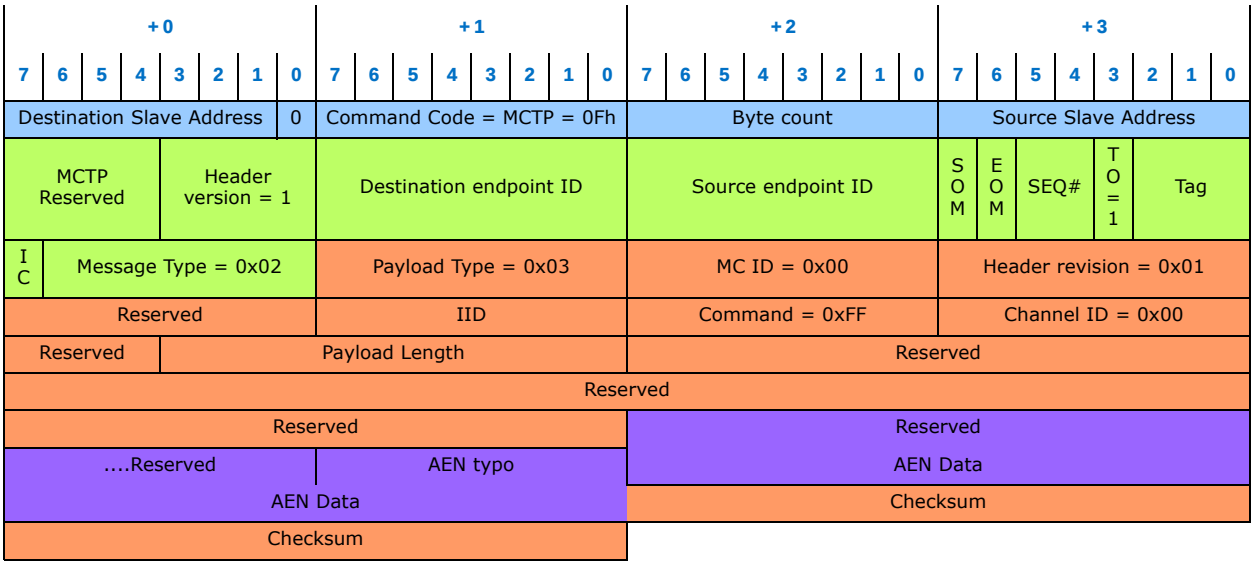


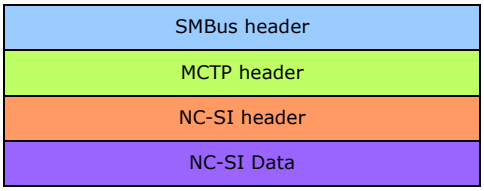


Note that the MAC header and MAC FCS present when working over NC-SI are not part of the packet in MCTP mode.

11.6.4.1.3 AEN Packets

The format used for AEN packets is as follows:





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 Manageability Programming Interfaces

11.7.1 NC-SI Programming

The X540 support the mandatory NC-SI commands as listed in Table 11-1. On top of these commands, the X540 supports also Intel vendor specific commands. The vendor specific commands are based on the NC-SI — "OEM Command". These commands are listed in the following subsections, enable the BMC to control the X540 specific features:

- RX filters:
  - Packet Addition Decision Filters 0x0…0x4
  - Packet Reduction Decision Filters 0x5…0x7
  - MNG2HOST register (Controls the forwarding of manageability packets to the host)
  - Flex 128 filters 0x0…0x3
  - Flex TCP/UDP port filters 0..0xA
  - IPv4/IPv6 filters
  - Ether type filters
- Get System Ethernet MAC Address – This command allows the BMC to retrieve the System Ethernet MAC Address used by the NC. This Ethernet MAC Address may be used for a "Shared Ethernet MAC Address" mode.
- Keep Phy Link Up (Veto bit) Enable/Disable – This feature allows the BMC to block Phy reset, which might cause session loss.
- TCO Reset – Allows the Management Controller to reset the Network Adapter.
- Checksum offloading – offloads IP/UDP/TCP checksum checking from the Management Controller.
- MACsec logic programming

These commands are designed to be compliant with their corresponding SMBus commands (if existing). All of the commands are based on a single NC-SI command, known as "OEM Command" described in Section 11.7.1.1.

11.7.1.1 OEM Command (0x50)

The OEM command may be used by the Management Controller to request the sideband interface to provide vendor-specific information. The Vendor Enterprise Number is the unique MIB/SNMP Private Enterprise number assigned by IANA per organization. Vendors are free to define their own internal data structures in the vendor data fields.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	**NC-SI Header**			
16..19	**Manufacturer ID (Intel 0x157)**			
20..	**Intel Command Number**	**Optional Data**		

11.7.1.2 OEM Response (0xD0)

The sideband interface shall return Unknown Command Type reason code for any un-recognized enterprise number using the following frame format. If the command is valid, the response, if any, is allowed to be vendor-specific. It is recommended to use the 0x8000 range for vendor-specific code.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	**NC-SI Header**			
16..19	**Response Code**		**Reason Code**	
20..23	**Manufacturer ID (Intel 0x157)**			
24..27	**Intel Command Number**	**Optional Return Data**		

Table 11-1 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
		0x5087	Invalid Driver State
		0x5088	Invalid NVM

11.7.1.3 Intel Commands

The table thatfollows lists the Intel commands and their associated Intel Command Number values. For detailed description of the commands and their parameters refer to following sections.

11.7.1.4 Control Command (Intel Command 0x00)

Table 11-1. Intel Command Summary

Intel Command	Parameter	Command Name	Supported Over MCTP
0x00	0x00	Set IP Filters Control	No
0x01	0x00	Get IP Filters Control	No
0x02	0x0A	Set Manageability to Host	No
	0x10	Set Flexible 128 Filter 0 Mask and Length	No
	0x11	Set Flexible 128 Filter 0 Data	No
	0x20	Set Flexible 128 Filter 1 Mask and Length	No
	0x21	Set Flexible 128 Filter 1 Data	No
	0x30	Set Flexible 128 Filter 2 Mask and Length	No
	0x31	Set Flexible 128 Filter 2 Data	No
	0x40	Set Flexible 128 Filter 3 Mask and Length	No
	0x41	Set Flexible 128 Filter 3 Data	No
	0x61	Set Packet Addition Filters	No
	0x63	Set Flex TCP/UDP Port Filters	No
	0x64	Set Flex IPv4 Address Filters	No
	0x65	Set Flex IPv6 Address Filters	No
	0x67	Set EtherType Filter	No
	0x68	Set Packet Addition Extended Decision Filter	No

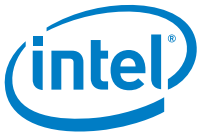
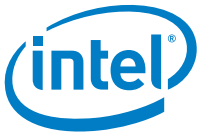


Table 11-1. Intel Command Summary (Continued)

Intel Command	Parameter	Command Name	Supported Over MCTP
0x3	0x0A	Get Manageability to Host	No
	0x10	Get Flexible 128 Filter 0 Mask and Length	No
	0x11	Get Flexible 128 Filter 0 Data	No
	0x20	Get Flexible 128 Filter 1 Mask and Length	No
	0x21	Get Flexible 128 Filter 1 Data	No
	0x30	Get Flexible 128 Filter 2 Mask and Length	No
	0x31	Get Flexible 128 Filter 2 Data	No
	0x40	Get Flexible 128 Filter 3 Mask and Length	No
	0x41	Get Flexible 128 Filter 3 Data	No
	0x61	Get Packet Addition Filters	No
	0x63	Get Flex TCP/UDP Port Filters	No
	0x64	Get Flex IPv4 Address Filters	No
	0x65	Get Flex IPv6 Address Filters	No
	0x67	Set EtherType Filter	No
	0x68	Set Packet Addition Extended Decision Filter	No
0x04	0x00	Set Unicast Packet Reduction	No
	0x01	Set Multicast Packet Reduction	No
	0x02	Set Broadcast Packet Reduction	No
	0x10	Set Unicast Extended Packet Reduction	No
	0x11	Set Multicast Extended Packet Reduction	No
	0x12	Set Broadcast Extended Packet Reduction	No
0x05	0x01	Get Multicast Packet Reduction	No
	0x02	Get Broadcast Packet Reduction	No
	0x10	Get Unicast Extended Packet Reduction	No
	0x11	Get Multicast Extended Packet Reduction	No
	0x12	Get Broadcast Extended Packet Reduction	No
0x06	N/A	Get System Ethernet MAC Address	No
0x20	N/A	Set Intel Management Control	No

Table 11-1. Intel Command Summary (Continued)

Intel Command	Parameter	Command Name	Supported Over MCTP
0x21	N/A	Get Intel Management Control	No
0x22	N/A	Perform TCO Reset or Firmware Reset	Yes
0x23	N/A	Enable IP/UDP/TCP Checksum Offloading	No
0x24	N/A	Disable IP/UDP/TCP Checksum Offloading	No
0x30	0x10	Transfer MACsec Ownership to MC	No
	0x11	Transfer MACsec Ownership to Host	No
	0x12	Initialize MACsec Rx	No
	0x13	Initialize MACsec Tx	No
	0x14	Set MACsec Rx Key	No
	0x15	Set MACsec Tx Key	No
	0x16	Enable network Tx Encryption	No
	0x17	Disable network Tx Encryption	No
	0x18	Enable network Rx Encryption	No
	0x19	Disable network Rx Encryption	No
0x31	0x01	Get MACsec Parameters	No
	0x02	Get MACsec Rx Parameters	No
	0x03	Get MACsec Tx Parameters	No
0x40	0x01	Enable OS2BMC Flow	No
	0x02	Enable Network to BMC Flow	No
	0x03	Enable Both Network to BMC and Host to BMC Flow	No
	0x04	Set BMC IP Address	No
0x41	N/A	Get 0S2BMC Parameters	No
0xF0	N/A	Write Configuration	Yes
0xF1	N/A	Read Configuration	Yes



11.7.1.4.1 Set Intel Filters Control – IP Filters Control Command (Intel Command 0x00)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x00	0x00	IP Filters control (3-2)	
24..27	IP Filters Control (1-0)			

While "IP Filters Control" has the following format:

Table 11-1 IP Filter Formats

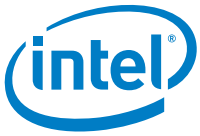
Bit #	Name	Description	Default Value
0	IPv4/IPv6 Mode	IPv6 (0b): There are 0 IPv4 filters and 4 IPv6 filters IPv4 (1b): There are 4 IPv4 filters and 3 IPv6 filters See the Management Filters Registers section or Section 11.3.4 for details.	1
15:1	Reserved	Reserved	
16	IPv4 Filter 0 Valid	Indicates if the IPv4 address configured in IPv4 address 0 is valid. Note: The Network Controller shall automatically set this bit to 1b if the "Set Intel Filter – IPv4 Filter Command" is used for filter 0.	0
17	IPv4 Filter 1 Valid	Indicates if the IPv4 address configured in IPv4 address 1 is valid. Note: The Network Controller shall automatically set this bit to 1b if the "Set Intel Filter – IPv4 Filter Command" is used for filter 1.	0
18	IPv4 Filter 2 Valid	Indicates if the IPv4 address configured in IPv4 address 2 is valid. Note: The Network Controller shall automatically set this bit to 1b if the "Set Intel Filter – IPv4 Filter Command" is used for filter 2.	0
19	IPv4 Filter 3 Valid	Indicates if the IPv4 address configured in IPv4 address 3 is valid. Note: The Network Controller shall automatically set this bit to 1b if the "Set Intel Filter – IPv4 Filter Command" is used for filter 3.	0
23:20	Reserved		
24	IPv6 Filter 0 Valid	Indicates if the IPv6 address configured in IPv6 address 0 is valid. Note: The Network Controller shall automatically set this bit to 1b if the "Set Intel Filter – IPv6 Filter Command" is used for filter 0.	0
25	IPv6 Filter 1 Valid	Indicates if the IPv6 address configured in IPv6 address 1 is valid. Note: The Network Controller shall automatically set this bit to 1b if the "Set Intel Filter – IPv6 Filter Command" is used for filter 1.	0

Table 11-1 IP Filter Formats (Continued)

26	IPv6 Filter 2 Valid	Indicates if the IPv6 address configured in IPv6 address 2 is valid. Note: The Network Controller shall automatically set this bit to 1b if the "Set Intel Filter – IPv6 Filter Command" is used for filter 2.	0
27	IPv6 Filter 3 Valid	Indicates if the IPv6 address configured in IPv6 address 3 is valid. Note: The Network Controller shall automatically set this bit to 1b if the "Set Intel Filter – IPv6 Filter Command" is used for filter 3.	0
28..31	Reserved		

11.7.1.4.2 Set Intel Filters Control – IP Filters Control Response (Intel Command 0x00, Filter Control Index 0x00)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x00	0x00		



11.7.1.5 Get Intel Filters Control Command (Intel Command 0x01)

11.7.1.5.1 Get Intel Filters Control – IP Filters Control Command (Intel Command 0x01, Filter Control Index 0x00)

This command controls different aspects of the Intel Filters.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x01	0x00		

11.7.1.5.2 Get Intel Filters Control – IP Filters Control Response (Intel Command 0x01, Filter Control Index 0x00)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x01	0x00	IP Filters Control (3-2)	
28..29	IP Filters Control (1-0)			

IP Filter Control: See Table 11-1.

11.7.1.6 Set Intel Filters Formats

11.7.1.6.1 Set Intel Filters Command (Intel Command 0x02)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x02	Filter Parameter	Filters Data (optional)	

11.7.1.6.2 Set Intel Filters Response (Intel Command 0x02)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..	0x02	Filter Parameter	Return Data (Optional)	

11.7.1.6.3 Set Intel Filters – Manageability to Host Command (Intel Command 0x02, Filter parameter 0x0A)

This command sets the Mng2Host register. The Mng2Host register controls whether Pass-Through packets destined to the BMC will also be forwarded to the host OS.

The Mng2Host register has the following structure:

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x02	0x0A	Manageability to Host (3-2)	
24..25	Manageability to Host (1-0)			

Table 11-1. Manageability to Host Field

Bits	Name	Description	Default
0	Decision Filter 0	Determines if packets that have passed Decision Filter 0 will be also forwarded to the host OS.	0b
1	Decision Filter 1	Determines if packets that have passed Decision Filter 1 will be also forwarded to the host OS.	0b
2	Decision Filter 2	Determines if packets that have passed Decision Filter 2 will be also forwarded to the host OS.	0b
3	Decision Filter 3	Determines if packets that have passed Decision Filter 3 will be also forwarded to the host OS.	0b
4	Decision Filter 4	Determines if packets that have passed Decision Filter 4 will be also forwarded to the host OS.	0b
5	Unicast & Mixed	Determines if Unicast & Mixed packets will be also forwarded to the host OS.	0b

Table 11-1. Manageability to Host Field

6	Global Multicast	Determines if Global Multicast packets will be also forwarded to the host OS.	1b
7	Broadcast	Determines if Broadcast packets will be also forwarded to the host OS.	1b
31:8	Reserved	Reserved	N/A

11.7.1.6.4 Set Intel Filters – Manageability to Host Response (Intel Command 0x02, Filter parameter 0x0A)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..	0x02	0x0A		

11.7.1.6.5 Set Intel Filters – Flex Filter 0/1/2/3 Enable Mask and Length Command (Intel Command 0x02, Filter Parameter 0x10/0x20/0x30/0x40)

The following command sets the Intel Flex filters Mask and length. Use Filter parameters 0x10/0x20/0x30/0x40 for Flexible filters 0/1/2/3 accordingly. See Section 11.3.5 for details of the programming.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x02	0x10/ 0x20/ 0x30/ 0x40	Mask Byte 1	Mask Byte 2
24..27	..	..	..	..
28...31	..	..	..	..
32...35	..	..	..	..
35...37	..	Mask Byte 16	Reserved	Reserved
38	Flexible Filter Length (8-128 bytes)			

11.7.1.6.6 Set Intel Filters – Flex Filter 0/1/2/3 Data Command (Intel Command 0x02, Filter Parameter 0x11/0x21/0x31/0x41)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x02	0x10/ 0x20/ 0x30/ 0x40		

11.7.1.6.7 Set Intel Filters – Flex Filter 0/1/2/3 Data Command (Intel Command 0x02, Filter parameter 0x11/0x21/0x31/0x41)

The following command sets the Intel Flex filters Data. Use Filter parameters 0x11/0x21/0x31/0x41 for Flexible filters 0/1/2/3 accordingly.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..	0x02	0x11/ 0x21/ 0x31/ 0x41	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 11-1. 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

Table 11-1. Filter Data Group

Code	Bytes Programmed	Filter Data Length
0x3	bytes 90-119	1 - 30
0x4	bytes 120-127	1 - 8

11.7.1.6.8 Set Intel Filters – Flex Filter 0/1/2/3 Data Response (Intel Command 0x02, Filter parameter 0x11/0x21/0x31/0x41)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x02	0x11/ 0x21/ 0x31/ 0x41		

Note: If Filter Data Length is larger than specified in Table 11-1 an Out of Range Reason code is returned.

11.7.1.6.9 Set Intel Filters – Packet Addition Decision Filter Command (Intel Command 0x02, Filter Parameter 0x61)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x02	0x61	Filter index	Decision Filter (MSB)
24..26	.....		Decision Filter (LSB)	

Filter index range: 0x0..0x4

If the filter index is bigger than 4, a command failed Response Code is returned with no reason.

Table 11-1. Filter Values

Bit #	Name	Description
0	Unicast (AND)	If set, packets must match a Unicast filter
1	Broadcast (AND)	If set, packets must match the Broadcast filter
2	VLAN (AND)	If set, packets must match a VLAN filter
3	IP Address (AND)	If set, packets must match an IP filter
4	Unicast (OR)	If set, packets must match a Unicast filter or a different "OR" filter
5	Broadcast	If set, packets must match the Broadcast filter or a different "OR" filter
6	Multicast (AND)	If set, packets must match the Multicast filter
7	ARP Request (OR)	If set, packets must match the ARP Request filter or a different OR filter
8	ARP Response (OR)	If set, packets can pass if match the ARP Response filter
9	Neighbor Discovery (OR)	If set, packets can pass if match the Neighbor Discovery filter
10	Port 0x298 (OR)	If set, packets can pass if match a fixed TCP/UDP Port 0x298 filter
11	Port 0x26F (OR)	If set, packets can pass if match a fixed TCP/UDP Port 0x26F filter
12	Flex port 0 (OR)	If set, packets can pass if match the TCP/UDP Port filter 0
13	Flex port 1 (OR)	If set, packets can pass if match the TCP/UDP Port filter 1
14	Flex port 2 (OR)	If set, packets can pass if match the TCP/UDP Port filter 2

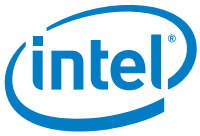


Table 11-1. Filter Values (Continued)

15	Flex port 3 (OR)	If set, packets can pass if match the TCP/UDP Port filter 3
16	Flex port 4 (OR)	If set, packets can pass if match the TCP/UDP Port filter 4
17	Flex port 5 (OR)	If set, packets can pass if match the TCP/UDP Port filter 5
18	Flex port 6 (OR)	If set, packets can pass if match the TCP/UDP Port filter 6
19	Flex port 7 (OR)	If set, packets can pass if match the TCP/UDP Port filter 7
20	Flex port 8 (OR)	If set, packets can pass if match the TCP/UDP Port filter 8
21	Flex port 9 (OR)	If set, packets can pass if match the TCP/UDP Port filter 9
22	Flex port 10 (OR)	If set, packets can pass if match the TCP/UDP Port filter 10
23	DHCPv6 (OR)	If set, packets can pass if match the DHCPv6 port (0x0223)
24	DHCP Client (OR)	If set, packets can pass if match the DHCP Server port (0x0043)
25	DHCP Server (OR)	If set, packets can pass if match the DHCP Client port (0x0044)
26	NetBIOS Name Service (OR)	If set, packets can pass if match the NetBIOS Name Service port (0x0089)
27	NetBIOS Datagram Service (OR)	If set, packets can pass if match the NetBIOS Datagram Service port (0x008A)
28	Flex TCO 0 (OR)	If set, packets can pass if match the Flex 128 TCO filter 0
29	Flex TCO 1 (OR)	If set, packets can pass if match the Flex 128 TCO filter 1
30	Flex TCO 2 (OR)	If set, packets can pass if match the Flex 128 TCO filter 2
31	Flex TCO 3 (OR)	If set, packets can pass if match the Flex 128 TCO filter 3

The filtering is divided into 2 decisions:

Bits 0,1,2,3,6 works in an “AND” manner –Thus, they all must be true in order for a packet to pass (if any were set).

Bits 5,7-31 work in an “OR” manner – Thus, at least one of them must be true for a packet to pass (if any were set).

See Section 11.3.6 for description of the decision filters.

Note: These filter settings will operate according to the VLAN mode, as configured according to the NC-SI specification. After disabling Packet Reduction filters the BMC must re-set the VLAN mode using the “Set VLAN” command.

11.7.1.6.10 Set Intel Filters – Packet Addition Decision Filter Response (Intel Command 0x02, Filter parameter 0x61)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x02	0x61		

11.7.1.6.11 Set Intel Filters – Flex TCP/UDP Port Filter Command (Intel Command 0x02, Filter Parameter 0x63)

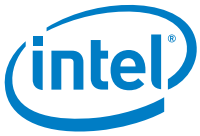
	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x02	0x63	Port filter index	TCP/UDP Port MSB
24	TCP/UDP Port LSB			

Filter index range: 0x0..0xA

If the filter index is bigger than 10, a command failed Response Code is returned with no reason.

11.7.1.6.12 Set Intel Filters – Flex TCP/UDP Port Filter Response (Intel Command 0x02, Filter parameter 0x63)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x02	0x63		



11.7.1.6.13 Set Intel Filters – IPv4 Filter Command (Intel Command 0x02, Filter Parameter 0x64)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x02	0x64	IP filter index	IPv4 Address (MSB)
24..26	...		IPv4 Address (LSB)	

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.

11.7.1.6.14 Set Intel Filters – IPv4 Filter Response (Intel Command 0x02, Filter parameter 0x64)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x02	0x64		

If the IP filter index is bigger than 3, a command failed Response Code is returned with no reason.

11.7.1.6.15 Set Intel Filters – IPv6 Filter Command (Intel Command 0x02, Filter Parameter 0x65)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x02	0x65	IP filter index	IPv6 Address (MSB, byte 15)
24..27	..	..	..	..

28..31	..	..	..	..
32..35	..	..	..	..
36..38	..	..	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

11.7.1.6.16 Set Intel Filters – IPv6 Filter Response (Intel Command 0x02, Filter Parameter 0x65)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x02	0x65		

If the IP filter index is bigger than 3, a command failed Response Code is returned with no reason.

11.7.1.6.17 Set Intel Filters – EtherType Filter Command (Intel Command 0x02, Filter Parameter 0x67)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x02	0x67	EtherType Filter Index	EtherType Filter MSB
24..27	..	..	EtherType Filter LSB	

Where the EtherType Filter has the format as described in the METF registers section.

Table 11-1. Ethertype Usage

Filter #	Usage	Note
2	User defined	
3	User defined	

11.7.1.6.18 Set Intel Filters - EtherType Filter Response (Intel Command 0x02, Filter parameter 0x67)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x02	0x67		

If the Ethertype filter Index is different than 2 or 3, a command failed Response Code is returned with no reason.

11.7.1.6.19 Set Intel Filters – Packet Addition Extended Decision Filter Command (Intel Command 0x02, Filter parameter 0x68)

DecisionFilter0 Bits 5,7-31 and DecisionFilter1 bits 8..10 work in an "OR" manner – Thus, at least one of them must be true for a packet to pass (if any were set).

See Figure 11-2 for description of the decision filters structure.

The command shall overwrite any previously stored value.

Note: Previous "Set Intel Filters – Packet Addition Decision Filter" command (0x61) should be kept and supported. For legacy reasons - If previous "Decision Filter" command is called – it should set the Decision Filter 0 as provided and set the extended Decision Filter to 0x0.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
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 11-1.

Filter 1: See table below:

Table 11-1. Extended Filter 1 Values

Bit #	Name	Description
0	Ethertype 0x88F8	AND filter
1	Ethertype 0x8808	AND filter
3:2	Ethertype 2 -3	AND filters
7:4	Reserved	Reserved
8	Ethertype 0x88F8	OR filter
9	Ethertype 0x8808	OR filter
11:10	Ethertype 2 -3	OR filters
31:12	Reserved	Reserved

11.7.1.6.20 Set Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x02, Filter Parameter 0x68)

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x02	0x68		

If the Extended Decision filter Index is bigger than 5, a command failed Response Code is returned with no reason.

11.7.1.7 Get Intel Filters Formats

11.7.1.7.1 Get Intel Filters Command (Intel Command 0x03)

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x03	Filter Parameter		

11.7.1.7.2 Get Intel Filters Response (Intel Command 0x03)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x03	Filter Parameter	Optional Return Data	

11.7.1.7.3 Get Intel Filters – Manageability to Host Command (Intel Command 0x03, Filter Parameter 0x0A)

This command retrieves the Mng2Host register. The Mng2Host register controls whether Pass-Through packets destined to the BMC will also be forwarded to the host OS.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x03	0x0A		

11.7.1.7.4 Get Intel Filters – Manageability to Host Response (Intel Command 0x03, Filter parameter 0x0A)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x03	0x0A	Manageability to Host (MSB)	
28..29	Manageability to Host (LSB)			

The Mng2Host register has the following structure.

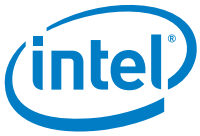
Table 11-1. Mng2Host Structure

Bits	Description	Default
0	Decision Filter 0	Determines if packets that have passed Decision Filter 0 will be also forwarded to the host OS.
1	Decision Filter 1	Determines if packets that have passed Decision Filter 1 will be also forwarded to the host OS.
2	Decision Filter 2	Determines if packets that have passed Decision Filter 2 will be also forwarded to the host OS.
3	Decision Filter 3	Determines if packets that have passed Decision Filter 3 will be also forwarded to the host OS.
4	Decision Filter 4	Determines if packets that have passed Decision Filter 4 will be also forwarded to the host OS.
5	Unicast & Mixed	Determines if Unicast & Mixed packets will be also forwarded to the host OS.
6	Global Multicast	Determines if Global Multicast packets will be also forwarded to the host OS.
7	Broadcast	Determines if Broadcast packets will be also forwarded to the host OS.
31:8	Reserved	Reserved

11.7.1.7.5 Get Intel Filters – Flex Filter 0/1/2/3 Enable Mask and Length Command (Intel Command 0x03, Filter Parameter 0x10/0x20/0x30/0x40)

The following command retrieves the Intel Flex filters Mask and length. Use Filter parameters 0x10/0x20/0x30/0x40 for Flexible filters 0/1/2/3 accordingly. See Section 11.3.5 for details of the values returned by this command.

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x03	0x10/ 0x20/ 0x30/ 0x40		



11.7.1.7.6 Get Intel Filters – Flex Filter 0/1/2/3 Enable Mask and Length Response (Intel Command 0x03, Filter parameter 0x10/0x20/0x30/0x40)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x03	0x10/ 0x20/ 0x30/ 0x40	Mask Byte 1	Mask Byte 2
28..31	..	..	..	..
32..35	..	..	..	..
36..39	..	..	..	..
40..43	..	Mask Byte 16	Reserved	Reserved
44	Flexible Filter Length			

11.7.1.7.7 Get Intel Filters – Flex Filter 0/1/2/3/4 Data Command (Intel Command 0x03, Filter Parameter 0x11/0x21/0x31/0x41)

The following command retrieves the Intel Flex filters Data. Use Filter parameters 0x11/0x21/0x31/0x41 for Flexible filters 0/1/2/3 accordingly.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x03	0x11/ 0x21/ 0x31/ 0x41	Filter Data Group 0..4	

The Filter Data Group parameter defines which bytes of the Flex filter are returned by this command:

Table 11-1. 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

11.7.1.7.8 Get Intel Filters – Flex Filter Data Response (Intel Command 0x03, Filter Parameter 0x11)

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..	0x03	0x11/ 0x21/ 0x31/ 0x41	Filter Data Group	Filter Data 1
	..	Filter Data N		

If the filter group number is bigger than 4, a command failed Response Code is returned with no reason.

11.7.1.7.9 Get Intel Filters – Packet Addition Decision Filter Command (Intel Command 0x03, Filter Parameter 0x61)

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x03	0x61	Decision filter index	

Filter index range: 0x0..0x4

11.7.1.7.10 Get Intel Filters – Packet Addition Decision Filter Response (Intel Command 0x03, Filter Parameter 0x0A)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x03	0x61	Decision Filter (MSB)	
28..29	Decision Filter (LSB)			

The Decision filter structure returned is described in Table 11-1.

If the Decision filter index is bigger than 4, a command failed Response Code is returned with no reason.

11.7.1.7.11 Get Intel Filters – Flex TCP/UDP Port Filter Command (Intel Command 0x03, Filter Parameter 0x63)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x03	0x63	TCP/UDP Filter Index	

Filter index range: 0x0..0xA

11.7.1.7.12 Get Intel Filters – Flex TCP/UDP Port Filter Response (Intel Command 0x03, Filter Parameter 0x63)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x03	0x63	TCP/UDP Filter Index	TCP/UDP Port (1)
28	TCP/UDP Port (0)			

Filter index range: 0x0..0xA

If the TCP/UDP Filter Index is bigger than 10, a command failed Response Code is returned with no reason

11.7.1.7.13 Get Intel Filters – IPv4 Filter Command (Intel Command 0x03, Filter Parameter 0x64)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x03	0x64	IPv4 Filter Index	

Note: The filters index range can vary according to the "IPv4/IPv6 Mode" setting in the "Filters Control" command

IPv4 Mode: Filter index range: 0x0..0x3
IPv6 Mode: This command should not be used in IPv6 mode.

11.7.1.7.14 Get Intel Filters – IPv4 Filter Response (Intel Command 0x03, Filter Parameter 0x64)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x03	0x64	IPv4 Filter Index	IPv4 Address (3)
28..29	IPv4 Address (2-0)			

If the IPv4 Filter Index is bigger than 3, a command failed Response Code is returned with no reason.

11.7.1.7.15 Get Intel Filters – IPv6 Filter Command (Intel Command 0x03, Filter Parameter 0x65)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x03	0x65	IPv6 Filter Index	

Note: The filters index range can vary according to the "IPv4/IPv6 Mode" setting in the "Filters Control" command.

IPv4 Mode: Filter index range: 0x0..0x2
IPv6 Mode: Filter index range: 0x0..0x3

11.7.1.7.16 Get Intel Filters – IPv6 Filter Response (Intel Command 0x03, Filter Parameter 0x65)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x03	0x65	IPv6 Filter Index	IPv6 Address (MSB, Byte 16)
28..31	..	..	..	..
32..35	..	..	..	..
36..39	..	..	..	..
40..42	..	..	IPv6 Address (LSB, Byte 0)	

If the IPv6 Filter Index is bigger than 3, a command failed Response Code is returned with no reason.

11.7.1.7.17 Get Intel Filters – EtherType Filter Command (Intel Command 0x03, Filter Parameter 0x67)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x03	0x67	EtherType Filter Index	

Valid indices: 2..3

See Table 11-1 for description of the various Ethertype filters usage.

11.7.1.7.18 Get Intel Filters - EtherType Filter Response (Intel Command 0x03, Filter Parameter 0x67)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			

16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x03	0x67	EtherType Filter Index	EtherType Filter MSB
28..30	..	..	EtherType Filter LSB	

If the Ethertype filter Index is different than 2 or 3, a command failed Response Code is returned with no reason.

11.7.1.7.19 Get Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x03, Filter Parameter 0x68)

This command allows the BMC to retrieve the Extended Decision Filter.

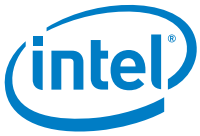
	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x03	0x68	Extended Decision Filter Index	

11.7.1.7.20 Get Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x03, Filter Parameter 0x68)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x03	0x68	Decision Filter Index	Decision Filter 1 MSB
28..31	..	..	Decision Filter 1 LSB	Decision Filter 0 MSB
32..34	..	..	Decision Filter 0 LSB	

Where Decision Filter 0 & 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 no reason.



11.7.1.8 Set Intel Packet Reduction Filters Formats

11.7.1.8.1 Set Intel Packet Reduction Filters Command (Intel Command 0x04)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x04	Filter Parameter	Optional 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 11-1. Packet Reduction Data

Bit #	Name	Description
2:0	Reserved	
3	IP Address	If set, all packets must also match an IP filter
9:4	Reserved	
10	Port 0x298	If set, all packets can pass if match a fixed TCP/UDP Port 0x298 filter
11	Port 0x26F	If set, all packets can pass if match a fixed TCP/UDP Port 0x26F filter
12	Flex port 0	If set, all packets can pass if match the TCP/UDP Port filter 0
13	Flex port 1	If set, all packets can pass if match the TCP/UDP Port filter 1
14	Flex port 2	If set, all packets can pass if match the TCP/UDP Port filter 2
15	Flex port 3	If set, all packets can pass if match the TCP/UDP Port filter 3
16	Flex port 4	If set, all packets can pass if match the TCP/UDP Port filter 4
17	Flex port 5	If set, all packets can pass if match the TCP/UDP Port filter 5
18	Flex port 6	If set, all packets can pass if match the TCP/UDP Port filter 6
19	Flex port 7	If set, all packets can pass if match the TCP/UDP Port filter 7
20	Flex port 8	If set, all packets can pass if match the TCP/UDP Port filter 8
21	Flex port 9	If set, all packets can pass if match the TCP/UDP Port filter 9
22	Flex port 10	If set, all packets can pass if match the TCP/UDP Port filter 10

Table 11-1. Packet Reduction Data (Continued)

27:23	Reserved	
28	Flex TCO 0	If set, all packets can pass if match the Flex 128 TCO filter 0
29	Flex TCO 1	If set, all packets can pass if match the Flex 128 TCO filter 1
30	Flex TCO 2	If set, all packets can pass if match the Flex 128 TCO filter 2
31	Flex TCO 3	If set, all packets can pass if match the Flex 128 TCO filter 3

For extended packet reduction command, the following fields should also be programmed.

Table 11-2. Extended Packet Reduction Format

Bit #	Name	Description
0..1	Reserved	Used by the regular NC-SI commands
2	EtherType2 (AND)	If set, packets must also match the EtherType filter 2.
3	EtherType3 (AND)	If set, packets must also match the EtherType filter 3.
4..7	Reserved	
8..9	Reserved	Used by the regular NC-SI commands
10	EtherType2 (OR)	If set, packets can pass if it match the EtherType filter 2.
11	EtherType3 (OR)	If set, packets can pass if it match the EtherType filter 2.
12..31	Reserved	

The filtering is divided into 2 decisions:

Unicast Reduction Filter - Bit 3 and Extended Unicast Reduction Filter bits 0..2 work in an "AND" manner –Thus, they all must be true in order for a packet to pass (if any were set).

Unicast Reduction Filter Bits 5, 7-31 and Extended Unicast Reduction Filter bits 8..10 work in an "OR" manner – Thus, at least one of them must be true for a packet to pass (if any were set).

See Section 11.3.6 for description of the decision filters.

11.7.1.8.2 Set Intel Packet Reduction Filters Response (Intel Command 0x04)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			

16..19	Response Code		Reason Code
20..23	Manufacturer ID (Intel 0x157)		
24..	0x04	Filter Parameter	Optional Return Data

11.7.1.8.3 Set Unicast/Multicast/Broadcast packet Reduction Command (Intel Command 0x04, Filter Parameter 0x00/0x01/0x02)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x04	0x00 / 0x01 / 0x02	Packet Reduction Table (MSB)	
24..25	Packet Reduction Table (LSB)			

This command shall cause the Network Controller to filter packets that have passed due to the Unicast/Multicast/Broadcast filter. Note that Unicast filtering might be affected by other filters, as specified in NC-SI.

The filtering of these packets will be done such that the Management Controller may add a logical condition that a packet must match, or it shall be discarded.

Note: Packets that might have been blocked may still pass due to other decision filters.

In order to disable Unicast/Multicast/Broadcast packet reduction the Management Controller should set all reductions filters to 0b. Following such a setting the Network Controller forwards to the BMC all packets that have passed the Unicast Ethernet MAC Address/Global Multicast/Broadcast filters as specified in NC-SI.

11.7.1.8.4 Set Unicast/Multicast/Broadcast Packet Reduction Response (Intel Command 0x04, Reduction Filter Parameter 0x00/0x01/0x02)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x04	0x00 / 0x01 / 0x02		

11.7.1.8.5 Set Unicast/Multicast/Broadcast Extended Packet Reduction Command (Intel Command 0x04, Filter Parameter 0x10/0x11/0x12)

In "Set Intel Reduction Filters" add another parameter "Unicast Extended Packet Reduction (Intel Command 0x04, Filter parameter 0x10)" such that the byte count is 0xE. The command shall have the following format:

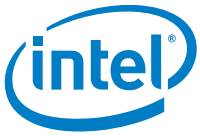
	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x04	0x10 /0x11 / 0x12	Extended Reduction Filter MSB	..
24..27	..	Extended Reduction Filter LSB	Reduction Filter Table (MSB)	..
28..29	..	Reduction Filter Table (LSB)		

The command shall overwrite any previously stored value.

Note: See Table 11-1 and Table 11-2 for description of the Unicast Extended Packet Reduction format.

11.7.1.8.6 Set Unicast/Multicast/Broadcast Extended Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x10 / 0x11 / 0x12)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x04	0x10 / 0x11 / 0x12		



11.7.1.9 Get Intel Packet Reduction Filters Formats

11.7.1.9.1 Get Intel Packet Reduction Filters Command (Intel Command 0x05)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x05	Filter Parameter		

11.7.1.9.2 Get Intel Packet Reduction Filters Response (Intel Command 0x05)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..	0x05	Filter Parameter	Optional Return Data	

11.7.1.9.3 Get Unicast/Multicast/Broadcast Packet Reduction Command & Response (Intel Command 0x05, Filter Parameter 0x00/0x01/0x02)

This command retrieves the requested Packet Reduction Filter. The format of the "Optional Return Data" follows the structure of the Unicast packet Reduction command described in Section 11.7.1.8.3.

11.7.1.9.4 Get Unicast/Multicast/Broadcast Extended Packet Reduction Command & Response (Intel Command 0x05, Filter Parameter 0x10/0x11/0x12)

This command retrieves the requested Extended Packet Reduction Filter. The format of the "Optional Return Data" follows the structure of the Unicast Extended Packet Reduction command described in Section 11.7.1.8.5.

11.7.1.10 System Ethernet MAC Address

11.7.1.10.1 Get System Ethernet MAC Address Command (Intel Command 0x06)

In order to support a system configuration that requires the Network Controller to hold the Ethernet MAC Address for the BMC (i.e. – "Shared Ethernet MAC Address" mode), the following command is provided to allow the BMC to query the Network Controller for a valid Ethernet MAC Address.

The Network Controller shall return the System Ethernet MAC Addresses. The BMC should use the returned Ethernet MAC Addressing as a shared Ethernet MAC Address by setting it using the "Set Ethernet MAC Address" command as defined in NC-SI 1.0.

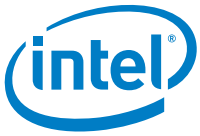
It is also recommended that the BMC uses the Packet Reduction & Manageability to Host command to set the proper filtering method.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20	0x06			

11.7.1.10.2 Get System Ethernet MAC Address Response (Intel Command 0x06)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x06	Ethernet MAC Address		
28..30	Ethernet MAC Address			

The MAC address is returned in Network Order.



11.7.1.11 Set Intel Management Control Formats

11.7.1.11.1 Set Intel Management Control Command (Intel Command 0x20)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x20	0x00	Intel Management Control	

The “Intel Management Control” byte is defined in the table that follows:

Bit #	Default	Description
0	0b	Enable Critical Session Mode (the “Keep PHY Link Up”, “Veto Bit”) 0b — Disabled; 1b — Enabled When Critical Session Mode is enabled, the PHY is not reset on PE_RST# nor PCIe resets (in-band and link drop). Other reset events are not affected — LAN Power Good reset, Device Disable, Force TCO, and PHY reset by SW. 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.
1…7	0x0	Reserved

11.7.1.11.2 Set Intel Management Control Response (Intel Command 0x20)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x20	0x00		

11.7.1.12 Get Intel Management Control Formats

11.7.1.12.1 Get Intel Management Control Command (Intel Command 0x21)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x20	0x00		

11.7.1.12.2 Get Intel Management Control Response (Intel Command 0x21)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..26	0x21	0x00	Intel Management Control 1	

"Intel Management Control 1" byte is as described above in Section 11.7.1.11.1.

11.7.1.13 TCO Reset

Depending on the bit set in the TCO mode field this command will cause the X540 to perform either:

1. TCO Reset, if Force TCO reset is enabled in the NVM (see Section 6.5.2). The Force TCO reset will clear the data-path (RX/TX) of the X540 to enable the BMC to transmit/receive packets through the X540.
   - If the BMC has detected that the OS is hung and has blocked the RX/TX path The Force TCO reset will clear 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 will consider Force TCO command as an indication that the OS is hung and will clear the DRV_LOAD flag (disable the driver). If TCO reset is disabled in NVM the X540

clears the *CTRL_EXT.DRV_LOAD* bit but does not reset the data-path and notifies BMC on successful completion.

2. Firmware Reset. This command will cause re-initialization of all the manageability functions and re-load of manageability related NVM words (e.g. Firmware patch code).
   - When the BMC has loaded new management related NVM image (e.g. Firmware patch) the Firmware Reset command will load management related NVM information without 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.
   -

Notes: Force TCO reset will affect only the channel (port) that the command was issued to.

Following firmware reset, BMC will need to re-initialize all ports.

11.7.1.13.1 Perform Intel TCO Reset Command (Intel Command 0x22)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20	0x22	TCO Mode		

Where:

TCO Mode:

Field	Bit(s)	Description
DO_TCO_RST[1]	0	Do TCO reset. 0b = Do Nothing 1b = Perform TCO Reset Note: Setting this bit generates a one time LAN port reset event.
Reserved	1	Reserved (set to 0).
Firmware Reset	2	Reset Manageability and re-load Manageability related NVM words (see Section 4.2.1.8). 0b = Do Nothing 1b = Issue Firmware Reset to manageability. Note: Setting this bit generates a one time Firmware reset event. Following Firmware Reset Management related data from NVM is loaded.
Reserved	3-7	Reserved. (Set to 0x00)

1. TCO Reset operation enabled in NVM.

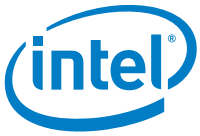
11.7.1.13.2 Perform Intel TCO Reset Response (Intel Command 0x22)

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..26	0x22			

11.7.1.14 Checksum Offloading

This command will enable the checksum offloading filters in the Network Controller.

When enabled, these filters will block any packets that did not pass IP, UDP & TCP checksums from being forwarded to the Management Controller. This feature does not support tunneled IPv4/IPv6 packet inspection.



11.7.1.14.1 Enable Checksum Offloading Command (Intel Command 0x23)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20	0x23			

11.7.1.14.2 Enable Checksum Offloading Response (Intel Command 0x23)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..26	0x23			

11.7.1.14.3 Disable Checksum Offloading Command (Intel Command 0x24)

This command shall cause the Network Controller to stop verifying the IP/UDP/TCP checksums

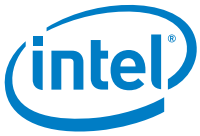
	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20	0x24			

11.7.1.14.4 Disable Checksum Offloading Response (Intel Command 0x24)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..26	0x24			

11.7.1.15 MACsec Support Commands

The following commands may be used by the MC to control the different aspects of the MACsec engine.



11.7.1.15.1 Transfer MACsec Ownership to BMC Command (Intel Command 0x30, Parameter 0x10)

This command shall cause the X540 to clear all MACsec parameters, forcefully release Host ownership and grant the ownership to the BMC. The BMC may allow the Host to use the BMC's key for traffic by setting the "Host Control – Allow Host Traffic" bit. Activating this command will clear all the MACsec parameters.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x30	0x10	Host Control	

Table 11-1 MACsec Host Control

Bit	Description
0	Reserved
1	Allow Host Traffic: 0b – Host traffic is blocked 1b – Host traffic is allowed
2..7	Reserved

11.7.1.15.2 Transfer MACsec Ownership to BMC Response (Intel Command 0x30, Parameter 0x10)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x30	0x10		

11.7.1.15.3 Transfer MACsec Ownership to Host Command (Intel Command 0x30, Parameter 0x11)

This command shall cause the X540 to clear all MACsec parameters, release BMC ownership and grant ownership to the host.

In this scenario traffic from/to the MC shall be validated by the host's programmed keys. It is recommended that the MC will try to establish network communication with a remote station to verify that the Host was successful in programming the keys.

Activating this command will clear all the MACsec parameters.

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x30	0x11		

11.7.1.15.4 Transfer MACsec Ownership to Host Response (Intel Command 0x30, Parameter 0x11)

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x30	0x11		

11.7.1.15.5 Initialize MACsec RX Command (Intel Command 0x30, Parameter 0x12)

This command may be used by the MC to initialize the MACsec RX engine. This command should be followed by a "Set MACsec RX Key" command to establish a MACsec environment.

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x30	0x12	RX Port Identifier	
24..27	RX SCI [0..3]			
28..29	RX SCI [4..5]			

Where:

- **RX Port Identifier** – the port number by which the NC will identify RX packets. It is recommended that the MC uses 0x0 as the port identifier. Note: The MC should use the same port identifier when performing the key-exchange.
- **RX SCI** – A 6 bytes unique identifier for the MACsec TX CA. It is recommended that the MC uses its Ethernet MAC Address value for this field.

11.7.1.15.6 Initialize MACsec RX Response (Intel Command 0x30, Parameter 0x12)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x30	0x12		

11.7.1.15.7 Initialize MACsec TX Command (Intel Command 0x30, Parameter 0x13)

This command may be used by the MC to initialize the MACsec TX engine. This command should be followed by a "Set MACsec TX Key" command to establish a MACsec environment.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x30	0x13	TX Port Identifier	
24..27	TX SCI [0..3]			
28..31	TX SCI [4..5]		Reserved	
32..35	Packet Number Threshold			
36	TX Control			

- **TX Port Identifier** – For this implementation this field is a "don't care" and is automatically set to 0x0.
- **TX SCI** – A 6 bytes unique identifier for the MACsec TX CA. It is recommended that the MC uses its Ethernet MAC Address value for this field.
- **PN Threshold** – When a new key is programmed, the Packet Number is reset to 0x1. With each TX packet, The Packet Number increments by 1 and is inserted to the packet (to avoid replay attacks). The PN Threshold value is the 3 MSBytes of the TX Packet number after which a "Key Exchange Required" AEN will be sent to the MC.

Example: a PN Threshold of 0x123456 means that when the PN reaches 0x123456FF a notification will be sent. The fourth byte of the PN Threshold can be seen as a reserved bit, because it will always be treated as 0xFF by the NC.

- **TX Control:**

Bit	Description
0..4	Reserved
5	Always Include SCI in TX: 0b – Do not include SCI in TX packets 1b – Include SCI in TX packets
6..7	Reserved

11.7.1.15.8 Initialize MACsec TX Response (Intel Command 0x30, Parameter 0x13)

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x30	0x13		

11.7.1.15.9 Set MACsec RX Key Command (Intel Command 0x30, Parameter 0x14)

This command may be used by the MC to set a new MACsec RX key. Upon receiving this command the NC shall switch to the new RX key and send the response.

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x30	0x14	Reserved	RX SA AN
24..27	RX MACsec Key MSB	..	..	..
28..31	..	..	..	..
32..35	..	..	..	..
36..39	..	..	..	RX MACsec Key LSB

Where:

- **RX SA AN** – The Association Number to be used with this key.
- **RX MACsec Key** – the 128 bits (16 bytes) key to be used for RX

11.7.1.15.10 Set MACsec RX Key Response (Intel Command 0x30, Parameter 0x14)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x30	0x14		

11.7.1.15.11 Set MACsec TX Key Command (Intel Command 0x30, Parameter 0x15)

This command may be used by the MC to set a new MACsec TX key. Upon receiving this command the NC shall switch to the new TX key and send the response.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x30	0x15	Reserved	TX SA AN
24..27	TX MACsec Key MSB	..	..	..
28..31	..	..	..	..
32..35	..	..	..	..
36..39	..	..	..	TX MACsec Key LSB

Where:

- **TX SA AN** – The Association Number to be used with this key.
- **TX MACsec Key** – the 128 bits (16 bytes) key to be used for TX

11.7.1.15.12 Set MACsec TX Key Response (Intel Command 0x30, Parameter 0x15)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x30	0x15		

11.7.1.15.13 Enable Network TX Encryption Command (Intel Command 0x30, Parameter 0x16)

This command may be used by the MC to (re)enable Encryption of outgoing Pass-Through packets.

After this command is issued and until a response is received, the state of any outgoing packets is undetermined.

By default Network TX Encryption is enabled.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x30	0x16		

11.7.1.15.14 Enable Network TX Encryption Response (Intel Command 0x30, Parameter 0x16)

Following sending this response the NC shall stop encrypting outgoing Pass-Through packets.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x30	0x16		

11.7.1.15.15 Disable Network TX Encryption Command (Intel Command 0x30, Parameter 0x17)

This command may be used by the MC to disable Encryption of outgoing Pass-Through packets.

After this command is issued and until a response is received, the state of any outgoing packets is undetermined.

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x30	0x17		

11.7.1.15.16 Disable Network TX Encryption Response (Intel Command 0x30, Parameter 0x17)

Following sending this response the NC shall start encrypting outgoing Pass-Through packets.

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x30	0x17		

11.7.1.15.17 Enable Network RX Decryption Command (Intel Command 0x30, Parameter 0x18)

This command may be used by the MC to (re)enable decryption of incoming Pass-Through packets. This will cause the NC to execute MACsec offload and to post the frames to the MC (or host) only if the MACsec operation succeeds.
After this command is issued and until a response is received, the state of any incoming packets is undetermined.

By default Network RX Decryption is disabled.

	Bits			
Bytes	31..24	23..16	15..08	07..00

00..15	NC-SI Header		
16..19	Manufacturer ID (Intel 0x157)		
20..21	0x30	0x18	

11.7.1.15.18 Enable Network RX Decryption Response (Intel Command 0x30, Parameter 0x18)

Following sending this response the NC shall begin decrypting incoming Pass-Through packets.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x30	0x18		

11.7.1.15.19 Disable Network RX Decryption Command (Intel Command 0x30, Parameter 0x19)

This command may be used by the MC to disable decryption of incoming Pass-Through packets.
After this command is issued and until a response is received, the state of any incoming packets is undetermined.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..21	0x30	0x19		

11.7.1.15.20 Disable Network RX Decryption Response (Intel Command 0x30, Parameter 0x19)

Following sending this response the NC shall stop decrypting incoming Pass-Through packets.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x30	0x19		

11.7.1.15.21 Get MACsec Parameters format (Intel Command 0x31)

The following commands may be used by the MC to retrieve the different MACsec parameters.

These commands responses are valid only if the BMC owns the MACsec.

11.7.1.15.22 Get MACsec RX Parameters Command (Intel Command 0x31, Parameter 0x01)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x31	0x01		

11.7.1.15.23 Get MACsec RX Parameters Response (Intel Command 0x31, Parameter 0x01)

This command allows the MC to retrieve the currently configured set of RX MACsec parameter.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x31	0x01	Reserved	
28..31	MACsec Owner Status	MACsec Host Control Status	RX Port Identifier	

32..35	SCI [0..3]		
36..39	SCI [4..5]	Reserved	RX SA AN
40..43	RX SA Packet Number		

Where:

Table 11-2 MACsec Owner Status

Value	Description
0x0	Host is MACsec owner
0x1	BMC is MACsec owner

Table 11-3 MACsec Host Control Status

Bit	Description
0	Reserved
1	Allow Host Traffic: 0b- Host traffic is blocked 1b – Host traffic is allowed
2..7	Reserved

- RX Port Identifier – The RX Port identifier
- RX SCI – The RX SCI identifier.
- RX SA AN – The association number associated with the active SA (for which the last valid RX MACsec packet was received).
- RX SA Packet Number – Is the last packet number, as read from the last valid RX MACsec packet.

11.7.1.15.24 Get MACsec TX Parameters Command (Intel Command 0x31, Parameter 0x02)

This command allows the MC to retrieve the currently configured set of TX MACsec parameter.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..22	0x31	0x02		

11.7.1.15.25 Get MACsec TX Parameters Response (Intel Command 0x31, Parameter 0x02)

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..27	0x31	0x2	Reserved	
28..31	MACsec Owner Status	MACsec Host Control Status	TX Port Identifier	
32..35	SCI [0..3]			
36..39	SCI [4..5]		Reserved	TX SA AN
40..43	TX SA Packet Number			
44.47	Packet Number Threshold			
48	TX Control Status			

Where:

Table 11-4 MACsec Owner Status:

Value	Description
0x0	Host is MACsec owner
0x1	BMC is MACsec owner

Table 11-5 MACsec Host Control Status

Bit	Description
0	Reserved
1	Allow Host Traffic: 0b- Host traffic is blocked 1b – Host traffic is allowed
2..7	Reserved

- TX Port Identifier – Reserved to 0x0 for this implementation.
- TX SCI – The RX SCI identifier.
- TX SA AN – The association number currently used for the active SA.
- TX SA Packet Number – Is the last packet number, as read from the last valid RX MACsec packet.

- Packet Number Threshold.

Table 11-6 TX Control Status:

Bit	Description
0..4	Reserved
5	Include SCI: 0b – Do not include SCI in TX packets 1b – Include SCI in TX packets
6..7	Reserved

11.7.1.16 MACsec AEN (Intel AEN 0x80)

The following is the AEN that may be sent by the NC following a MACsec event.

This AEN must be enabled using the NC-SI "AEN Enable" command, using bit 16 (0x10000) of the AEN Enable mask.

	Bits			
Bytes	31..24	23..16	15..08	07..00
00..15	NC-SI AEN Header			
20..23	Reserved			0x80
24..27	Reserved			MACsec Event Cause

Where:

MACsec Event Cause has the following format:

Bit #	Description
0	Host requested ownership
1	Host released ownership
2	TX Key Packet Number (PN) threshold met
3..7	Reserved

11.7.1.17 OS2BMC configuration

These commands control enabling of the OS2BMC flow

11.7.1.17.1 Enable OS2BMC Flow Command (Intel Command 0x40, Index 0x1)

	Bits			
Bytes	**31...24**	**23...16**	**15...08**	**07...00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x01		

11.7.1.17.2 Enable OS2BMC Flow Response (Intel Command 0x40, Index 0x1)

	Bits			
Bytes	**31...24**	**23...16**	**15...08**	**07...00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x01		

11.7.1.17.3 Enable Network to BMC Flow Command (Intel Command 0x40, Index 0x2)

	Bits			
Bytes	**31...24**	**23...16**	**15...08**	**07...00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x02		

11.7.1.17.4 Enable Network to BMC Flow Response (Intel Command 0x40, Index 0x2)

	Bits			
Bytes	**31...24**	**23...16**	**15...08**	**07...00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x02		

11.7.1.17.5 Enable both Host and Network to BMC flows Command (Intel Command 0x40, Index 0x3)

	Bits			
Bytes	31...24	23...16	15...08	07...00
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x03		

11.7.1.17.6 Enable both Host and Network to BMC flows Response (Intel Command 0x40, Index 0x3)

	Bits			
Bytes	31...24	23...16	15...08	07...00
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x03		

11.7.1.17.7 Set BMC IP address Command (Intel Command 0x40, Index 0x4)

This command is used to expose the BMC IP address in the BMCIP registers. When this command is sent, the OS is alerted of the IP address change by the setting of an EICR.MNG interrupt cause. This command should be used at power up and every time the IP address changes. It is used by the host to identify local IPSec flows so that they are not offloaded.

The IP type entry indicate whether the IP address is an IPv4 or an IPv6 address:

0 = Ipv4

1 = IPv6

2 = No IP address, then the command should not include an IP address.

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x40	0x04	IP type	IPv6 Address (MSB, byte 15)/IPv4 Address (MSB, byte 3)
24..27	IPv6 Address (byte 14)/ IPv4 Address (byte 2)	IPv6 Address (byte 13)/ IPv4 Address (byte 1)	IPv6 Address (byte 12)/ IPv4 Address (LSB, byte 0)	IPv6 Address (byte 11)/ Reserved
28..31	..	..	..	..
32..35	..	..	..	..
36..38	..	..	IPv6 Address (LSB, byte 0)/Reserved	

11.7.1.17.8 Set BMC IP address Response (Intel Command 0x40, Index 0x4)

	Bits			
Bytes	**31...24**	**23...16**	**15...08**	**07...00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x04		

11.7.1.17.9 Get OS2BMC parameters Command (Intel Command 0x41)

	Bits			
Bytes	**31...24**	**23...16**	**15...08**	**07...00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20	0x41			

11.7.1.17.10 Get OS2BMC parameters Response (Intel Command 0x41)

	Bits			
Bytes	**31...24**	**23...16**	**15...08**	**07...00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x41	Status	IPv6 Address (MSB, byte 15)/IPv4 Address (MSB, byte 3)	IPv6 Address (byte 14)/ IPv4 Address (byte 2)
28..31	IPv6 Address (byte 13)/ IPv4 Address (byte 1)	IPv6 Address (byte 12)/ IPv4 Address (LSB, byte 0)	IPv6 Address (byte 11)/ Reserved	..
32..35	..	..	..	..
36..39	..	..	..	..
40..42	..	IPv6 Address (LSB, byte 0)/Reserved		

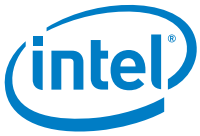
Where the Status byte partition is as follow:

Table 11-1. Status Byte Description

Bits	Content
0	0 = IPv4 1 = IPv6 Relevant only if the IP address valid bit is set.
1	IP address valid
2	Network to BMC status 0 = network 2 BMC flow is disabled 1 = network 2 BMC flow is enabled.
3	OS2BMC status 0 = OS2BMC flow is disabled 1 = OS2BMC flow is enabled.
7:4	Reserved.

11.7.2 ActionC2SMBus Programming

This section describes the SMBus transactions supported in Advanced Pass-Through (APT) mode.



11.7.2.1 Write SMBus Transactions (BMC → the X540)

The following table lists the different SMB write transactions supported by the X540.

TCO Command	Transaction	Command		Fragmentation	Section
Transmit Packet	Block Write	First: Middle: Last:	0x84 0x04 0x44	Multiple	11.7.2.1.1
Transmit Packet	Block Write	Single:	0xC4	Single	11.7.2.1.1
Receive Enable	Block Write	Single:	0xCA	Single	11.7.2.1.3
Management Control	Block Write	Single:	0xC1	Single	
Update MNG RCV filter parameters	Block Write	Single:	0xCC	Single	11.7.2.1.6
Force TCO	Block Write	Single:	0xCF	Single	11.7.2.1.4
Request Status	Block Write	Single:	0xDD	Single	11.7.2.1.2
Update MACsec parameters	Block Write	Single:	0xC9	Single	11.7.2.1.7

11.7.2.1.1 Transmit Packet Command

The Transmit Packet command behavior is detailed in section 3.2.5. The Transmit Packet fragments have the following format:

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				

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 by the X540.

11.7.2.1.2 Request Status Command

The TCO controller can initiate a request to read the X540 manageability status by sending this command.

When it receives this command, the X540 initiates a notification to the external controller (when it is ready with the status), and then the external controller will be able to read the status, by issuing a read status command (see section 11.7.2.2.3). Request Status Command format:

Function	Command	Byte Count	Data 1
Request status	0xDD	1	0

11.7.2.1.3 Receive Enable Command

The Receive Enable command is a single fragment command that is used to configure the X540.

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).

Note: If the receive enable command is short and thus does not include all the parameters, then the parameters will be taken from most recent previous configuration (either the most recent long Receive-Enable command in which the particular value was set, or the NVM if there was no such previous long Receive-Enable command).

Func.	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.	Interf. data byte	Alert value byte

While...

- **Receive control byte** (data byte 1) has the following format

Field	Bit(s)	Description
RCV_EN	0	Receive TCO enable. 0b = Disable Receive TCO packets. Rx Packets are not directed to BMC and Auto ARP response is not enabled. 1b = Enable Receive TCO packets. Setting this bit enables all manageability receive filtering operation. The enable of the specific filtering is done through loading the "Receive Enable 1" word in the NVM, or through special configuration command (see Section 11.7.2.1.6).
RCV_ALL	1	Receive All enable. When set to 1, all LAN packets received over the wire that passed L2 filtering are forwarded to the BMC. This flag is meaningful only if the RCV_EN bit is set as well.
EN_STA	2	Enable Status reporting when set to 1.

Field	Bit(s)	Description
EN_ARP_RES	3	Enable ARP response 0b = Disable. The X540 treats ARP packets as any other packet. These packets are forwarded to BMC if it pass other (non-ARP) filtering. 1b = Enable. The X540 automatically respond to all received ARP requests which match its IP address. Note that setting this bit doesn't change the Rx filtering settings. Appropriate Rx filtering to enable ARP request packets to reach manageability unit should be set by the BMC or by the NVM. The BMC IP address is provided as part of the Receive Enable message (bytes 8-11). If short version of the command is used the X540 uses IP address configured in the most recent long version of the command in which the EN_ARP_RES bit was set. If no such previous long command exists, then the X540 uses IP address configured in the NVM as ARP Response IPv4 Address in PT LAN configuration structure. If CBDM bit is set the X540 uses BMC dedicated Ethernet MAC Address in ARP response packets. If the CBDM bit is not set, BMC uses the host Ethernet MAC Address. Setting this bit requires appropriate assertion of bits RCV_EN and RCV_ALL. Otherwise, the command aborts with no processing.
NM	5:4	Notification Method. Define the notification method that the X540 uses. 00b = SMB Alert 01b = Asynchronous Notify 10b = Direct Receive 11b = Not Supported. **Note**: Both SMB addresses must be configured to the same notification method.
reserved	6	Reserved
CBDM	7	Configure BMC dedicated Ethernet MAC Address. **Note**: This bit should be zero when the RCV_EN bit (bit 0) is not set. 0b = The X540 shares the same Ethernet MAC Address for MNG and host defined in the "NVM LAN Core 0/1 Modules" in the NVM. 1b = The X540 uses a dedicated Ethernet MAC Address. The BMC Ethernet MAC Address is set in bytes 2-7 in this command. If short version of the command is used, the X540 uses the Ethernet MAC Address configured in the most recent long version of the command in which the CBDM bit was set. If no such previous long command exists, then the X540 uses the Ethernet MAC Address configured in the MMAL and MMAH fields in the NVM. When Dedicated Ethernet MAC Address feature is activated, the X540 uses the following registers for Rx filtering. The BMC should not modify the following registers: MNG Decision Filter – MDEF7 (and its corresponding bit MANC2H[7]) MNG Ethernet MAC Address 3 – MMAL3 and MMAH3 (and its corresponding bit MFVAL[3]).

- MNG Ethernet MAC Address (data bytes 2-7)

Ignored if CBDM bit is not set. This Ethernet MAC Address will be used for configuration of the dedicated Ethernet MAC Address. In addition, it will be used in the ARP response packet, when EN_ARP_RES bit is set. This Ethernet MAC Address will continue to be used when CBDM bit is set in subsequent short versions of this command.

- MNG IP address (data bytes 8-11)

Ignored if EN_ARP_RES bit is not set. This IP address will be used to filter ARP request packets. This IP address will continue to be used when EN_ARP_RES is set in subsequent short versions of this command.

- Asynchronous Notification SMB address (data byte 12)

This address will be used for the Asynchronous Notification SMB transaction and for Direct Receive.

- Interface Data (Data byte 13)

Interface data byte to be used in Asynchronous Notification.

- Alert data (Data byte 14).

Alert Value data byte to be used in the Asynchronous Notification.

11.7.2.1.4 Force TCO Command

This command will cause the X540 to perform TCO Reset, if Force TCO reset is enabled in word "Management HW Config Control" in the NVM. The Force TCO reset will clear the data-path (RX/TX) of the X540 to enable the BMC to transmit/receive packets through the X540.

Note: Force TCO reset will be asserted only to the port related to the SMB address the command was issued to.

The X540 will consider Force TCO command as an indication that the OS is hung and will clear the DRV_LOAD flag.

Force TCO Reset Command format:

Function	Command	Byte Count	Data 1
Force TCO reset	0xCF	1	TCO mode

While TCO Mode is shown in the following table:

Field	Bit(s)	Description
DO_TCO_RST	0	Do TCO reset. 0b = Do Nothing 1b = Perform TCO Reset
Reserved	1	Reserved. (Set to 0x0)
Firmware Reset[1]	2	Reset Manageability and re-load Manageability related NVM words 0b = Do Nothing 1b = Issue Firmware Reset to manageability. Note: Setting this bit generates a one time Firmware reset event. Following Firmware Reset Management related data from NVM is loaded.
Reserved	3-7	Reserved. (Set to 0x00)

[1] Before initiating a Firmware reset command, one should disable TCO receive via Receive Enable Command, setting RCV_EN to 0, and wait for 200 milliseconds before initiating Firmware Reset command. In addition, the BMCshould not transmit during this period.

11.7.2.1.5 Management Control

This command is used to set generic manageability parameters. The parameters list is shown in the table below. The command is C1h, which states that it is a management

control command. The first data byte is the parameter number and the data after-words (length and content) are parameter specific as shown in the table.

Note: If in the update configuration, the parameter that the BMC sets is not supported by the X540, the X540 will not NACK the transaction. After the transaction ends, the X540 will discard the data and will assert a transaction abort status (see Section 3.2.5.2).

This is the format of the Management control command:

Function	Command	Byte Count	Data 1	Data 2	...	Data N
Management control	0xC1	N	Parameter number (PN#)	Parameter dependent		

The table below shows the different parameters and their content:

Parameter	PN#	Parameter Data
Keep PHY Link Up	0x00	A single byte parameter — Data 2: Bit 0 Programming of the MMNGC.MNG_VETO bit. Bit [7:1] Reserved

11.7.2.1.6 Update MNG RCV Filter Parameters

This command is used to set the manageability receive filters parameters. The parameters list is shown in the table below. The command is CCh, which states that it is a parameter update. The first data byte is the parameter number and the data after-words (length and content) are parameter specific as shown in the table.

Note: If in the update configuration, the parameter that the BMC sets is not supported by the X540, the X540 will not NACK the transaction. After the transaction ends, the X540 will discard the data and will assert a transaction abort status (see Section 3.2.5.2).

Detailed description of receive filtering capabilities and configuration is in Section 11.3.

The format of the Update MNG RCV filter parameters is shown in the table below:

Function	Command	Byte Count	Data 1	Data 2	...	Data N
Update MNG RCV filter parameters	0xCC	N	Parameter number (PN#)	Parameter dependent		

The following table lists the different parameters and their contents:

Parameter	PN#	Parameter Data
Filters Enable	0x 1	Defines generic filters configuration. The structure of this parameter is 4 bytes as the "MANC Value LSB" and "MANC Value MSB" loaded from the NVM. Note the general filter enable is in the receive enable command. Which enable receive filtering. This parameter specifies which filters should be enabled. ARP filtering and dedicated Ethernet MAC Address can also be enabled through the receive enable command (see Section 11.7.2.1.3).
MNG2HOST configuration	0x A	This parameter defines which manageability packets will be directed to the HOST memory as well. Data 2:5: MNG2H register setting (Data 2 is the LSB)
Flex filter 0 enable MASK and length	0x 10	Flex filter 0 mask. Data 2:17 – MASK. Bit 0 in data 2 is the first bit of the MASK Data 18:19 – Reserved. Should be zero. Data 20 – Flexible Filter length (must be >= 2).
Flex filter 0 data	0x 11	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 mandatory 30 bytes; it may vary according to filter's length and must NOT be padded by zeros.
Flex filter 1 enable MASK and length	0x 20	Same as parameter 0x10 but for filter 1.
Flex filter 1 data	0x 21	Same as parameter 0x11 but for filter 1
Flex filter 2 enable MASK and length	0x 30	Same as parameter 0x10 but for filter 2.
Flex filter 2 data	0x 31	Same as parameter 0x11 but for filter 2.
Flex filter 3 enable MASK and length	0x 40	Same as parameter 0x10 but for filter 3.
Flex filter 3 data	0x 41	Same as parameter 0x11 but for filter 3.
Filters Valid	0x 60	4 bytes to determine which of the the X540 filter registers contain valid data. Loaded into the MFVAL0 and MFVAL1 registers. Should be updated after the contents of a filter register are updated. Data 2 – MSB of MFVAL ... Data 5 is the LSB
Decision Filters	0x 61	5 bytes to load the manageability decision filters (MDEF) Data 2 – Decision Filter number Data 3 – MSB of MDEF register for this decision filter ... Data 6 is the LSB

Parameter	PN#	Parameter Data
VLAN Filters	0x 62	3 bytes to load the VLAN tag filters (MAVTV) Data 2 – VLAN Filter number Data 3 – MSB of VLAN Filter Data 4 – LSB of VLAN Filter
Flex Ports Filters	0x 63	3 bytes to load the manageability flex port filters (MFUTP). Data 2 – Flex Port Filter number Data 3 – MSB of flex port filter Data 4 — LSB of flex port filter
IPv4 Filters	0x 64	5 bytes to load the IPv4 address filter (MIPAF, DW 15:12) Data 2 – IPv4 address filter number (0-3) Data 3 – MSB of IPv4 address filter … Data 6 is the LSB
IPv6 Filters	0x 65	17 bytes to load IPv6 address filter (MIPAF) Data 2 – IPv6 address filter number (0-3) Data 3 – MSB of IPv6 address filter … Data 18 is the LSB
MAC Filters	0x 66	7 bytes to load Ethernet MAC Address filters (MMAL, MMAH) Data 2 – Ethernet MAC Address filters pair number (0-3) Data 3 – MSB of Ethernet MAC Address … Data 8 is the LSB
Ethertype Filters	0x 67	6 bytes to load Ethertype filters (METF) Data 2 — METF filter index (valid values are 0..3) Data 3 — MSB of METF ... Data 6 is the LSB
Extended Decision Filter	0x 68	10 bytes to load the extended decision filters (MDEF_EXT & MDEF) Data 2 - MDEF filter index (valid values are 0..6) Data 3 - MSB of MDEF_EXT (DecisionFilter1)... Data 6 is the LSB Data 7 - MSB of MDEF (DecisionFilter0)... Data 10 is the LSB The command shall overwrite any previously stored value. Note: Previous "Decision Filter" command (0x61) is still supported. For legacy reasons - If previous "Decision Filter" command (0x61) is called - it should set the MDEF as provided and set the extended Decision Filter (MDEF_EXT) to 0x0.

11.7.2.1.7 Update MACsec Parameters

This command is used to set the manageability MACsec parameters. The parameters list is shown in the table below. The first data byte is the parameter number and the data after-words (length and content) are parameter specific as shown in the table.

This is the format of the Update MACsec parameters command:

Function	Command	Byte Count	Data 1	Data 2	...	Data N
Update MACsec filter parameters	0xC9	N	Parameter number (PN#)	Parameter dependent		

The following table lists the different parameters and their contents:

Parameter	PN#	Parameter Data
Transfer MACsec ownership to BMC	0x 10	Data 2: Host Control: Bit 0 – Reserved Bit 1 – Allow host traffic (0b – blocked, 1b – allowed) Bit 2...31 – Reserved.
Transfer MACsec ownership to Host	0x 11	No data needed
Initialize MACsec RX	0x 12	Data 2: RX Port Identifier (MSB) ... Data 3: (LSB) RX Port Identifier – the port number by which the X540 will identify RX packets. It is recommended that the BMC uses 0x0 as the port identifier. Note: The BMC should use the same port identifier when performing the key-exchange. Data 4 : RX MAC SecY (MSB) … Data 9: (LSB)
Initialize MACsec TX	0x 13	Data 2: TX Port Identifier (MSB) ... Data 3: (LSB) — must be set to zero Data 4: TX SCI (MSB) … Data 7: TX SCI (LSB) TX SCI – A 6 bytes unique identifier for the MACsec TX CA. It is recommended that the BMC uses its Ethernet MAC Address value for this field. Data 8: Reserved Data 9: Reserved Data 10: Packet Number Threshold (MSB) … Data 12: (LSB) PN Threshold – When a new key is programmed, the Packet Number is reset to 0x1. With each TX packet, The Packet Number is incremented by 1 and inserted to the packet (to avoid replay attacks). The PN Threshold value is 3 MSBytes of the TX Packet number after which a “Key Exchange Required” (ToDo: Add link) AEN will be sent to the BMC. Example: a PN Threshold of 0x123456 means that when the PN reaches 0x12345600 a notification will be sent Data 22: TX Control — See Table 11-1
Set MACsec RX Key	0x 14	Data 2: Reserved Data 3: RX SA AN (The Association Number to be used with this key) Data 4: RX MACsec Key (MSB) … Data 19: (LSB) — (16 bytes key to be used)
Set MACsec TX Key	0x 15	Data 3: TX SA AN (The Association Number to be used with this key) Data 4: TX MACsec Key (MSB) … Data 19: (LSB) — (16 bytes key to be used)
Enable MACsec Network TX encryption	0x 16	No data needed
Disable MACsec Network TX encryption	0x 17	No data needed

Table 11-1 TX Control

Bit	Description
0..4	Reserved

Table 11-1 TX Control

5	Always Include SCI in TX: 0b – Do not include SCI in TX packets 1b – Include SCI in TX packets
6..7	Reserved

11.7.2.2 Read SMBus Transactions (the X540 to BMC)

The following table lists the different SMB read transactions supported by the X540. All the read transactions are compatible with SMBus Read Block Protocol format.

TCO Command	Transaction	Command	Op-Code		Fragmentation	Section
Receive TCO Packet	Block Read	0xC0 or 0xD0	First: Middle: Last[1]	0x90 0x10 0x50	Multiple	11.7.2.2.1
Read Receive Enable configuration	Block Read	0xDA	Single:	0xDA	Single	11.7.2.2.7
Read the X540 Status	Block Read	0xC0 or 0xD0 or 0xDE	Single:	0xDD	Single	11.7.2.2.3
Read Management parameters	Block Read	0xD1	Single:	0xD1	Single	11.7.2.2.5
Read MNG RCV filter parameters	Block Read	0xCD	Single:	0xCD	Single	11.7.2.2.6
Get system Ethernet MAC Address	Block Read	0xD4	Single	0xD4	Single	11.7.2.2.4
Read MACsec parameters	Block Read	0xD9	Single	0xD9	Single	11.7.2.2.8

1. Last fragment of the receive TCO packet is the packet status.

Note: The X540 will respond to one of the commands 0xC0/0xD0 within the time defined in the SMBus notification timeout and flags word in the NVM (see Section 6.5.4.3.)

Note: 0xC0/0xD0 commands are used for more than one payload. If the BMC issues these read commands, and the X540 has no pending data to transfer, it will always return as default opcode 0xDD with the X540 status, and will not NACK the transaction.

Note: If an SMBus quick read command is received, it is handled as a Read the X540 Status command (See Section 11.7.2.2.3 for details).

11.7.2.2.1 Receive TCO LAN Packet Transaction

The BMC will use this command to read the packet received on the LAN and its status. When the X540 has a packet to deliver to the BMC, it asserts the SMB notification, for the BMC to read the data (or direct receive). Upon receiving notification of the arrival of LAN receive packet, the BMC should begin issuing a Receive TCO packet command using the block read protocol. The packet can be delivered in more than one SMB fragment (at least two — one for the packet, and the other one for the status), and the BMC should follow the 'F' and 'L' bit.

The opcode can have these values:

- 0x90 — First Fragment
- 0x10 — Middle Fragment.
- 0x50 — Packet status (last fragment) as described below in Section 11.7.2.2.2.

If the external BMC does not finish reading the whole packet within a timeout period since the packet has arrived, the packet is silently discarded. The timeout period is set according to the SMBus notification timeout NVM parameter (see Section 6.5.4.3).

Function	Command
Receive TCO packet	0xC0 or 0xD0

Data returned from the X540:

Function	Byte Count	Data 1 (Op-Code)	Data 2	...	Data N
Receive TCO First Fragment	N	90	Packet Data Byte	...	Packet Data Byte
Receive TCO Middle Fragment		10			
Receive TCO Last Fragment		50			

11.7.2.2.2 Receive TCO LAN Status Payload Transaction

This transaction is the last transaction that the X540 will issue when a packet that was received from the LAN is transferred to the BMC. The transaction will contain the status of the received packet. The format of the status transaction is as follows::

Function	Byte Count	Data 1 (Op-Code)	Data 2 – Data 17 (Status data)
Receive TCO long status	17 (0x11)	0x50	See Table 11-2 below. For more details on the specific bit fields see Section 7.1.6.

Table 11-2 Receive TCO Last Fragment Status Data Content

Name	Bit(s)	Description
Packet Length	13:0	Packet length including CRC, only 14 LSB bits.
Reserved	24:14	Reserved
CRC	25	CRC stripped indication
Reserved	28:26	Reserved
VEXT	29	Additional VLAN present in packet
Reserved	33:30	Reserved
Reserved	34	Reserved
LAN	35	LAN Number
Reserved	63:36	Reserved
Reserved	71:64	Reserved
Status	79:72	See Table 11-3
Reserved	87:80	Reserved
MNG status	127:88	See Table 11-4. This field should be ignored if Receive TCO is not enabled,

Table 11-3 Status Info

Field	Bit(s)	Description
reserved	7:4	reserved
IPCS	3	Ipv4 Checksum Calculated on Packet
L4CS	2	L4 Checksum Calculated on Packet
UDPCS	1	UDP checksum calculated on packet
Reserved	0	Reserved

Table 11-4 MNG Status

Name	Bits	Description
Pass RMCP 0x026F	0	Set when the UDP/TCP port of the MNG packet is 0x26F
Pass RMCP 0x0298	1	Set when the UDP/TCP port of the MNG packet is 0x298
Pass MNG broadcast	2	Set when the MNG packet is a broadcast packet

Table 11-4 MNG Status

Pass MNG neighbor	3	Set when the MNG packet is a neighbor discovery packet.
Pass ARP req / ARP resp	4	Set when the MNG packet is an ARP response/request packet
Reserved	7:5	Reserved
Pass MNG VLAN Filter Index	10:8	
MNG VLAN Address Match	11	Set when the MNG packet matches one of the MNG VLAN filters
Unicast Address Index	14:12	Indicates which of the 4 Unicast Ethernet MAC Addresses match the packet. Valid only if the Unicast Address match is set.
Unicast Address match	15	Set when there is a match to any of the 4 Unicast Ethernet MAC Addresses.
L4 port Filter Index	22:16	Indicate the flex filter number
L4 port match	23	Set when there is a match to any of the UDP / TCP port filters
Flex TCO Filter index	26:24	
Flex TCO filter match	27	
IP address Index	29:28	Set when there is a match to the IP filter number. (IPv4 or IPv6)
IP address match	30	Set when there is a match to any of the IP address filters
IPv4 packet	31	Set to '0' when packet is IPv4 (regardless of address match).
Decision Filter match	39:32	Set when there is a match to one of the Decision filters

11.7.2.2.3 Read Status Command

The BMC can read the X540 status. The X540 asserts an alert prior to the BMC reading the status bytes. There can be two reasons for the X540 to send status to the BMC (described in Section 3.2.3):

1. The external BMC asserts a request for reading the X540 status.
2. The X540 detects a status change as described in Section 3.2.3.

Note that commands 0xC0/0xD0 are for backward compatibility. 0xD0/0xC0 can be used for other payloads the X540 will define in the opcode, which payload this transaction will be. When 0xDE command is set, the X540 will always return opcode 0xDD with the X540 status. The BMC reads the event causing the notification, using the read status command as follows:

Function	Command
Read status	0xC0 or 0xD0 or 0xDE

Function	Byte Count	Data 1 (Op-Code)	Data 2 (Status data 1)	Data 3 (Status Data 2)
Receive TCO partial status	3	0xDD	See below	

The following table lists the status data byte 1:

Bit	Name	Description
7	LAN Port	0b = Alert came from LAN port 0. 1b = Alert came from LAN port 1
6	TCO command aborted	0b = A TCO command abort event has not occurred since the last Read Status cycle. 1b = A TCO command abort event has occurred since the last Read Status cycle. See Section 3.6.5.2 for command abort flow.
5	Link Status indication	0b = LAN link down 1b = LAN link is up
4	PHY Link Forced Up	Contains the value of the MMNGC.MNG_VETO bit.
3	Initialization indication[1]	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.
2	Reserved	Reserved as 0b
1:0	Power state[2]	00b = Dr state. 01b = D0u state. 10b = D0 state. 11b = D3 state.

1. This indication is asserted when the X540 manageability block reloadsthe NVM and its internal data-base is updated to NVM default values. This is an indication that the external BMC should re-configure the X540, if other values beside the NVM default should be configured.
2. The "D" state is marked in this order: D0, D0u, Dr, and D3.

Status data byte 2 is used for the BMC for an indication whether the LAN driver is alive and running.

The driver valid indication is a bit that is set by the driver when it is coming up, and cleared when it goes down to Dx state or cleared by the HW on PCI reset.

Bits 2 and 1 indicate that the LAN driver is not stuck. Bit 2 indicates whether the interrupt line of the LAN function is asserted, and bit 1 indicates whether driver between the last Read Status Cycle dealt the interrupt line.

The following table lists thestatus data byte 2:

Bit	Name	Description
7	Reserved	Reserved
6	Reserved	Reserved
5	Reserved	Reserved
4	MACsec indication	If set — indicates that a MACsec event has occurred. Use the "Read MACsec parameters" with "MACsec interrupt cause" parameter to read the interrupt cause
3	Driver Valid indication	0b = LAN driver is not alive 1b = LAN driver is alive
2	Interrupt pending indication	0b = LAN interrupt is not asserted. 1b = LAN interrupt is asserted.
1	ICR register read/write	0b = ICR register was not read since the last Read Status Cycle. 1b = ICR register was read since the last Read Status cycle. Reading the ICR means that the driver has dealt with the interrupt that was asserted
0	Reserved	Reserved

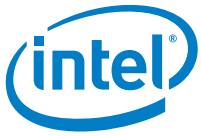
The following table lists the possible values of bits 2, 1 and what the BMC can assume according to that:

Previous	Current	
Don’t care	00b	Interrupt is not pending – O.K
00b	01b	New interrupt is asserted – O.K
10b	01b	New interrupt is asserted – O.K
11b	01b	Interrupt is waiting for reading – O.K
01b	01b	Interrupt is waiting for reading by driver more than one read status Cycle – Not OK (possible driver hang state).
Don’t Care	11b	Previous interrupt was read and current interrupt is pending – O.K
Don’t Care	10b	Interrupt is not pending – O.K

Note: The BMC reads should consider the time it takes for the driver to deal with the interrupt (few micro-seconds), too frequent reads will give false indications.

11.7.2.2.4 Get System Ethernet MAC Address

The Get System Ethernet MAC Address returns the system Ethernet MAC Address (RAL0, RAH0) over the SMBus. This command is a single fragment Read Block transaction, with the following format:



Function	Command
Get system Ethernet MAC Address	0xD4

Data returned from the X540:

Function	Byte Count	Data 1 (Op-Code)	Data 2	...	Data 7
Get system Ethernet MAC Address	7	0xD4	Ethernet MAC Address MSB	...	Ethernet MAC Address LSB

11.7.2.2.5 Read Management Parameters

In order to read the management parameters the BMC should execute two SMB 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.

This is the 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 will read the parameter that was set in the block write command:

Function	Command
Read management parameter	0xD1

Data returned from the X540:

Function	Byte Count	Data 1 (Op-Code)	Data 2	Data 3	...	Data N
Read management parameter	N	0xD1	Parameter number (PN#)	Parameter dependent		

The returned data is as follows:

Parameter	PN#	Parameter Data
Keep PHY Link Up	0x00	A single byte parameter — Data 2: Bit 0 Reflects the setting of the MMNGC.MNG_VETO bit. Bit [7:1] Reserved
Wrong parameter request	0xFE	the X540 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 X540.
the X540 not ready	0xFF	the X540 only: Returned on read parameters command when the data that should have been read is not ready. The BMC should retry the read transaction.

Note: It might be that the parameter that is returned is not the parameter requested by the BMC. The BMC should verify the parameter number (default parameter to be returned is 1h).

Note: It is BMC's responsibility to follow the procedure defined above. If the BMC sends a block read command (as described above) which is not preceded by a block write command with bytecount=1, the X540 will set Parameter Number in the read block transaction to be 0xFE.

11.7.2.2.6 Read MNG RCV Filter Parameters

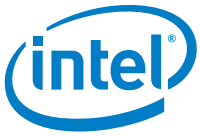
In order to read the MNG RCV filter parameters, the BMC should execute two SMB transactions. The first transaction is a block write that sets the parameter that the BMC wants to read. The second transaction is block read that reads the parameter.

This is the block write transaction:

Function	Command	Byte Count	Data 1	Data 2
Update MNG RCV filter parameters	0xCC	1 or 2	Parameter number (PN#)	Parameter data

The following table lists the different parameters and their contents:

Parameter	PN#	Parameter Data
Filters Enable	0x1	None
MNG2HOST configuration	0xA	None
Flex filter 0 enable MASK and length	0x10	None



Parameter	PN#	Parameter Data
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
Flex filter 1 enable MASK and length	0x20	None
Flex filter 1 data	0x21	Same as parameter 0x11 but for filter 1.
Flex filter 2 enable MASK and length	0x30	None
Flex filter 2 data	0x31	Same as parameter 0x11 but for filter 2.
Flex filter 3 enable MASK and length	0x40	None
Flex filter 3 data9	0x41	Same as parameter 0x11 but for filter 3.
Filters Valid	0x60	None
Decision Filters	0x61	1 byte to define the accessed manageability decision filter (MDEF) Data 2 – Decision Filter number
VLAN Filters	0x62	1 byte to define the accessed VLAN tag filter (MAVTV) Data 2 – VLAN Filter number
Flex Ports Filters	0x63	1 byte to define the accessed manageability flex port filter (MFUTP). Data 2 – Flex Port Filter number
IPv4 Filter	0x64	1 byte to define the accessed IPv4 address filter (MIPAF) Data 2 – IPv4 address filter number
IPv6 Filters	0x65	1 byte to define the accessed IPv6 address filter (MIPAF) Data 2 – IPv6 address filter number
MAC Filters	0x66	1 byte to define the accessed Ethernet MAC Address filters pair (MMAL, MMAH) Data 2 – Ethernet MAC Address filters pair number (0-3)
Wrong parameter request	0xFE	Returned by the X540 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 X540.
the X540 not ready	0xFF	Returned by the X540 only, on read parameters command when the data that should have been read is not ready. This parameter has no data.

Following the block write the BMC should issue a block read that will read the parameter that was set in the block write command:

Function	Command
Request MNG RCV filter parameters	0xCD

Data returned from the X540:

Function	Byte Count	Data 1 (Op-Code)	Data 2	Data 3	...	Data N
Read MNG RCV filter parameters	N	0xCD	Parameter number (PN#)	Parameter dependent		

The returned data is in the same format of the "update" command. 1

Note: If the parameter that is returned is not the parameter requested by the BMC, the BMC should verify the parameter number (default parameter to be returned is 1h).

Note: If the parameter number is 0xFF, it means that the data that the X540 should supply is not ready yet. The BMC should retry the read transaction.

Note: It is BMC's responsibility to follow the procedure defined above. sends a block read command (as described above) which is not preceded by a block write command with bytecount=1, the X540 will set Parameter Number in the read block transaction to be 0xFE.

11.7.2.2.7 Read Receive Enable Configuration

The BMC uses this command to read the receive configuration data. This data can be configured in receive enable command or through NVM loading upon power-up.

Read Receive Enable Configuration command format (SMBus Read Block Protocol):

Function	Command
Read receive enable	0xDA

Data returned from the X540:

Function	Byte Count	Data 1 (Op-Code)	Data 2	Data 3	...	Data 8	Data 9	...
Read receive enable	15 (0x0F)	0xDA	Receive control byte	Ethernet MAC Address MSB	...	Ethernet MAC Address LSB	IP address MSB	...

Function	Data 12	Data 13	Data 14	Data 15
Read receive enable	IP address LSB	BMC SMBus address	Interface data byte	Alert value byte

The detailed description of each field is specified in the receive enable command description in Section 11.7.2.1.3.

11.7.2.2.8 Read MACsec Parameters

The BMC uses this command to read the MACsec status and parameters.

Function	Command	Byte Count	Data 1	Data 2 — n
Read MACsec parameters	0xD9	2,18 or 22	Parameter number (PN#)	Parameter data

The following table lists the different parameters and their contents:

Parameter	PN#	Parameter Data
MACsec Interrupt Cause	0x0	This command shall return 1 byte (Data2). This byte contains the MACsec Interrupt cause, according to the following values: Data2: Bit 0 – TX Key Packet Number (PN) threshold met Bit 1 – Host requested ownership Bit 2 – Host released ownership Bit 3...31 — Reserved
MACsec RX parameters	0x1	Data 2: Reserved Data 3: MACsec Ownership Status. See Table 11-5 Data 4: MACsec Host Control Status. See Table 11-6 Data 5: RX Port Identifier (MSB) Data 6: RX Port Identifier (LSB) Data 7 : RX SCI (MSB) … Data 12: (LSB) Data 13: Reserved Data 14: RX SA AN — The association number currently used for the active SA Data 15: RX SA Packet Number (MSB) … Data 18: (LSB) RX SA Packet Number is the last packet number, as read from the last valid RX MACsec packet
MACsec TX parameters	0x2	Data 2: Reserved Data 3: MACsec Ownership Status. See Table 11-5 below Data 4: MACsec Host Control Status. See Table 11-6 below Data 5: TX Port Identifier (MSB) Data 6: TX Port Identifier (LSB) Note: TX Port Identifier is reserved to 0x0 for this implementation. Data 7 : TX SCI (MSB) … Data 12: (LSB) Data 13: Reserved Data 14: TX SA AN — The association number currently used for the active SA Data 15: TX SA Packet Number (MSB) … Data 18: (LSB) Data 19: Packet Number Threshold (MSB) … Data 21: (LSB) TX SA Packet Number is the last packet number, as read from the last valid TX MACsec packet. Data 22: TX Control Status. See Table 11-7 below

Table 11-5 MACsec Owner Status

Value	Description
0x0	Host is MACsec owner
0x1	BMC is MACsec owner

Table 11-6 MACsec Host Control Status

Bit	Description
0	Reserved
1	Allow Host Traffic: 0b = Host traffic is blocked 1b = Host traffic is allowed
2..7	Reserved

Table 11-7 TX Control Status:

Bit	Description
0..4	Reserved
5	Include SCI: 0b = Do not include SCI in TX packets 1b = Include SCI in TX packets
6..7	Reserved

11.7.2.2.9 Read Receive Enable Configuration Command

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 X540:

Function	Byte Count	Data 1 (Op-Code)	Data 2	Data 3	...	Data 8	Data 9	...	Data 12	Data 13	Data 14	Data 15
Read Receive Enable	15 (0x0F)	0xDA	Receive Control Byte	MAC Addr MSB	...	MAC Addr LSB	IP Addr MSB	...	IP Addr LSB	BMC SMBus Addr	I/F Data Byte	Alert Value Byte

The detailed description of each field is specified in the receive enable command description in Section 11.7.2.1.3.

11.7.2.3 SMB ARP Transactions

Note: All SMB-ARP transactions include PEC byte.

11.7.2.3.1 Prepare to ARP

This command will clear the Address Resolved flag (set to false). It will not affect the status or validity of the dynamic SMBus Address. It 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	

11.7.2.3.2 Reset Device (General)

This command will clear the Address Resolved flag (set to false). It will 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	

11.7.2.3.3 Reset Device (Directed)

The Command field is NACK-ed if the bits 7 through 1 do not match the current the X540 SMBus address.

It will clear the Address Resolved flag (set to false). It will 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	

11.7.2.3.4 Assign Address

This command assigns the X540 SMBus address. The address and command bytes are always acknowledged.

The transaction will be aborted immediately (NACK-ed in simple English) if any of the UDID bytes differ from the X540 UDID bytes. If successful, the MNG will update the SMBus address internally. This command will also set the Address Resolved flag 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	

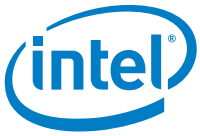
8	1	8	1	8	1	8	1	
Data-1	A	Data-2	A	Data-3	A	Data-4	A	•••
UDID byte 15 (MSB)	0	UDID byte 14	0	UDID byte 13	0	UDID byte 12	0	

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	



11.7.2.3.5 UDID Content

The Unique Device Identifier (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:

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 as follows	See as follows	0x8086	As reflected in the Device ID field in the PCI config space	0x0004/ 0x0024	0x0000	0x0000	See as follows
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: 0x1512.
- 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
MSB							LSB

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 as follows		
MSB							LSB

Silicon Revision ID:

Silicon Version	Revision ID
A0	000b
B0	001b

Vendor Specific ID: Four LSB bytes of the device Ethernet MAC address. The device Ethernet MAC address is taken from the NVM LAN Core 0/1 Modules in the NVM (see Section 6.4.6). Note that in the X540 there are two Ethernet MAC addresses (one for each port).

1 Byte	1 Byte	1 Byte	1 Byte
Ethernet MAC Address, Byte 3	Ethernet MAC Address, Byte 2	Ethernet MAC Address, Byte 1	Ethernet MAC Address, Byte 0
MSB			LSB

11.7.2.3.6 Get UDID (General and Directed)

The Get UDID command depends on whether this is a directed or general command.

The General Get UDID SMBus transaction will support a constant command value of 0x03.

The Directed Get UDID SMBus transaction will support a dynamic command value equal to the dynamic SMBus address with the LSB bit set.

Note: Bit 0 (LSB) of Data byte 17 will always be 1b.

If the SMBus Address has been resolved (Address Resolved flag is true); for a general command the MNG will not acknowledge (NACK) this transaction, for a directed command the MNG will always acknowledge (ACK) this transaction.

This command will not affect the status or validity of the dynamic SMBus Address nor of the Address Resolved flag.

The command returns the UDID bytes as defined in Section 3.2.7.

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	

11.7.3 MCTP Programming

The MCTP programming model is based on:

1. A set of MCTP commands used for the discovery process and for the link management. The list of supported commands is described in section Section 11.7.3.1.
2. A subset of the NC-SI commands used in the regular NC-SI interface, including all the OEM commands as described in Section 11.7.1 (NC-SI programming I/F). The specific commands supported are listed in Table 11-1 and Table 11-1.

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.

11.7.3.1 MCTP Commands Support

Table 11-1 lists the MCTP commands supported by the X540.

Table 11-1. MCTP Commands Support

Command Code	Command Name	General Description	the X540 support as Initiator	the X540 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 for details.	No	Yes
0x03	Get Endpoint UUID	Retrieves a per-device unique UUID associated with the endpoint. See Section 11.7.3.1.2 for details.	No	Yes
0x04	Get MCTP Version Support	Lists which versions of the MCTP control protocol are supported on an endpoint. See Section 11.7.3.1.3 for details.	No	Yes
0x05	Get Message Type Support	Lists the message types that an endpoint supports. See Section 11.7.3.1.4 for details.	No	Yes
0x06	Get Vendor Defined Message Support	Used to discover an MCTP endpoint's vendor specific MCTP extensions and capabilities.	No	No
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

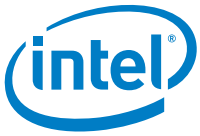


Table 11-1. MCTP Commands Support

Command Code	Command Name	General Description	the X540 support as Initiator	the X540 support as Responder
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.	No	N/A
0x0E	Reserved	Reserved	–	–
0x0F	Query Hop	Used to discover what bridges, if any, are in the path to a given target endpoint and what transmission unit sizes the bridges will pass for a given message type when routing to the target endpoint.	No	No

1. These commands are supported only for MCTP over PCIe.

11.7.3.1.1 Get Endpoint ID

The Get Endpoint ID response of the X540 is described in the following table:

Byte	Description	Value
1	Completion Code	
2	Endpoint ID	0x00 - EID not yet assigned Otherwise - returns EID assigned using Set Endpoint ID command
3	Endpoint Type	0x00 (Dynamic EID, Simple Endpoint)
4	Medium Specific	SMBUs: 0x01 - Fairness arbitration protocol supported. PCIe: 0x00

11.7.3.1.2 Get Endpoint UUID

The UUID returned is calculated according to the following function:

Time Low = Read from NVM words at offset 0x9 and 0xA of Sideband Configuration Structure.

Time mid = Read from NVM word at offset 0xB of Sideband Configuration Structure

Time High and version = Read from NVM word at offset 0xC of Sideband Configuration Structure

Clock Sec and Reserved = Read from NVM word at offset 0xD of Sideband Configuration Structure

Node = Host MAC address of port 0 as stored in NVM.

11.7.3.1.3 Get MCTP Version Support

The following table describes the returned value according to the requested message type.

Byte	Description	Message Type			
		0xFF(Base)	0x00 (Control Protocol Message)	0x02 (NC-SI over MCTP)	All Other
1	Completion Code				0x80
2	Version Number entry count	1	1	1	0
6:3	Version number entry	0xF1F0FF00	0xF1F0FF00	1	0

11.7.3.1.4 Get Message Type Support Command

The Get Message type support response of the X540 is described in the following table:

Byte	Description	Value
1	Completion Code	0x00
2	MCTP Message Type Count	0x01 - The X540 supports one additional message type
3	List of Message Type numbers	0x02 (NC-SI over MCTP)

11.7.3.1.5 Set Endpoint ID Command

The X540 supports the Set EID and Force EID operations defined in the Set Endpoint ID command . As endpoints in the X540 can be set only through their own interface, Set EID and Force EID are equivalent. The Reset EID and Set Discovered Flag operations are not relevant to the X540.

The Set Endpoint ID response of the X540 is described in the following table:

Byte	Description	Value
1	Completion Code	0x00
2	Completion Status	[7:6] = 00 - Reserved
		[5:4] = 00 - EID assignment accepted
		[3:2] = 00 - Reserved
		[1:0] = 00 - Device does not use an EID pool.

11.7.4 Manageability Host Interface

11.7.4.1 HOST CSR Interface (Function 1/0)

The software device driver of function 0/1 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 function 0 or from function 1 is targeted to the same address in the RAM.

11.7.4.2 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 ARC data RAM: The internal data RAM is mapped to address space 0x15800 to 0x15EFF. Writing/reading to this address space goes directly to the RAM.
- Control registers are located at address 0x15F00.

11.7.4.3 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 control register.

The following flow shows the process of initiating a command to the manageability block:

1. The software device driver reads the control register and checks that the *Enable* bit is set.
2. The software device driver writes the relevant command block into the RAM area.
3. The software device driver sets the *Command* bit in the control register. Setting this bit causes an interrupt to the ARC (can be masked).
4. The software device driver polls the control register for the command bit to be cleared by hardware.
5. When manageability finishes with the command, it clears the command bit (if the manageability should reply with data, it should clear the bit only after the data is in the RAM area where the software device driver can read it).

If the software device driver reads the control register and the *SV* bit is set, then there is a valid status of the last command in the RAM. If the *SV* bit is not set, then the command has failed with no status in the RAM.

11.7.4.4 Host Slave Command Registers

The Host Slave Command registers (listed below) are described in the Manageability Host Interface Registers section.:

Host Interface Control Register — CSR Address 0x15F00; AUX 0x0700

Firmware Status 0 (FWS0R) Register — CSR Address 0x15F0C; AUX 0x0702

Software Status Register — CSR Address 0x15F10; AUX 0x0703

11.7.4.5 Host Interface Command Structure

The following table describes the structure used by the host driver to send a command to manageability FW via the host interface slave command interface:

#Byte	Description	Bit	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	Default/ Implicit Interface	0	Command Dependent	Used for commands might refer to one of two interfaces (LAN or SMBus). 0b = Use default interface. 1b = Use specific interface.
	Interface Number	1	Command Dependent	Used when bit 0 (Default/Implicit interface) is set: 0b = Apply command for interface 0. 1b = Apply command for interface 1. When bit 0 is set to 0b, it is ignored.
	Reserved	7:2	0x0	Reserved
3	Checksum	7:0	Defined Below	Checksum signature.
255:4	Data Buffer	7:0	Command Dependent	Command Specific Data Minimum buffer size: 0. Maximum buffer size: 252.

11.7.4.6 Host Interface Status Structure

The following table lists the structure used by manageability firmware to return a status to the host driver via the host interface slave command interface. A status is returned after a command has been executed.

#Byte	Description	Bit	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	Defined in commands 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.

11.7.4.7 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.

11.7.4.8 Host Slave Interface Commands

11.7.4.8.1 Driver Info Host Command

This command is used to provide the driver information in NC-SI mode.

Byte	Name	Bit	Value	Description
0	Command	7:0	0xDD	Software driver information command.
1	Buffer Length	7:0	0x5	64 bytes of software driver information.
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 currently reporting its software driver information.
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

Following is the status returned on this command:

Byte	Name	Bit	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	0x1 for good status.
3	Checksum	7:0		Checksum signature.

11.7.5 SW and FW synchronization

SW and FW synchronize accesses to shared resources in the X540 through a semaphore mechanism and a shared configuration register between the host interface of the two

ports and the FW. This semaphore enables synchronized accesses to the following shared resources:

- NVM
- PHY 0 and PHY 1 registers
- MAC (LAN controller) shared registers

The SW_FW_SYNC.REGSMP bit is used as a semaphore mechanism between SW and FW. Once SW or FW takes control over this semaphore flag, it can access the SW_FW_SYNC Register and claim ownership of the specific resources. The SW_FW_SYNC includes pairs of bits (one owned by SW and the other by FW), while each pair of bits control a different resource. A resource is owned by SW or FW when the respective bit is set. It is illegal to have both bits in a pair set at the same time. There is also one bit owned by HW to control its access to the NVM resource. When NVM ownership is taken by HW, SW or FW shall not access NVM. Described below are the required sequences for gaining and releasing control over the shared resources:

Gaining Control of shared resource by the Software

- The SW checks that the SW on the other LAN function does not use the SW/FW semaphore
  - SW polls the SWSM.SMBI bit till it is read as ‘0’ or time expires (recommended expiration is ~10msec + expiration time used for the SW_FW_SYNC.REGSMP).
  - If the SWSM.SMBI is found at ‘0’, the semaphore is taken. Note that following this read cycle the hardware auto set the bit to ‘1’.
  - If time expired, it is assumed that the SW of the other function malfunctions. The SW proceeds to the next steps.
- The SW checks that the FW does not use the SW/FW semaphore and then takes its control
  - SW polls the SW_FW_SYNC.REGSMP bit till it is read as ‘0’ or time expires (recommended expiration is ~3sec). If time has expired the SW assumes that the FW malfunctions and proceeds to the next step while ignoring the FW bits in the SW_FW_SYNC register.
- The SW takes control of the requested resource(s)
  - The SW reads the FW and SW bit(s) of the requested resource(s) in the SW_FW_SYNC register. For the NVM resource it shall read also the HW bit.
  - If the bit(s) are cleared it means that the resource(s) is accessible (i.e. no other entity owns the resource(s)). In this case the SW sets the SW bit(s) of the requested resource(s) in the SW_FW_SYNC register. Then the SW clears the SW_FW_SYNC.REGSMP and SWSM.SMBI bits (releasing the SW/FW semaphore register) and can use the specific resource(s).
  - Otherwise (i.e., either FW or HW or the SW of the other LAN function owns the resource), the SW clears the SW_FW_SYNC.REGSMP and SWSM.SMBI bits and then repeats the whole process after some delay (recommended 5-10msec).
    - If the resources are not released by the SW of the other LAN function long enough (recommended expiration time is ~1sec) the SW can assume that the other SW malfunctions. In that case the SW should clear all SW flags that it does not own (including SW_FW_SYNC.REGSMP bit) and then repeat the whole process once again.

- If the resource is not released by the FW/HW (recommended expiration time for FW is ~3sec and ~1sec for HW) the SW can assume that the FW/HW malfunctions. In that case the SW should sets the SW bit(s) of the requested resource(s) while ignoring the corresponding FW/HW bits in the SW_FW_SYNC register.

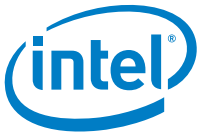
Note that the FW initializes its semaphore flags as part of its init flow

Releasing a shared resource by the SW

- The SW takes control over the SW/FW semaphore as described above for gaining shared resources.
- The SW clears the bit(s) of the released resource(s) in the SW_FW_SYNC register.
- The SW releases the SW/FW semaphore by clearing the SW_FW_SYNC.REGSMP and SWSM.SMBI bits
- The SW should wait a minimum delay (recommended 5-10msec) before trying to gain the semaphore again

Gaining Control of shared resource by the FW

- The FW takes control over the SW/FW semaphore (SW_FW_SYNC register)
  - The FW polls the SW_FW_SYNC.REGSMP bit till it is read as '0' or time is expired (recommended expiration time is ~10msec).
  - If time has expired the FW clears the SW_FW_SYNC.REGSMP bit (assuming the SW does not function well).
- The FW takes ownership of the requested resources
  - The FW reads the matched SW bit(s) to the requested resource(s) in the SW_FW_SYNC register. For the NVM resource it shall read also the HW bit.
  - If the SW and HW bit(s) are cleared (i.e., SW or HW does not own the resource), the FW sets the FW bit(s) of the requested resource(s). Then the FW clears the SW_FW_SYNC.REGSMP bit (releasing the SW/FW semaphore) and can use the specific resource(s).
  - Otherwise (i.e., SW owns the resource), the FW clears the SW_FW_SYNC.REGSMP bit and then repeats the above process after some delay (recommended delay of 5-10msec).
    - If the resources owned by SW are not released long enough (~1sec) the FW accesses "by force" the requested resources. The FW also clears the SW flags of the requested resources in the SW_FW_SYNC register (assuming the SW that set those flags malfunction).
    - If the resource (NVM) is not released by the HW (recommended expiration time is ~1sec) the FW can assume that the HW malfunctions. In that case the FW should sets the SW bit of the NVM resource while ignoring the corresponding HW bit in the SW_FW_SYNC register.



Releasing a shared resource by the FW

- The FW takes control over the SW/FW semaphore as described above for gaining shared resources.
- The FW clears the bit(s) of the selected resource(s) in the SW_FW_SYNC register.
- The FW releases the SW/FW semaphore by clearing the SW_FW_SYNC.REGSMP bit
- The FW should wait some delay before trying to gain the semaphore once again (recommended 5-10msec)

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

12.0 Electrical / Mechanical Specification

12.1 Introduction

This section describes the X540 DC and AC (timing) electrical characteristics and the X540 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 X540.

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		-500	3600	mV
VCC2P5		-500	2750	mV
VCC1P2		-500	1320	mV
VCC0P8		-500	1100	mV
VCC0P67		-500	1100	mV
V_{ESD}	ESD On All Pins Except Line		2	kV
V_{CD}	Cable Discharge On Line		2	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
Ta	Operating Temperature Range Commercial (Ambient; 0 CFS airflow)	0		55	°C
Tj	Junction Temperature			110	°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 Section 14.0 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 Section 14.0).
- Total Line Regulation — The sum of DC and AC voltage regulation.

12.3.2 Power Supply Specifications

Description	Parameter
VCC3P3V	
Nominal Voltage	3.3V
DC Voltage Regulation	± 5% DC Regulation

Description	Parameter
AC Voltage Regulation	± 4% AC Regulation (132 mV Pk-Pk)
Total Line Regulation	± 9%
VCC25	
Nominal Voltage	2.5V
DC Voltage Regulation	± 2% DC Regulation
AC Voltage Regulation	± 3% AC Regulation (75 mV Pk-Pk)
Total DC/AC Voltage Regulation	± 5%
Step Load Size	0.75 A
Step Load Slew Rate di/dt	1 A/ 1 µs
Maximum Step Load Slew Rate di/dt	
VCC1P2	
Nominal Voltage	1.2V
DC Voltage Regulation	± 2% DC Regulation
AC Voltage Regulation	± 3% AC Regulation (36 mV Pk-Pk)
Total DC/AC Voltage Regulation	± 5%
Step Load Size	0.9 A
Step Load Slew Rate di/dt	1 A/ 1 µs
VCC0P8	
Nominal Voltage	0.8V
DC Voltage Regulation	± 2% DC Regulation
AC Voltage Regulation	± 3% AC Regulation (24 mV Pk-Pk)
Total DC/AC Voltage Regulation	± 5%
Step Load Size	4 A
Step Load Slew Rate di/dt	1 A/1 µs
VCC0P67	
Nominal Voltage	0.67V
DC Voltage Regulation	± 2% DC Regulation

Description	Parameter
AC Voltage Regulation	± 3% AC Regulation (20 mV Pk-Pk)
Total DC/AC Voltage Regulation	± 5%
Step Load Size	2.4 A
Step Load Slew Rate di/dt	1 A/1 µs

12.3.3 VCC3P3 External Power Supply Specification (3.3V)

Parameter	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark	0.1	5	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)	480	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. (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 VCC2P5 External Power Supply Specification (2.5V)

Title	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark	0.1	5	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)	380	21000	V/S

Operational Range	Voltage range for normal operating conditions	2.38	2.63	V
Overshoot	Maximum overshoot allowed	N/A	100	mV
Overshoot Settling Time	Maximum overshoot allowed duration. (At that time delta voltage should be lower than 5 mV from steady state voltage)	N/A	0.1	ms
Suggested Decoupling Capacitance	Capacitance range	174	N/A	µF

12.3.5 VCC1P2 External Power Supply Specification (1.2V)

Title	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark	0.1	5	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)	182	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. (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	98	N/A	µF

12.3.6 VCC0P8 External Power Supply Specification (0.8V)

Title	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark	0.1	5	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)	121	6720	V/S
Operational Range	Voltage range for normal operating conditions	0.76	0.85	V
Overshoot	Maximum overshoot allowed	N/A	40	mV
Overshoot Duration	Maximum overshoot allowed duration. (At that time delta voltage should be lower than 5 mV from steady state voltage)	N/A		ms
Suggested Decoupling Capacitance	Capacitance range	174	N/A	µF

12.3.7 VCC0P67 External Power Supply Specification (0.67V)

Title	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark	0.1	5	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)	102	5460	V/S
Operational Range	Voltage range for normal operating conditions	0.64	0.7	V
Overshoot	Maximum overshoot allowed	N/A	100	mV
Overshoot Duration	Maximum overshoot allowed duration. (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	174	N/A	µF

12.3.8 Power On/Off Sequence

The following power-up sequence is recommended for the X540.

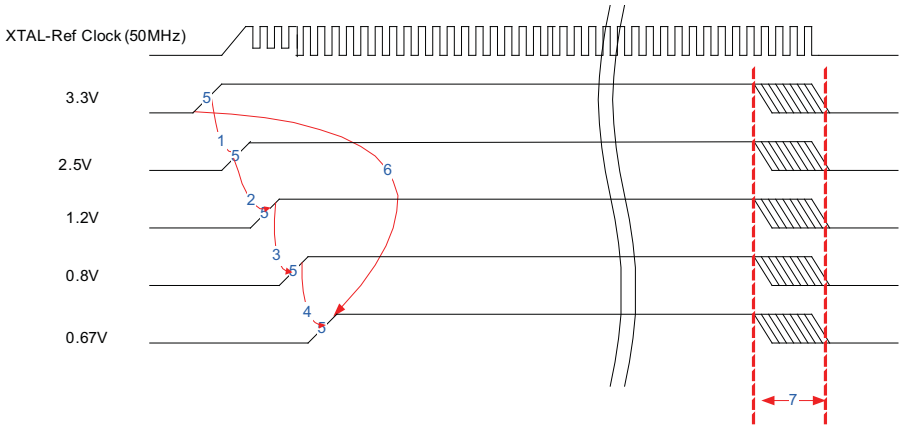


Figure 12-1 Power On/Off Sequence Flow

Table 12-1 Notes for Power On/Off Sequence Diagram

Note	Description
1	The 2.5V rail does not start to ramp before the 3.3V rail is 80% of its final value.
2	The 1.2V rail does not start to ramp before the 2.5V rail is 80% of its final value. Tmax (2.5V to 1.2V) should be less than 5 ms.
3	The 0.8V rail does not start to ramp before the 1.2V rail is 80% of its final value.
4	The 0.67V rail does not start to ramp before the 0.8V rail is 80% of its final value. Tmax (0.8V to 0.67V) should be less than 5 ms.
5	Tramp for each power rail: 0.1 ms < Tramp < 5 ms.
6	Total power-up time from the start of the 3.3V rail raising until the 0.67V rail gets to its final level is <20 ms.
7	When powering off, 2.5V, 1.2V, and 0.8V must get to a 50% level within 20 ms from the point the first power rail starts powering off. 3.3V and 0.67V must get to a 50% level within 200 ms from the point the first power rail starts powering off. Minimum time between Power off to power on 1 second.

12.3.9 Power On Reset

Symbol	Parameter	Specification			Units
		Min	Typ	Max	
2.5V	Threshold for 2.5 Vdc supply	1.81		2.14	Vdc
1.2V	Threshold for 1.2 Vdc supply	0.86		1.05	Vdc
0.8V	Threshold for 0.8 Vdc supply	0.57		0.66	Vdc
LAN_PWR_Good	High-threshold (VIH)	0.83		3.6	Vdc
LAN_PWR_Good	Low-threshold (VIL)	0.3		0.36	Vdc

12.4 DC/AC Specifications

12.4.1 Current Consumption

Power numbers are based on measurement data.

- Maximum power mode: 3 sigma FAST material, Vnom, Tj-max (110 °C).
- Other operational modes: Typical material, Vnom.

Table 12-2 X540-AT2 Steady State Current Consumption

Note: Current consumption values listed in Table 12-2 reflect the X540 dual-port configuration using the X540 dual-port NVM/EEPROM.

X540 Mode	3.3V		2.5V		1.2V		0.8V		0.67V		Device Total
	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Power (W)
10 GbE MAX	17	56	1300	3250	1460	1750	5040	4030	5150	3450	12.5
10 GbE Max 30 m reach	17	56	1330	3325	1600	1920	4950	3960	3850	2580	11.8
10 GbE Typical	17	56	1300	3250	1460	1750	4500	3600	4180	2800	11.46
10 GbE (idle)	17	56	1300	3250	1458	1750	4375	3500	4179	2800	11.2
10 GbE 30mreach	17	56	1330	3325	1600	1920	4400	3520	2900	1940	10.76
1 GbE	17	56	1052	2630	1358	1630	2313	1850	0	0	6
100 Mb/s	17	56	600	1500	917	1100	1938	1550	0	0	4
1 GbE WoL	17	56	1020	2550	1167	1400	1438	1150	0	0	5
100 Mb/s WoL	17	56	580	1450	725	870	1500	1200	0	0	3.5
Standby (Dr, No-MNG, No-WoL)	10	33	400	1000	550	660	1000	800	0	0	2.5

Table 12-3 X540-BT2 Steady State Current Consumption

Note: Current consumption values listed in Table 12-3 reflect the X540 dual-port configuration using the X540 dual-port NVM/EEPROM.

X540 Mode	3.3V		2.5V		1.2V		0.8V		0.67V		Device Total
	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Power (W)
10 GbE MAX	17	56	1300	3250	1460	1752	6250	5000	6000	4020	14
10 Gbe Max 30m reach	17	56	1330	3325	1660	1992	6250	5000	4500	3015	13.4
10 GbE Typical	17	56	1300	3250	1440	1728	5350	4280	5300	3551	12.85
10 GbE (idle)	17	56	1300	3250	1440	1728	5150	4120	5300	3551	12.7
10 GbE30mreach	17	56	1330	3325	1660	1992	5350	4280	3860	2586	12.25
1 GbE	17	56	1050	2625	1360	1632	2600	2080	0	0	6.4
100 Mb/s	17	56	600	1500	920	1104	2100	1680	0	0	4.35
1 GbE WoL	17	56	1020	2550	1170	1404	2400	1920	0	0	5.95
100 Mb/s WoL	17	56	580	1450	725	870	1750	1400	0	0	3.78
Standby (Dr, No-MNG, No-WoL)	10	33	400	1000	550	660	1400	1120	0	0	2.82

Table 12-4 X540-BT2 Steady State Current Consumption Using Single-port NVM

Note: Current consumption values listed in Table 12-4 reflect the X540 single-port configuration using the single-port NVM/EEPROM.

X540 Mode	3.3V		2.5V		1.2V		0.8V		0.67V		Device Total
	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Power (W)
10 GbE MAX	17	56	800	2000	1000	1200	4150	3320	3250	2178	8.5
10 Gbe Max 30 m reach	17	56	810	2025	1020	1224	4150	3320	2550	1709	8.3
10 GbE Typical	17	56	800	2000	980	1176	3350	2680	2750	1843	7.7
10 GbE Typical30 m reach	17	56	810	2025	1000	1200	3350	2680	2050	1374	7.3
1 GbE	17	56	700	1750	945	1134	1660	1328	0	0	4.27
100 Mb/s	17	56	440	1100	710	852	1500	1200	0	0	3.2
1 GbE WoL	17	56	690	1725	750	900	1580	1264	0	0	3.95
100 Mb/s WoL	17	56	420	1050	520	624	1400	1120	0	0	2.85
Standby (Dr, No-MNG, No-WoL)	10	33	340	850	440	528	1400	1120	0	0	2.53

12.4.2 Peak Current Consumption

Table 12-5 X540-AT2/BT2 Aux Power On Peak Current Consumption

Conditions: 3 sigma FAST material, Vnom, Tj = 85 °C, and 100 Mb/s link.

Configuration	3.3V	2.5V	1.2V	0.8V	0.67V
Dual Port	54 mA	1.6 A	1.5 A	5 A	2.5 A
Single Port	54 mA	1.2 A	800 mA	5 A	2.5 A

Table 12-6 X540-AT2/BT2 Main Power On Peak Current Consumption During 10GBASE-T Training

Conditions: 3 sigma FAST material, Vnom, Tj-max (110 °C), and 10 Gb/s link.

Configuration	3.3V	2.5V	1.2V	0.8V	0.67V
Dual Port	100 mA	2 A	2 A	8 A	8 A

12.4.3 Digital Functional 2.5V I/O DC Electrical Characteristics

Table 12-7 Digital I/O LAN_PWR_GOOD

Symbol	Parameter[1]	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=2.5V or 0V					µA
IOZ	Tri-state Output Leakage Current at VO=2.5V or 0V					µA
PD	Internal Pull Down		51	60	69	KΩ
PU	Internal Pull Up		51	60	69	KΩ

1. Parameters listed are 3.3V tolerant.

Table 12-8 Digital I/O RSVD, LED, SDP

Symbol	Parameter[1]	Conditions	Min	Max	Units	Note
VOH	Output High Voltage	IOH = -12 mA; VCC2P5 = minimum	1.7		V	
VOL	Output Low Voltage	IOL = 12 mA; VCC2P5 = minimum		0.7	V	
IOL	Low Level Output Voltage	VOL (maximum)	16	42	mA	
IOH	High Level Output Voltage	VOH (minimum)	15	50	mA	
VIH	Input High Voltage		1.7	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	
PD	Internal Pull Down		34	101	KΩ	
Cin	Pin Capacitance			5	pF	
Cload	Pin Capacitance			5	pF	

1. Parameters listed are 3.3V tolerant.

Note: Table 12-8 applies to RSVDL4_NC, RSVDL3_NC, RSVDL21_NC, RSVDL22_NC, RSVDM21_NC, RSVDM22_NC, RSVDM4_NC, RSVDM3_NC, RSVDN21_NC, RSVDN22_NC, RSVDN4_NC, RSVDN3_NC, RSVDP21_NC, RSVDP22_NC, RSVDP4_NC, and RSVDR21_NC, RSVDR22_NC, SEC_EN, LED0_[3:0], LED1_[3:0], SDP0_[3:0], and SDP1_[3:0].

Note: Digital I/O is 3.3V input tolerance.

Table 12-9 Digital I/O Flash

Symbol	Parameter[1]	Conditions	Min	Max	Units	Note
VOH	Output High Voltage	IOH = -12 mA; VCC2P5 = minimum	1.7		V	
VOL	Output Low Voltage	IOL = 12 mA; VCC2P5 = minimum		0.7	V	
IOL	Low Level Output Current (12 mA)	VOL (maximum)	16	42	mA	
IOH	High Level Output Current (12 mA)	VOH (minimum)	15	50	mA	
IOL	Low Level Output Current (8 mA)	VOL (maximum)	16	42	mA	
IOH	High Level Output Current (8 mA)	VOH (minimum)	15	50	mA	
VIH	Input High Voltage		1.7	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		35	90	KΩ	
Cin	Pin Capacitance			5	pF	
Cload	Pin Capacitance			5	pF	

1. Parameters listed are 3.3V tolerant.

Note: Table 12-9 applies to FLSH_SO, FLSH_SI, FLSH_CE_N, FLSH_SCK.

Note: Digital I/O is 3.3V input tolerance.

Table 12-10 Digital I/O JTAG

Symbol	Parameter[1]	Conditions	Min	Max	Units	Note
VOH	Output High Voltage	IOH = -12 mA; VCC2P5 = minimum	1.7		V	
VOL	Output Low Voltage	IOL = 12 mA; VCC2P5 = minimum		0.7	V	
IOL	Low Level Output Current (12 mA)	VOL (maximum)	16	42	mA	
IOH	High Level Output Current (12 mA)	VOH (minimum)	15	50	mA	
VIH	Input High Voltage		1.7	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		35	90	KΩ	
PD	Internal Pull Down		34	101	KΩ	
Cin	Pin Capacitance			5	pF	
Cload	Pin Capacitance			5	pF	

1. Parameters listed are 3.3V tolerant.

Note: Table 12-10 applies to TDO, TDI, TMS, TCK. TRST_N.

Note: Digital I/O is 3.3V input tolerance.

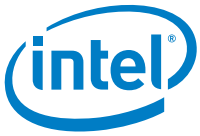
Table 12-11 Digital I/O Miscellaneous, PE_WAKE_N, and PE_RST_N

Symbol	Parameter[1]	Conditions	Min	Max	Units	Note
VOH	Output High Voltage	IOH = -12 mA; VCC2P5 = minimum	1.7		V	
VOL	Output Low Voltage	IOL = 12 mA; VCC2P5 = minimum		0.7	V	
IOL	Low Level Output Current (12 mA)	VOL (maximum)	16	42	mA	
IOH	High Level Output Current (12 mA)	VOH (minimum)	15	50	mA	
IOL	Low Level Output Current (8 mA)	VOL (maximum)	16	42	mA	
IOH	High Level Output Current (8 mA)	VOH (minimum)	15	50	mA	
VIH	Input High Voltage		1.7	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		35	90	KΩ	
Cin	Pin Capacitance			5	pF	
Cload	Pin Capacitance			5	pF	

1. Parameters listed are 3.3V tolerant.

Note: Table 12-11 signals SEC_EN, PE_WAKE_N and PE_RST_N apply to AUX_PWR, MAIN_PWR_OK, LAN1_DIS_N, LAN0_DIS_N.

Digital I/O is 3.3V input tolerance.



12.4.4 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 = 0.4V	4		mA	
Cin	Input Pin Capacitance			7	pF	[3]
Cload	Maximum Load Pin Capacitance			30	pF	[3]
Ioffsmb	Input leakage Current	VCC3P3 off or floating		±10	µA	[2]

Notes:

1. Section 12.4.4 applies to SMBD, SMBCLK, SMBALRT _N, PE_WAKE_N.
2. Device must meet this specification whether powered or unpowered.
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=0.4V at 16 mA, VOL max=0.2V at 0.1 mA.

The buffer specification meets the SMBus specification requirements defined at: www.smbus.org.

12.4.5 NC-SI I/O DC Specification

Parameter	Symbol	Conditions	Min.	Typ.	Max	Units
Bus High Reference	Vref[1]		3.0	3.3	3.46	V
Signal Voltage Range	Vabs		-0.3		3.765	V
Input Low Voltage	ViL				0.8	V
Input High Voltage	ViH		2.0			V
Output Low Voltage	VoL	IoL = 4 mA, Vref= $Vref_{min}$	0		0.4	V
Output High Voltage	VoH	IoL = -4 mA, Vref= $Vref_{min}$	2.4		Vref	V

Parameter	Symbol	Conditions	Min.	Typ.	Max	Units
Input High Current	IiH	Vin = 3.6V, Vref = 3.6V	0		200	µA
Input Low Current	IiL	Vin = 0V, $Vref_{min}$ to $Vref_{max}$	-20		0	µA
Clock Midpoint Reference Level	Vckm				1.4	V
Leakage Current for Output Signals in High-Impedance State	Iz	$0 \le Vin \le Vih_{max}$ @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. In order 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 spec define the Vrefmax up to 3.6V, the X540 supports the Vrefmax up to 3.46V (3.3V +5%).
2. Section 12.4.5 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*.

12.4.6 Digital I/F AC Specifications

12.4.6.1 Digital I/O AC Electrical and Timing Characteristics

12.4.6.2 Digital 2.5V I/O AC Specifications — JTAG AC Specifications

Table 12-12 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 12-12 applies to JTCK, JTMS, JTDI and JTDO.
2. Timing measured relative to JTCK reference voltage of VCC2P5/2.

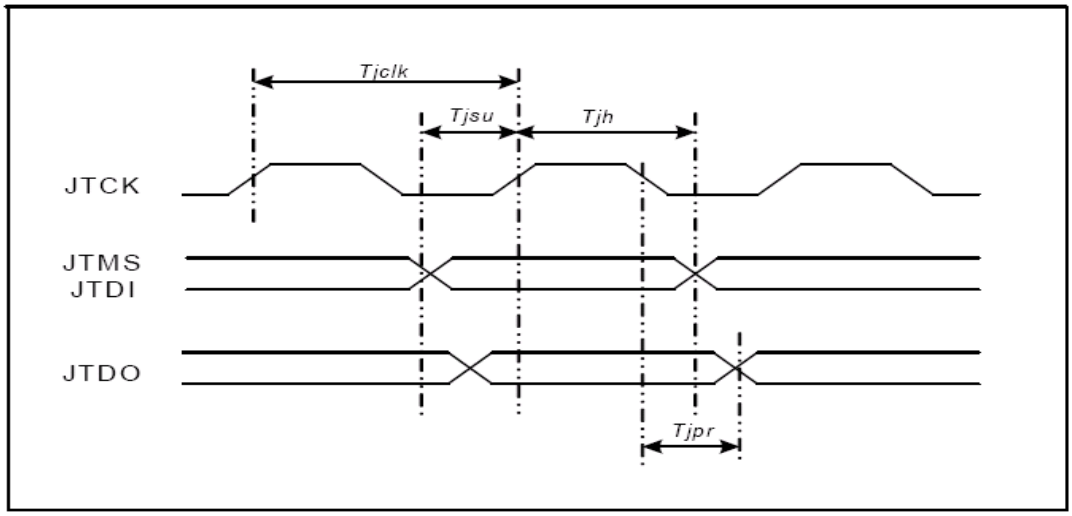


Figure 12-2 JTAG AC Timing Diagram

12.4.6.3 SMBus AC Specifications

The X540 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 X540 also supports a 400 KHz SMBus (as an input clock from the MC and as a slave).The X540 meets the100 KHz and 400 KHz specifications listed in Table 12-1.

Table 12-1 SMBus Timing Parameters (Master Mode)

Symbol	Parameter	Min	Typ	Max	Units
F_{SMB}	SMBus Frequency		84	100	kHz
T_{BUF}	Time between STOP and START condition driven by the X540	4.7	6.56		µs
$T_{HD:STA}$	Hold time after Start Condition. After this period, the first clock is generated.	4	6.72		µs
$T_{SU:STA}$	Start Condition setup time	4.7			µs
$T_{SU:STO}$	Stop Condition setup time	4	6.88		µs
$T_{HD:DAT}$	Data hold time	0.3	0.48		µs
$T_{SU:DAT}$	Data setup time	0.25			µs
$T_{TIMEOUT}$	Detect SMBClk low timeout	26.2		31.5	ms
T_{LOW}	SMBClk low time	4.7	5.76		µs
T_{HIGH}	SMBClk high time	4	6.56		µs

Table 12-1 SMBus Timing Parameters (Slave Mode)

Symbol	Parameter	Min	Typ	Max	Units
F_{SMB}	SMBus Frequency	10		400	KHz
T_{BUF}	Time Between STOP and START	1.44[1]			µs
$T_{HD,STA}$	Hold Time After Start Condition. After This Period, the First Clock is Generated.	0.48[1]			µs
$T_{SU,STA}$	Start Condition Setup Time	1.6[1]			µs
$T_{SU,STO}$	Stop Condition Setup Time	1.76[1]			µ
$T_{HD,DAT}$	Data Hold Time	0.32			µs
T_{LOW}	SMBCLK Low Time	0.8[1]			µs
T_{HIGH}	SMBCLK High Time	1.44[1]			µs

1. The actual minimum requirement has to be less. Many of these values are below the minimums specified by the SMBus specification.

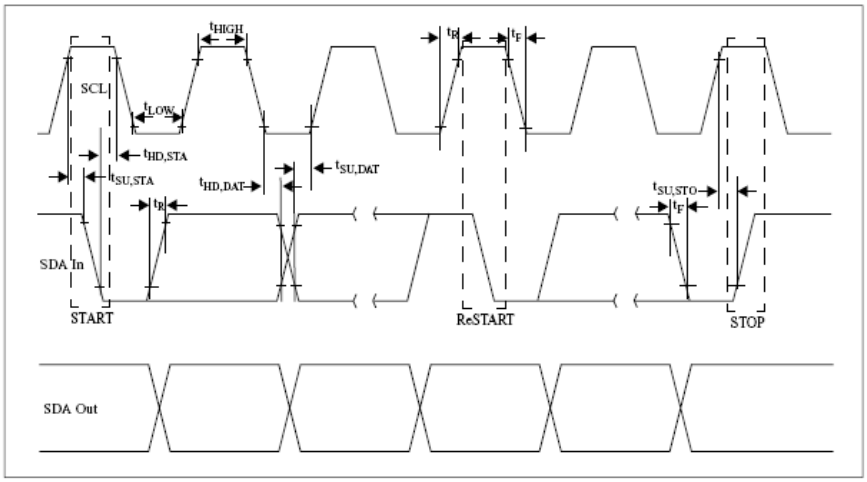


Figure 12-1 SMBus I/F Timing Diagram

12.4.6.4 Flash AC Specification

The X540 is designed to support a serial Flash. For Flash I/F timing specifications, see Table 12-2 and Figure 12-2.

Table 12-2 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

Table 12-2 Flash I/F Timing Parameters from Supported Flash Devices

Symbol	Parameter	Min	Max	Units
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

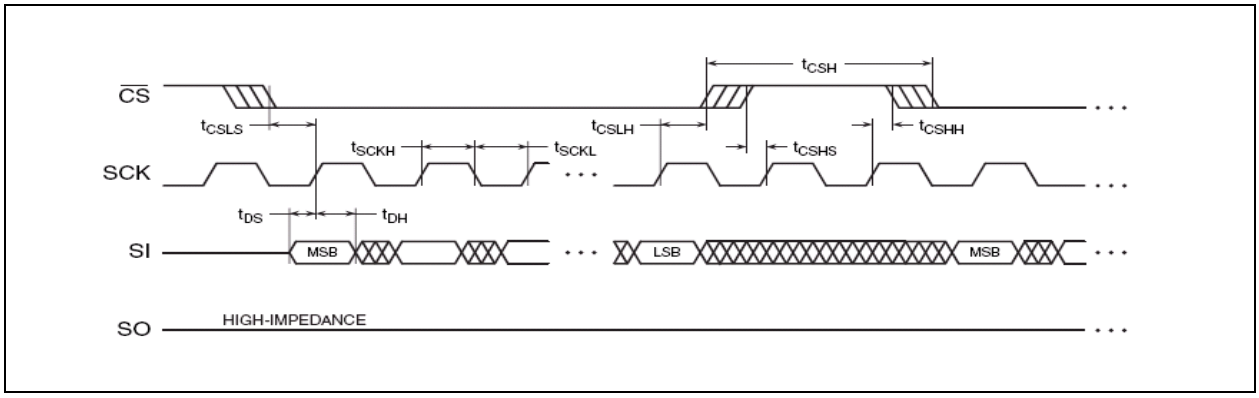


Figure 12-2 Serial Input Timing

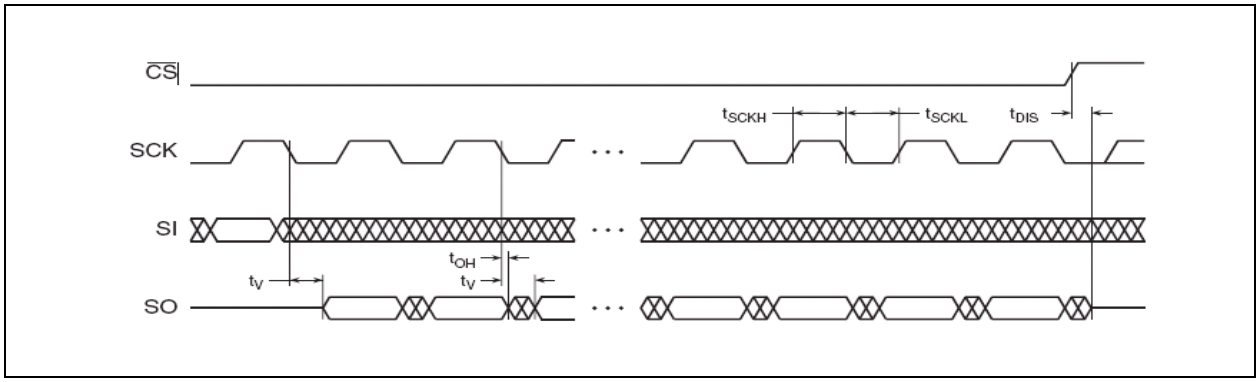


Figure 12-3 Serial Output Timing

12.4.6.5 NC-SI AC Specifications

The X540 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 12-3 and Figure 12-4.

Table 12-3 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 12.4.5). 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 12.4.5). 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 12-4). The middle points (50% of Vref) are marked as Vckm and Vm for clock and data, respectively.

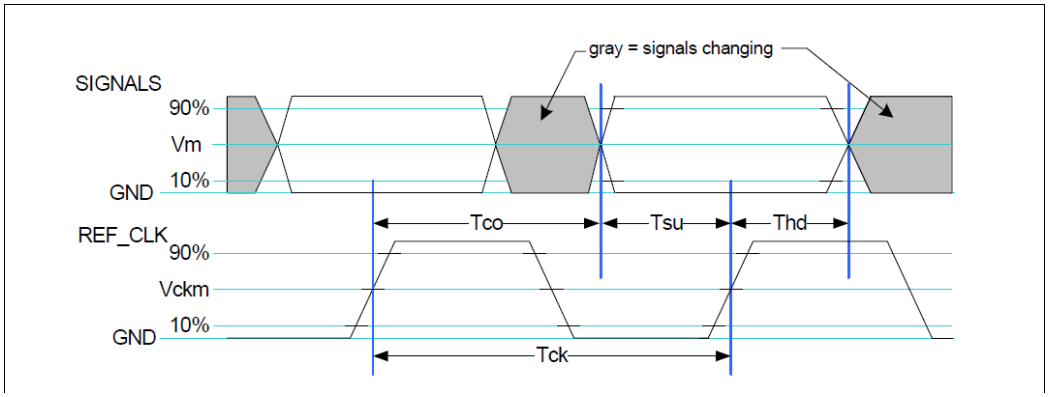


Figure 12-4 NC-SI AC Specification

12.4.6.6 Reset Signals

For power-on indication, the X540 can either use an internal power-on circuit, which monitors the 2.5V, 1.2V, and 0.8V power supply, or external reset using the LAN_PWR_GOOD pad.

12.4.7 PCIe Interface AC/DC Specification

The X540 PCIe interface supports the electrical specifications defined in:

- PCI Express* 2.0 Card Electromechanical Specification.
- PCI Express* 2.0 Base Specification, Chapter 4.

12.4.8 Network Interface AC/DC Specification

The AC/DC specification of the network interface is according to the 10GBASE-T, 1000BASE-T, 100BASE-TX and 10BASE-T Standard (802.3 802.3an 802.3u 802.3ab). The 100BASE-T parameters are also described in standard ANSI X3.263.

12.5 Thermal Diode

The X540 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 be used to monitor the die temperature of the X540 for thermal management or characterization. Table 12-4 lists the parameters that pertain to the X540's thermal diode pins (THERM_D1_P, THERM_D1_N).

Table 12-4 Thermal Diode Parameters

Symbol	Min	Typical	Max	Unit	Notes
I forward bias	10	100	1000	µA	1
n_ideality	1.02	1.05	1.09		3
ESR		3		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: X540's thermal diode has a -10 °C offset from the Tj at the center of the die.

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)	

Parameter Name	Symbol	Recommended Value	Conditions
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]

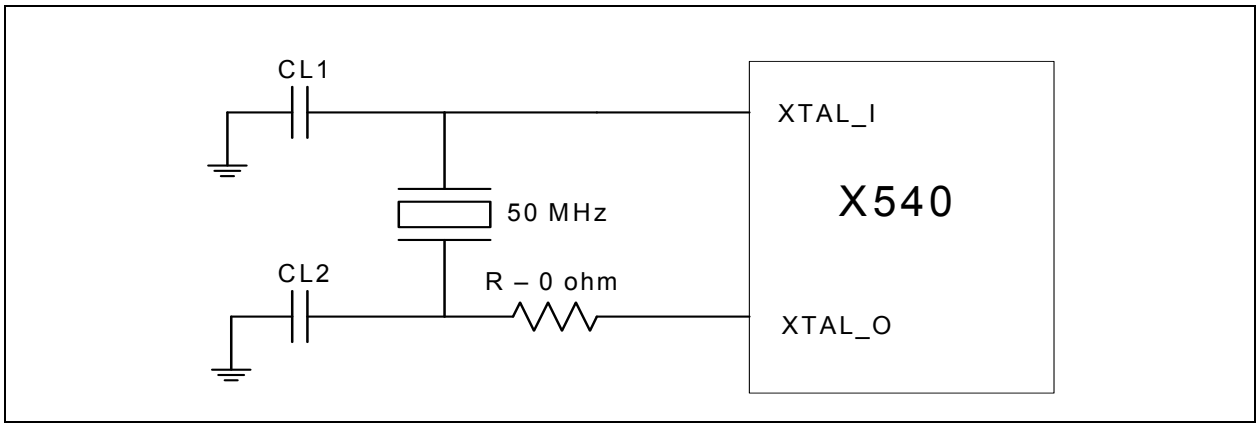
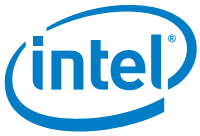


Figure 12-5 Crystal Specification Schematic



12.7 Package

12.7.1 Mechanical

The X540 is assembled into a 25 x 25 FCBGA package with a 10-layer substrate.

Body Size	Ball Count	Ball Pitch	Substrate
25 x 25 mm	576	1.0 mm	10 layers

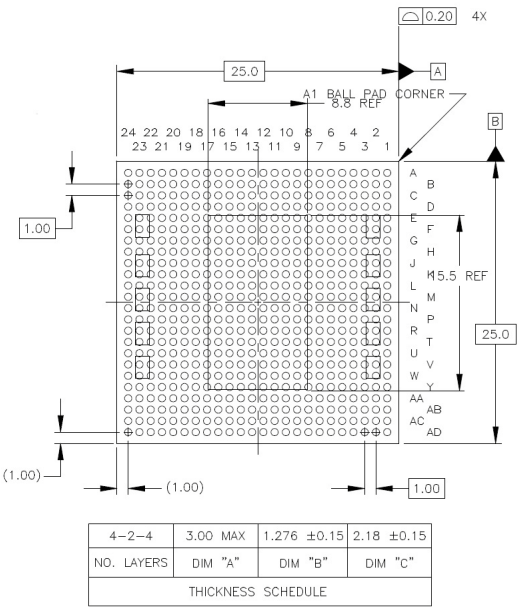
12.7.2 Thermal

For the X540's package thermals please refer to Section 14.0.

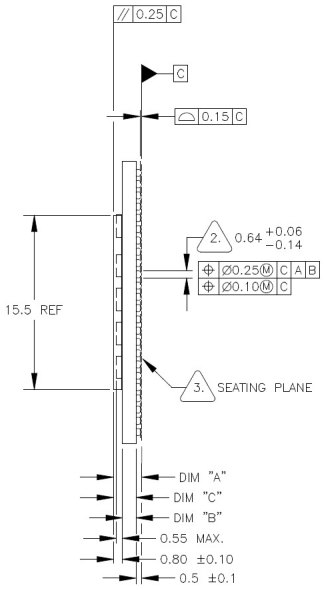
12.7.3 Electrical

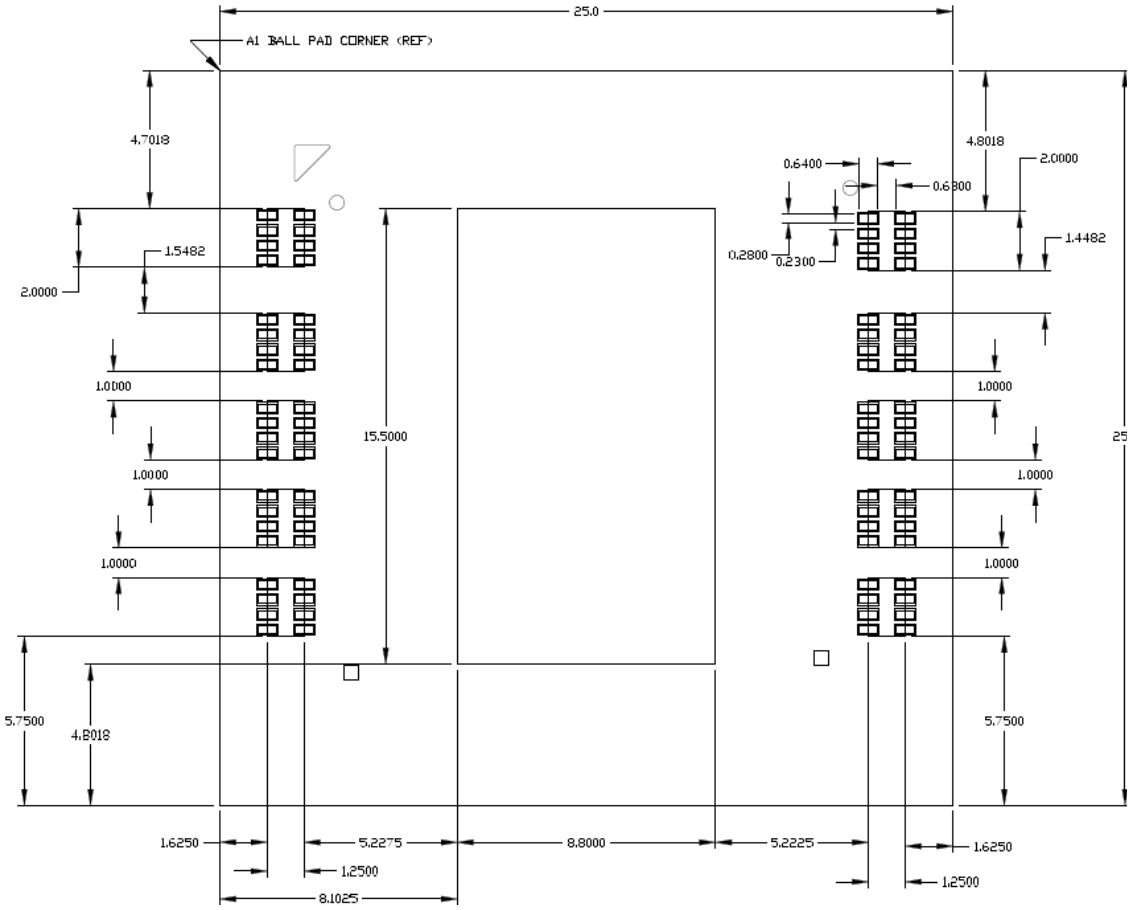
Package electrical models are part of the IBIS files.

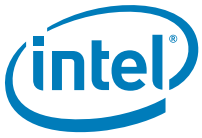
12.7.4 Mechanical Package Diagrams



4–2–4	3.00 MAX	1.276 ±0.15	2.18 ±0.15
NO. LAYERS	DIM "A"	DIM "B"	DIM "C"
THICKNESS SCHEDULE			




25.0
A1 BALL PAD CORNER (REF)
4.7018
4.8018
0.6400
0.6800
2.0000
1.4482
0.2800
0.2300
1.5482
2.0000
1.0000
1.0000
1.0000
1.0000
1.0000
1.0000
15.5000
25
5.7500
4.8018
5.7500
1.6250
5.2275
8.8000
5.2225
1.6250
1.2500
1.2500
8.1025



13.0 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 2.0 or reference schematics. There are reserved pins, identified as RSVD_2P5, RSVD_NC and RSVD_VSS. The X540 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 X540 connects to the host system using a PCIe interface. The interface can be configured to operate in several link modes as detailed in Section 3.0. 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 2.0* and *PCI Express* Card Electromechanical Specification, Revision 2.0.*

13.1.1 Link Width Configuration

The X540 supports link widths of x8, x4, x2, or x1 as determined by the *Flash Lane_Width* field in the PCIe initialization configuration. The configuration is loaded using bits 9:4 in the *Max Link Width* field of the Link Capabilities register (0xAC); default setting is 0x08. The X540 default is the x8 link width.

During link configuration, the platform and the X540 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 X540. 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 3.0.

13.1.3 PCIe Reference Clock

The X540 requires a 100 MHz differential reference clock called PE_PLL_REF_p and PE_PLL_REF_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 3.01 KΩ 1.0% resistor needs to be connected from PE_REFRES0 to the VSS supply and a 3.01 KΩ 1.0% resistor needs to be connected from PE_REFRES1 to VSS. To avoid noise coupled onto this reference signal, place the bias resistor close to the X540 and keep traces as short as possible.

13.1.5 Miscellaneous PCIe Signals

The X540 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 X540, 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 X540 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 1GBASE-T 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 RF-45 jack into a single device are also supported.

The line interface on the X540 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 with stack-jacks via strapping resistors, which sets this configuration at power-up. Refer to Section 2.1.10, specifically pin descriptions PHY0_RVSL and PHY1_RVSL.

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 X540 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 X540.

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 X540 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 12.0 for specific electrical requirements that the magnetics need to meet in the Intel® 10-GBASE-T Magnetic Specification Electrical/Mechanical Requirements for 10GBASE-T Magnetic Components.

13.2.3 5th Channel

In order to sense and cancel common-mode noise, the X540 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. In order 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 External 5th Channel Filtering

5th channel optimal performance requires an external filter to attenuate noise above 400 MHz to prevent aliasing. Because aliased signals can still be cancelled, it doesn't work if there are two simultaneous narrow band alien cross talk sources the same distance from 400 MHz, such as 300 MHz and 500 MHz. The 5th channel filter should be placed as close as reasonably possible to the X540.

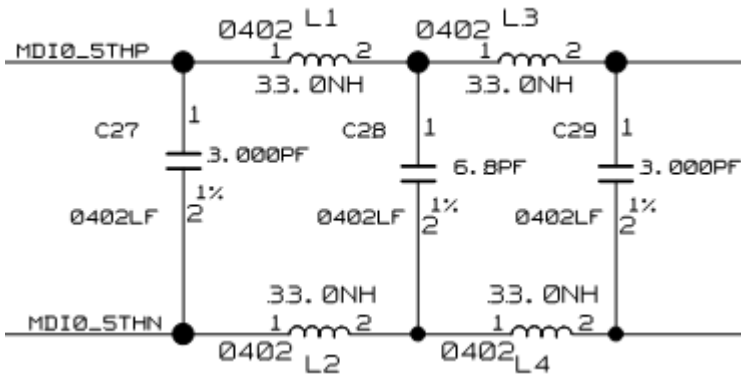


Figure 13-1. External 5th Channel Filtering Schematic

13.2.5 Board Noise Cancellation

An advantage of the 5th channel is that in addition to cancelling 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 X540. Consequently, every effort should be made to route the 5th channel traces along the same path as the other four MDI traces.

13.2.6 MDI Layout Guidance

The MDI that was chosen for the X540 designs consist of an RJ-45 right-angle PCB jack, Bob Smith termination, discrete magnetics, and a filter (see Figure 13-2). 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 13-2 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 X540 is approximately two inches behind the magnetics module for both the discrete and integrated solutions.

13.2.6.1 Discrete Magnetics

The X540 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 X540-facing common-mode choke. This provides isolation from board noise being radiated on Channel D. Following this is an anti-alias filter that is only required on the 5th channel as this filter is integrated on the four active channels. 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.

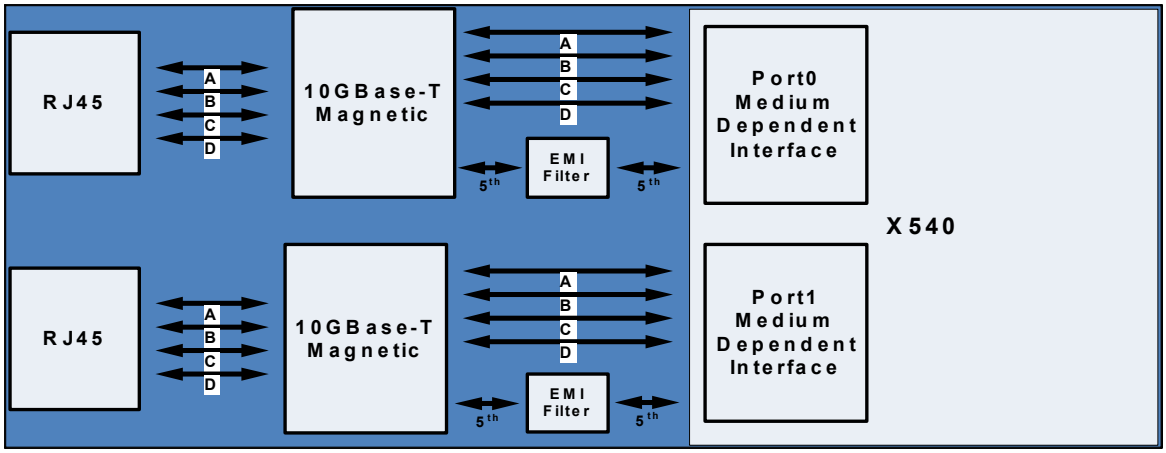


Figure 13-2. Basic MDI Placement Guidelines

The X540, referred as LAN Silicon in Figure 13-4 and Figure 13-3, must be at least one inch 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.

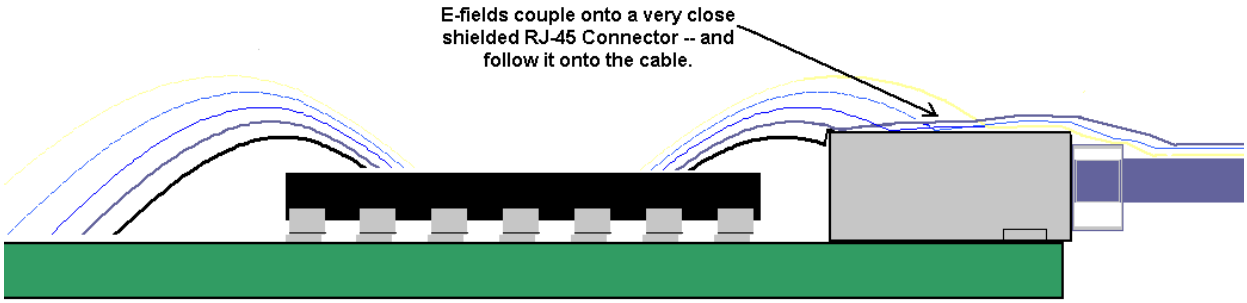


Figure 13-3. Effect of Device Placed Less Than One Inch from the RJ-45 Connector

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 13-4).

From these figures (Figure 13-4 through Figure 13-6), the following should be noted:

1. The ground is split underneath the magnetics, with the chassis ground being present under the front-end (Figure 13-4), and the circuit ground under the X540 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 X540. This enables a Bob Smith termination to be implemented from the line side transformer center taps to chassis ground.

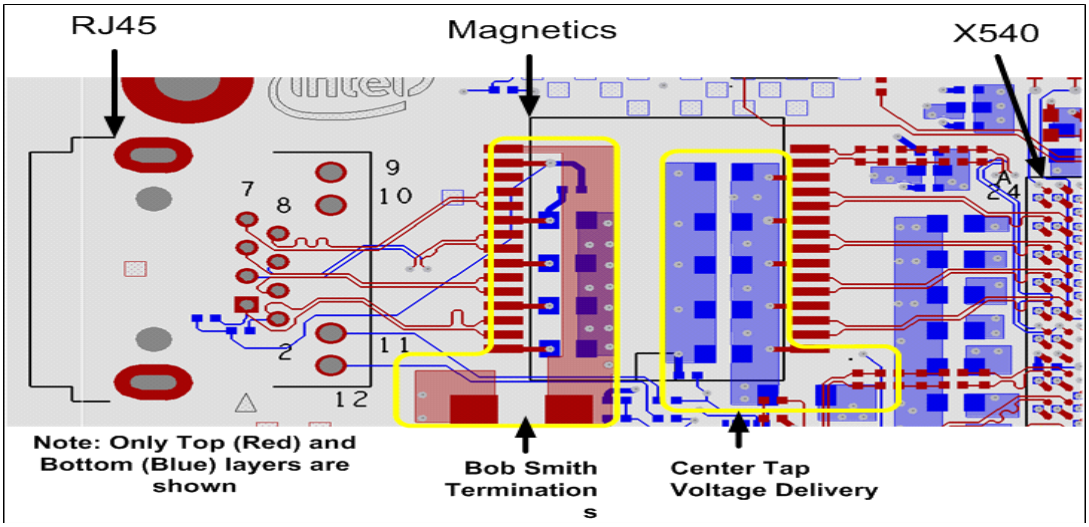


Figure 13-4. Transformer Bypass Components

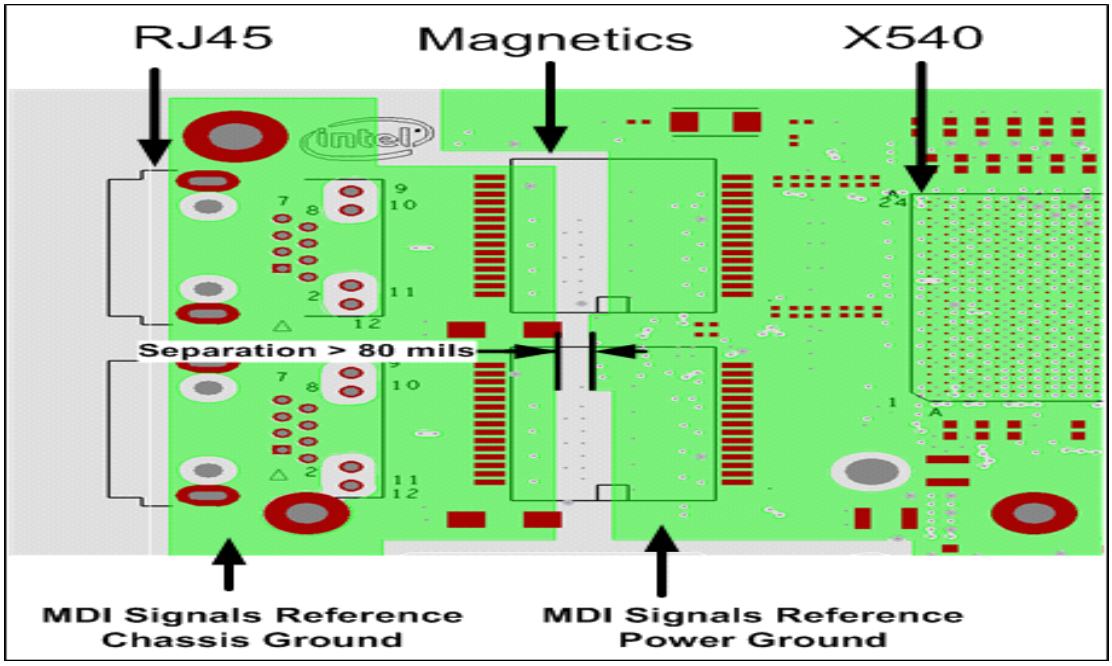


Figure 13-5. MDI Grounds

2. Bob Smith termination and the shield of the RJ-45 jack are both connected to chassis ground, which are the two large circular metallizations in the MDI (see Figure 13-5).
3. The resistors in the Bob Smith termination are required to be 0805s to handle the cable discharge voltages.
4. Between the transformer and the X540, there is an individual, differential EMI filter on the 5th channel pair that is designed to improve the performance of the X540 (see Figure 13-6). This filter should be placed as close as possible to the X540.

5. Similarly, the magnetics should be placed as close as possible to the RJ-45 jack (see Figure 13-6).

Note: As long as the filter is placed as close to the X540 as possible, the distance (assuming PCB insertion loss is accounted for in link performance) between the X540 and the filter is not critical.

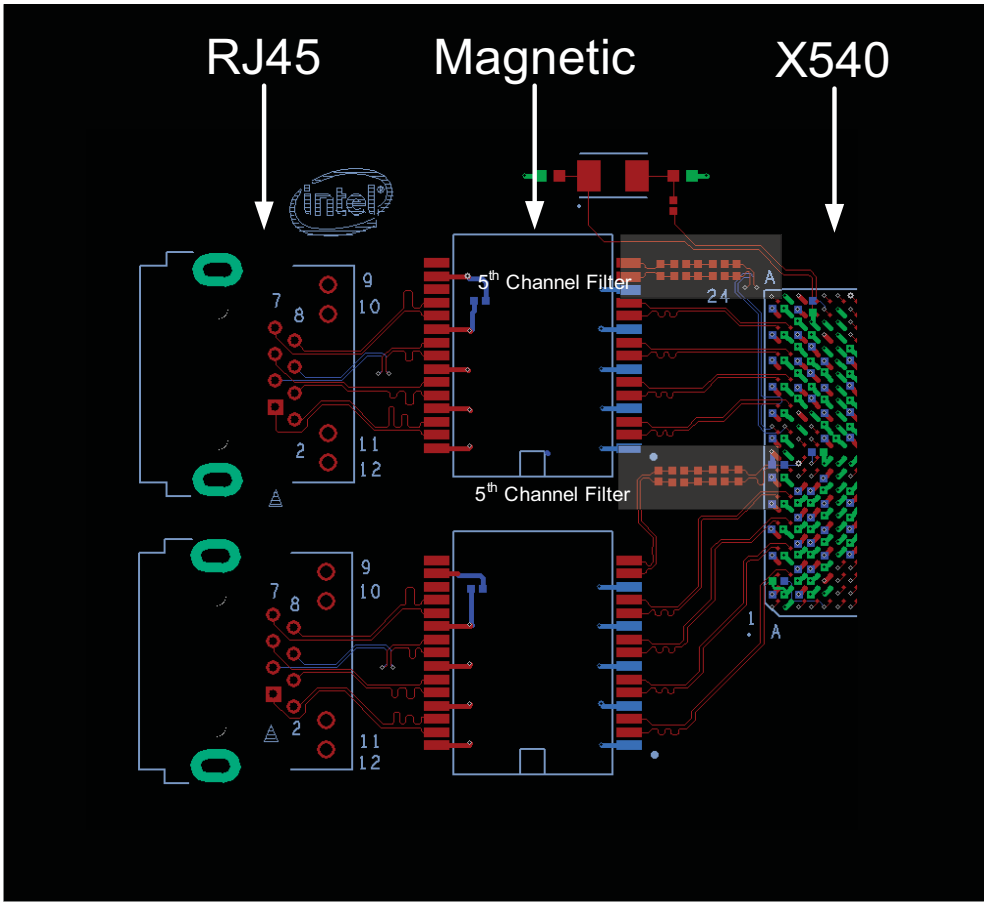


Figure 13-6. MDI Filter and Front End

6. In Figure 13-4, 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.

13.2.6.2 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 13-7). 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 X540 and the integrated module remains the same as with discrete magnetics, including length requirements, placement of coupling capacitors, and target impedance.

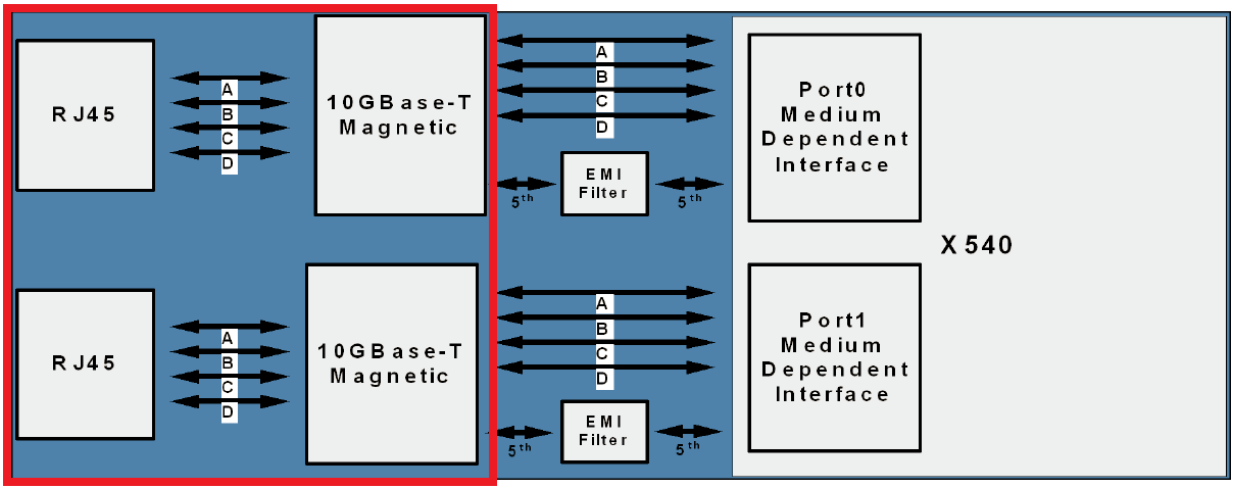


Figure 13-7. Basic MDI Placement Guidelines

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:

1. 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.
2. The required Bob Smith termination is implemented internally when using integrated magnetics.
3. Between the transformer and the X540, there is an individual, differential EMI filter on the 5th channel pair that is designed to improve the performance of the X540. This filter should be placed as close as possible to the X540.

Note: As long as the filter is placed as close as possible to the X540, the distance (assuming PCB insertion loss is accounted for in link performance) between the X540 and the filter is not critical.

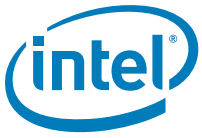
13.2.6.3 MDI Differential Pair Trace Routing for LAN Design

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.

13.2.6.4 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.

13.2.6.4.1 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 10G X540 Controller 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 80Ωdifferential 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.

13.2.6.5 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 100 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.

13.2.7 PHY MDI Lane Swap Configuration

The X540 provides flexible MDI LAN swaps for MDI board routing (see Figure 13-8).

- Default configuration - 0, 1, 2, 3 <-> A, B, C, D
  - PHY0_RVSL signals (connected to pull-down resistors)

  - PHY1_RVSL signals (connected to pull-down resistors)
- MDI swap configuration - 3, 2, 1, 0 <-> A, B, C, D
  - PHY0_RVSL signals (connected to pull-up resistors)
  - PHY1_RVSL signals (connected to pull-up resistors)
- 5th channel swap not supported (AFE_LINE_4P and AFE_LINE_4N)

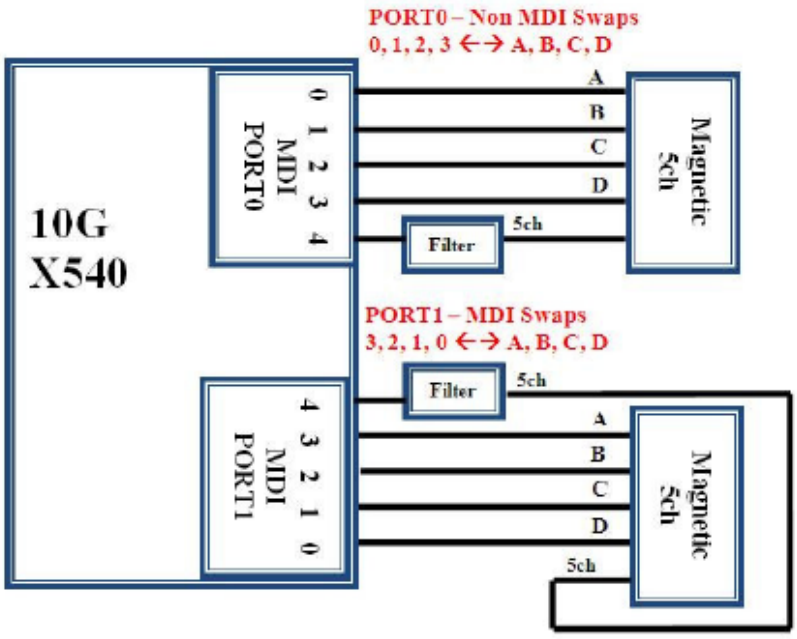


Figure 13-8. PHY AFE_LINE Lane Swap Configuration

13.2.8 Routing the Center Tap Voltage to Magnetics

In 10GBASE-T applications where the X540 is used it is required that a center tap voltage of 2.5V be delivered to the 10GBASE-T magnetics (for both discrete and integrated magnetics). There are a number of requirements necessary to comply with in order to have a reliable solution. The main ideas are that the 2.5V magnetics modules each can consume up to 600 mA (1.2 amperes total for both ports), and that only a 15 mV drop is allowed on the 2.5V rail by the time it reaches the magnetics center taps. This means that the power delivery to the center tap connections of each magnetic module needs to have enough copper such that the current and resistance drop across the entire delivery network is <15mV.

In order to achieve and meet this requirement it is necessary to ensure that the power delivery network of the 2.5V center tap voltage is wide enough to deliver the necessary current with minimal IR drop. The requirement shown in the below picture is that the power delivery network for the 2.5V center tap voltage be greater than 100 mils wide.

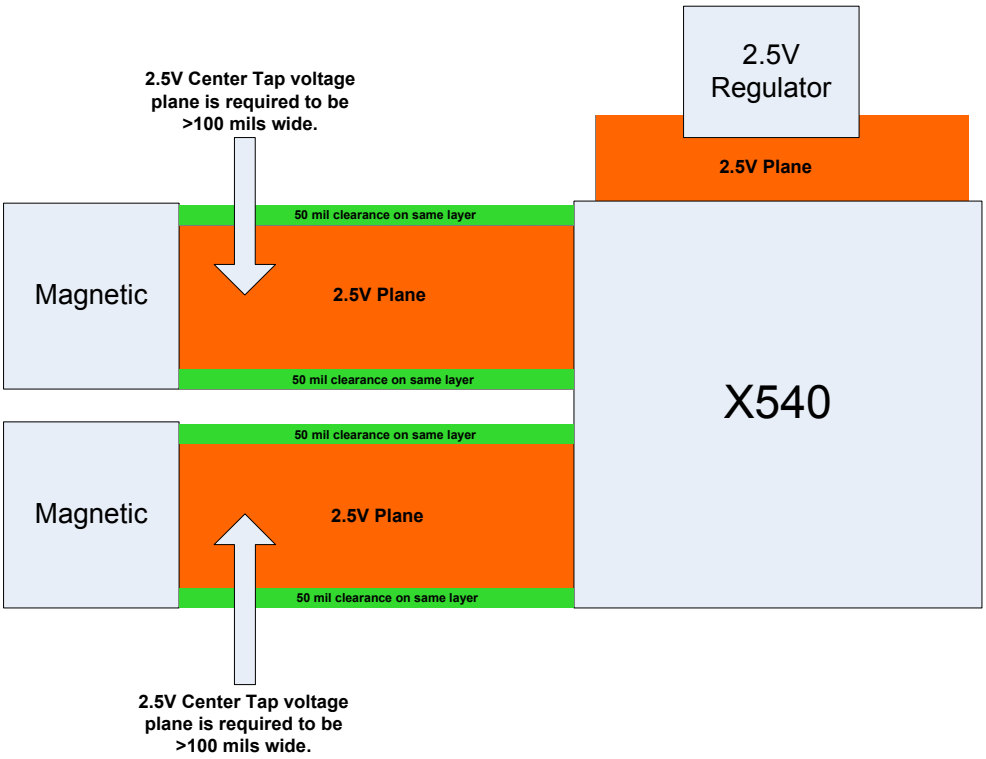


Figure 13-9. Routing Center Tap Voltage to Magnetics

In addition to requiring a width greater than 100 mils Intel also requires that there be a clearance of 50 mils on the same layer for the power delivery network. This is required to ensure a low noise environment for this center tap voltage delivery.

13.3 Connecting the Power Supply Delivery Network

For the X540, it is necessary to provide an adequate power supply delivery network such that the performance of the device meets expected performance requirements. In order 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 *X540 10GBASE-T Dual Port Ethernet Controller Reference Design* schematics.

In order to further ensure that these specific high frequency and bulk decoupling capacitors are placed in locations around and under the X540, Figure 13-10 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 X540, 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 the capacitance value and along with the inductance of copper connecting them still perform optimally.

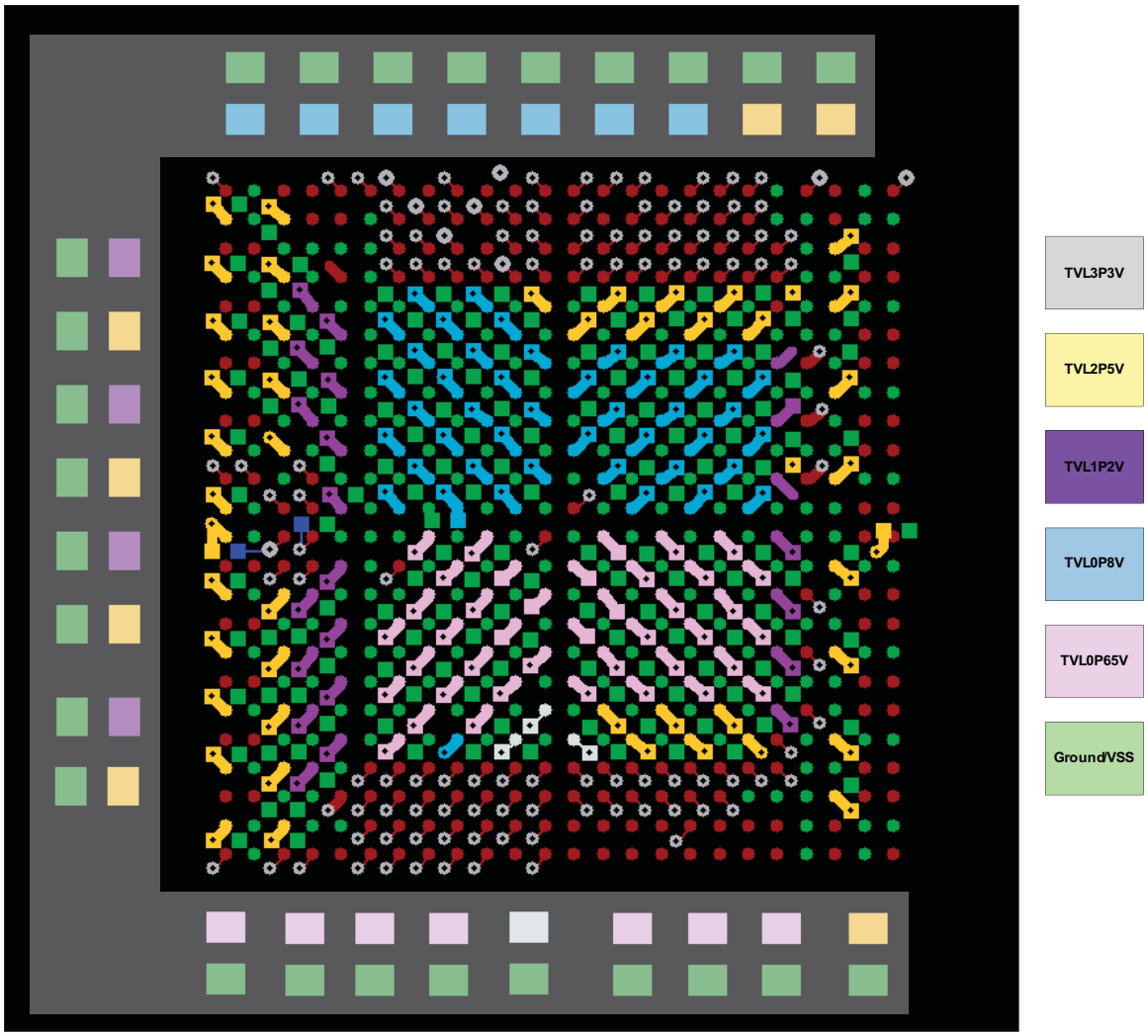


Figure 13-10. Power Supply Delivery Network

Within the BGA landing pattern it is recommended that the high speed decoupling capacitors be placed directly below the X540 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.

13.3.1 Connecting 2.5V and 1.2V Isolating Analog Power Supply Through Ferrite Bead

The 1.2V and 2.5V supplies are primarily analog supplies so they must be kept very quiet. The X540 defined two 1.2V isolating power rails that should be connected to E3 and E21 through a ferrite bead and one 2.5V power rail that should be connected to C24.

These rails should have a wide power plane and both the ferrite bead and the bulk decoupling capacitor should placed close to the X540.

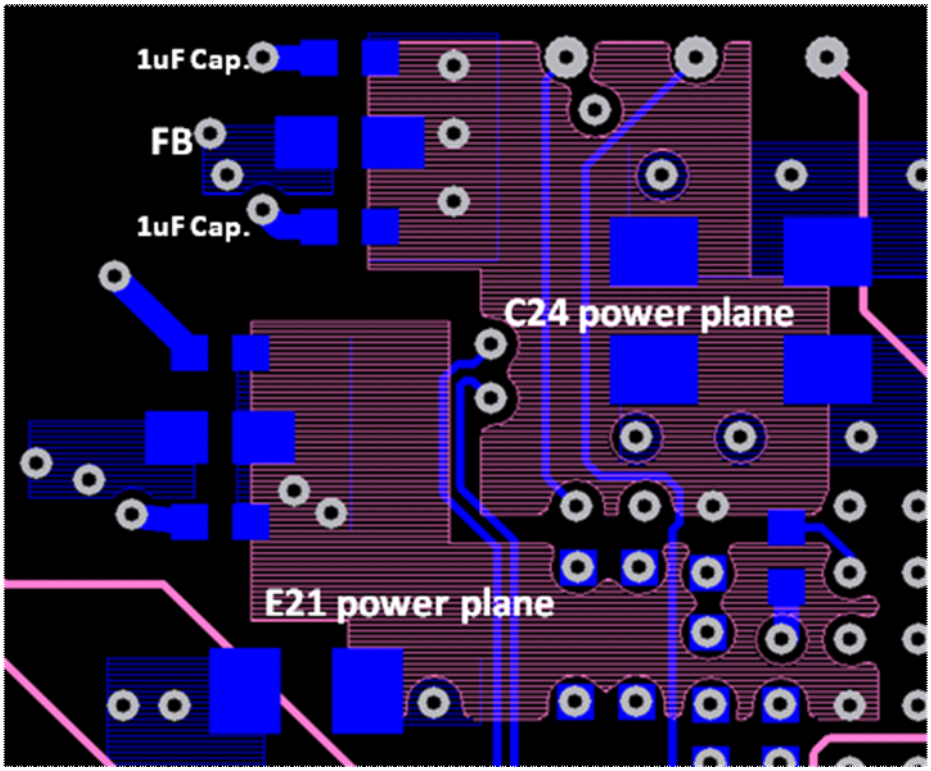


Figure 13-11. Routing the Isolating Analog Power Supply

13.4 Connecting the Flash Interface

13.4.1 Connecting the Flash

The X540 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 X540 uses a SPI Flash. Several words of the Flash are accessed automatically by the X540 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 6.0.

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 X540 contains 2.5V Flash I/O in order to provide flexible devices:

- 2.5V Flash family with single power supply of 2.5V

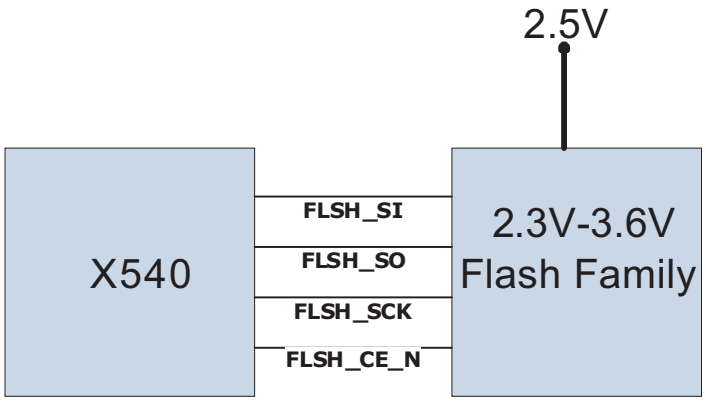


Figure 13-12. 2.5V Flash Family

- 3.3V Flash family with single power supply of 2.8V by additional low cost Linear Voltage Regulator (LDO)

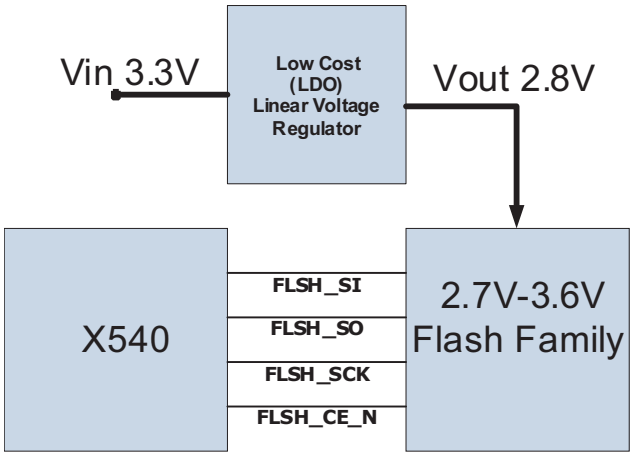


Figure 13-13. 3.3V Flash Family

Note: For supported 2.5V and 3.3V Flash types, see Section 13.13.

For more information on how to properly attach the Flash device to the X540, follow the example provided in the reference schematics. Contact your Intel representative for access.

13.5 Connecting Manageability Interfaces

SMBus and NC-SI are optional interfaces for pass-through and/or configuration traffic between the MC and the X540. For a description of the operation mode of these interfaces, refer to Section 7.0 and the Section 12.0.

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 X540 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 13-14 shows the connectivity requirements and provides information about a single-drop application. This configuration only has a single package connected to the MC. The X540 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.

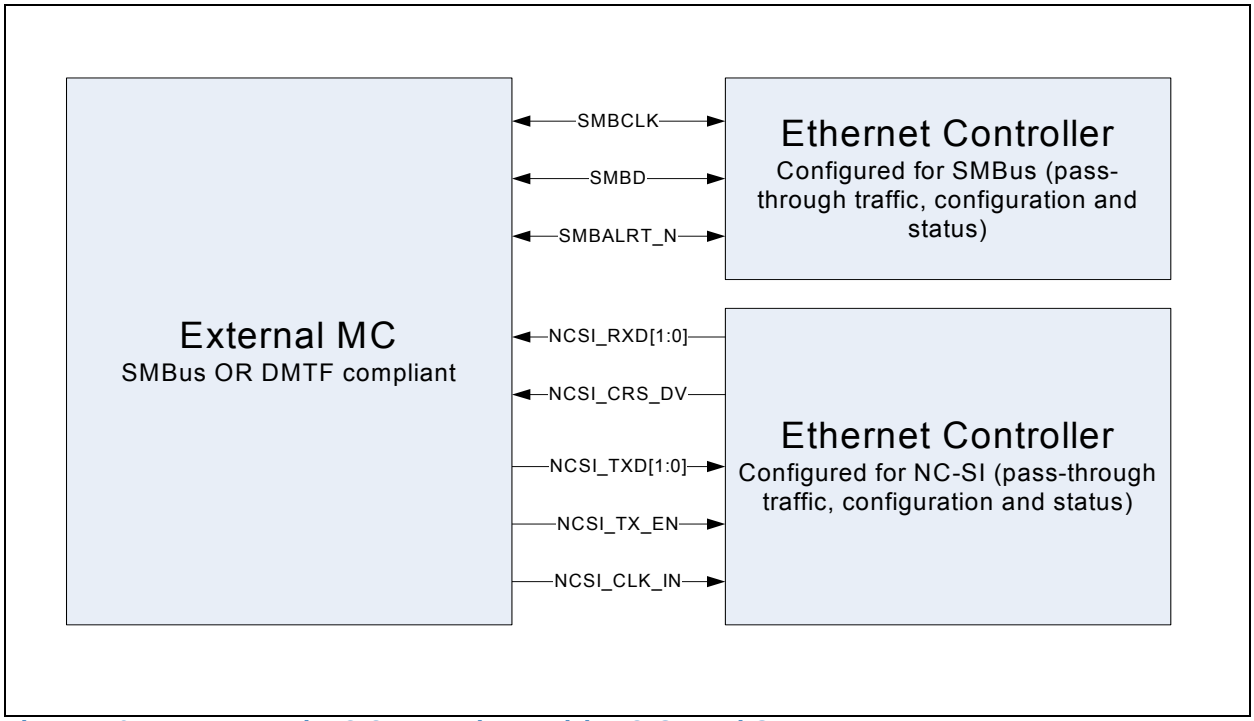


Figure 13-14. External MC Connections with NC-SI and SMBus

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 13-13 shows the connectivity requirements necessary for interfacing a NC-SI compliant MC.

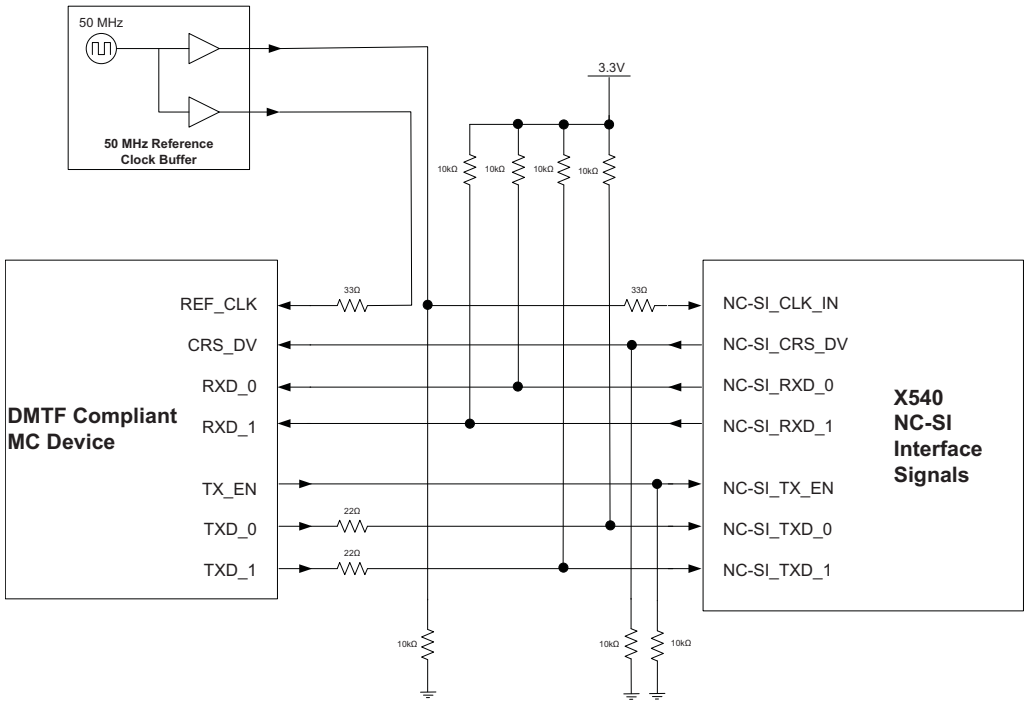


Figure 13-15. NC-SI Connections to an External MC

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.

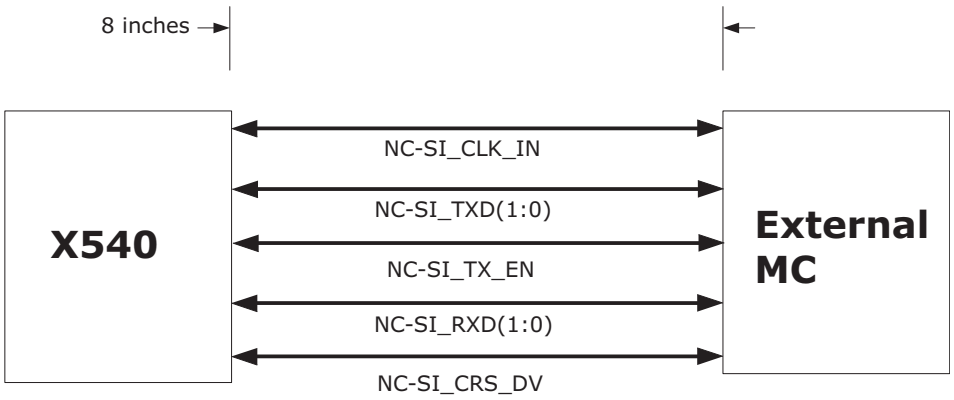


Figure 13-16. NC-SI Maximum Trace Length Requirement Between the X540 and MC for Direct Connect Applications

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 X540 within specification. See Figure 13-17

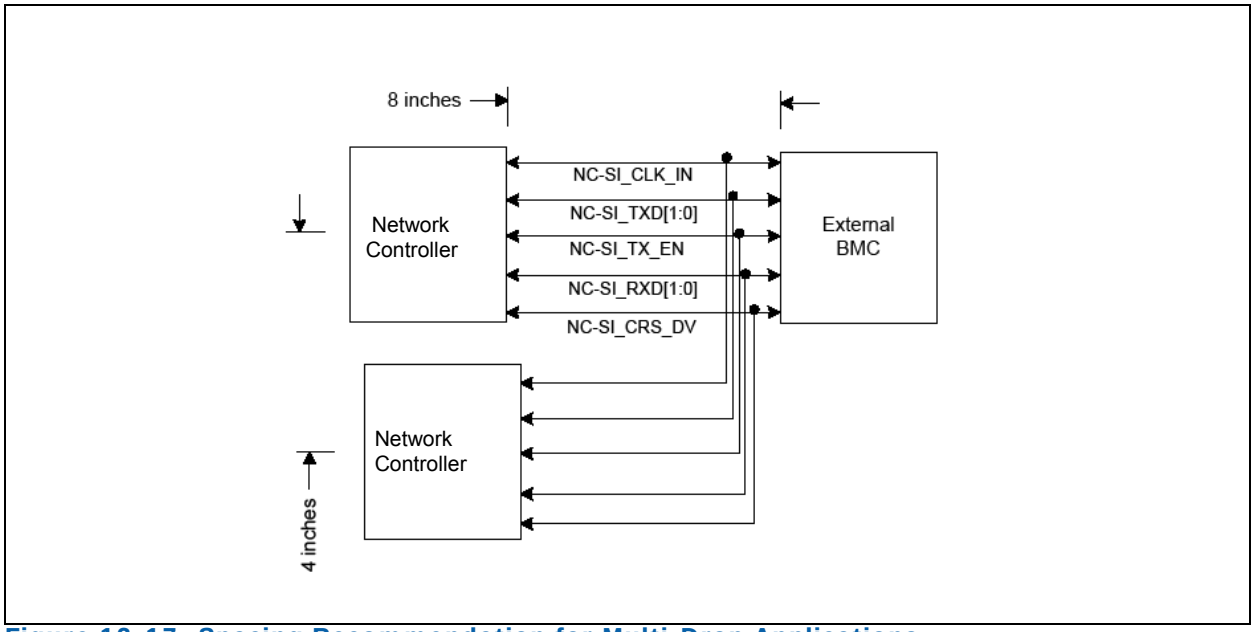


Figure 13-17. Spacing Recommendation for Multi-Drop Applications

13.5.3.3 Special Delay Requirements

To ensure the X540 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 X540 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 X540 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, 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 2.5V 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 X540 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.

Note: LED forward voltage requires ~2.0V. In some conditions the X540 cannot provide the proper diode forward voltage because of the 2.5V I/O. As a result, a 3.3V buffer is recommended (see Figure 12.13).

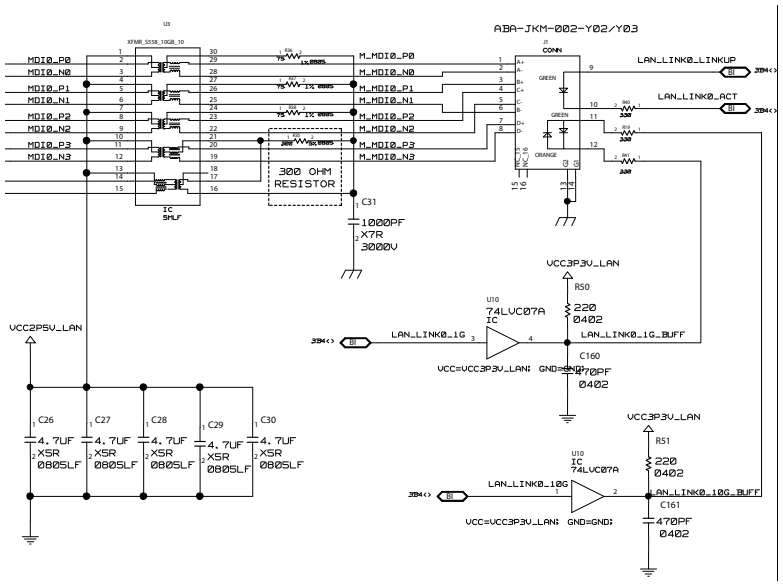


Figure 13-18. LED Connectivity

13.8 Connecting Miscellaneous Signals

13.8.1 LAN Disable

The X540 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	LAN1	Disable
LAN 1 is disabled	x	LAN 0	Disable
Both LAN functions are disabled	Both PCI functions are disabled. The X540 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	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.		

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 X540 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 X540 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 X540 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 X540.

13.9 Connecting the JTAG Port

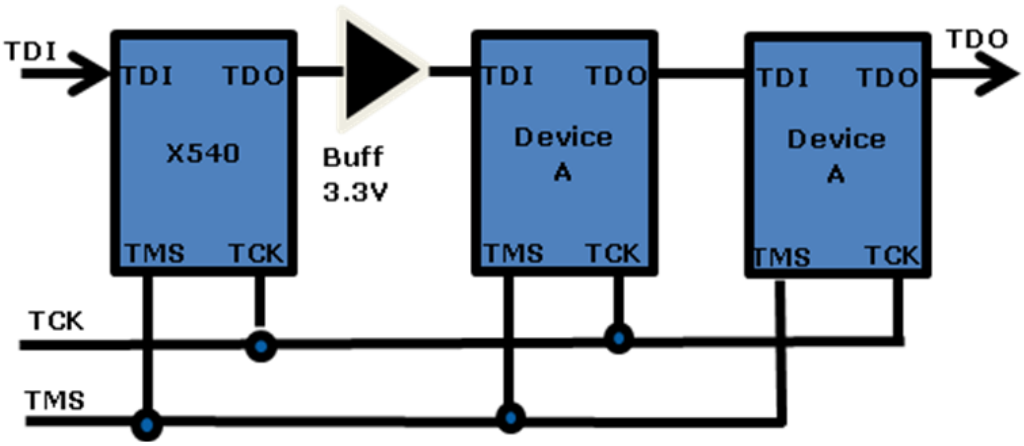
The X540 contains a test access port (3.3 V only) conforming to the IEEE 1149.1-2001 Edition (JTAG) specification. To use the test access port, connect these balls to pads accessible by specific test equipment.

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 X540 is available for use in specific test environment.

Because the X540 has a 2.5V I/O output and input buffer that is 3.3V tolerant, two design considerations can be implemented:

- Non daisy chain — Use test equipment with a 2.5V adjustment output voltage like CORELIS NetUSB-1149.1/E™
- Daisy chain — Connect the X540 TDO to TDI device through a 3.3V buffer



13.10 Power On Reset (POR)

After power is applied the X540 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 2.5V, 1.2V, 0.8V, 0.67V 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 up).
   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 X540.

All designs require an external clock. The only option for this clock source is a 50 MHz external crystal. The X540 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 13-19 illustrates a simplified schematic of the internal oscillator circuit. Pin X1 and X2 refers to XTAL1 and XTAL2 in the X540, 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.

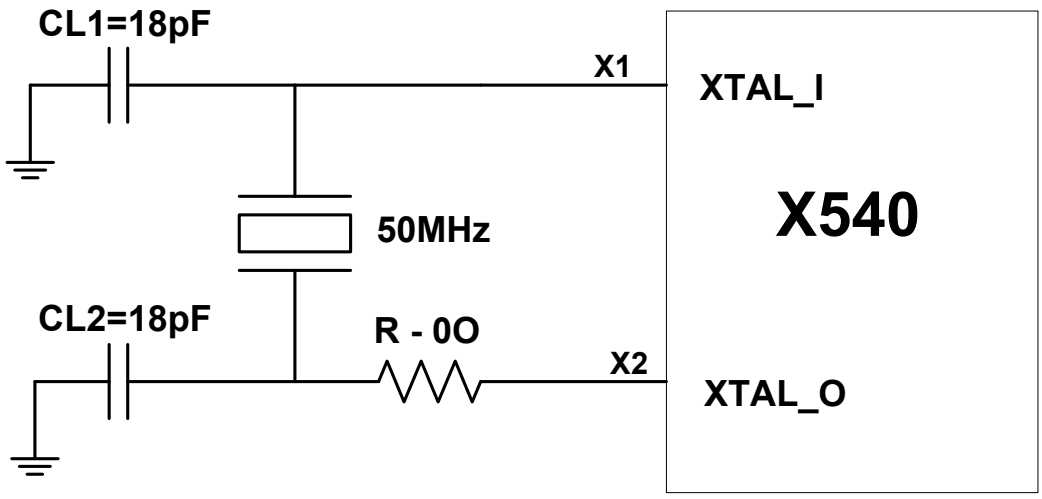


Figure 13-19. Oscillator Circuit

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 Cstray = 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.

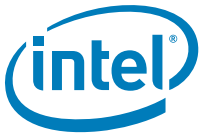
13.11.12 Reference Crystal

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 will 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 will be 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, then 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 will be 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, then 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) will also vary. If the thickness tolerance for the outer layers of dielectric are controlled within ±17 percent of nominal, then 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 will increase 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 X540 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.

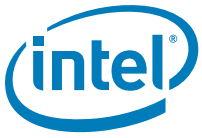
Note: Layers 4 and 5 should be used mostly for low-speed signals because they are referenced to potentially noisy power planes that might also be slotted.

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	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				
							.005 @ 49 OHMS				
				106-74%							
				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				
							.005 @ 52 OHMS				
				106-74%							
				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 13-20 shows correct via usage. Figure 13-21 shows the type of topology that should be avoided.

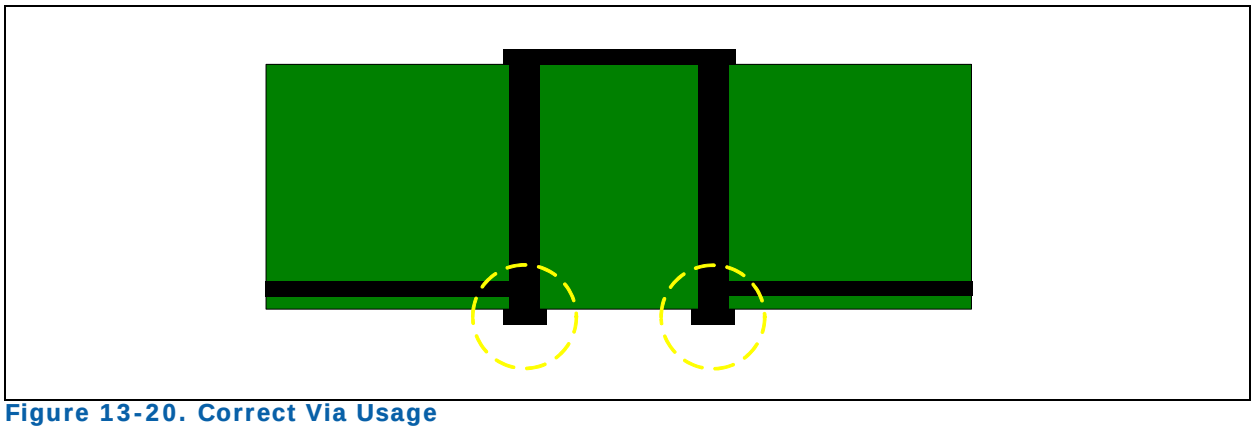


Figure 13-20. Correct Via Usage

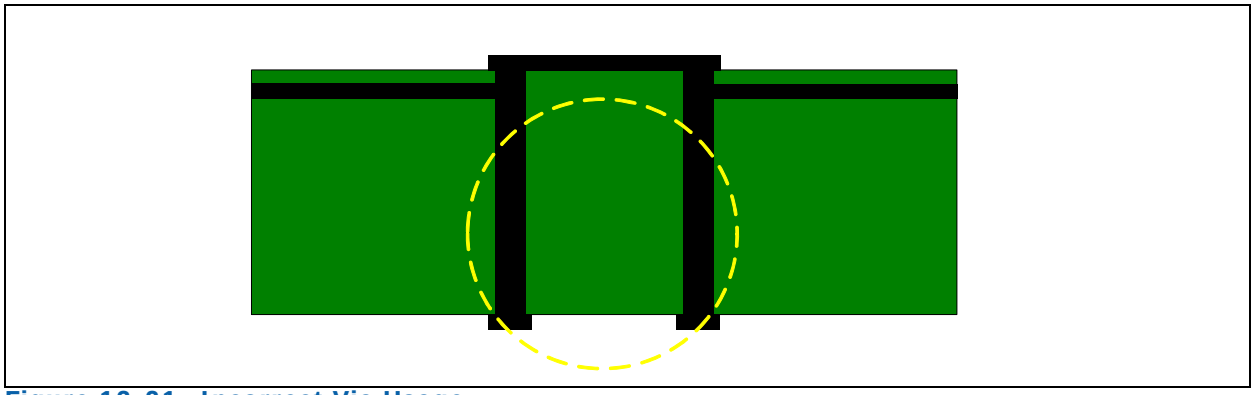


Figure 13-21. Incorrect Via Usage

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.

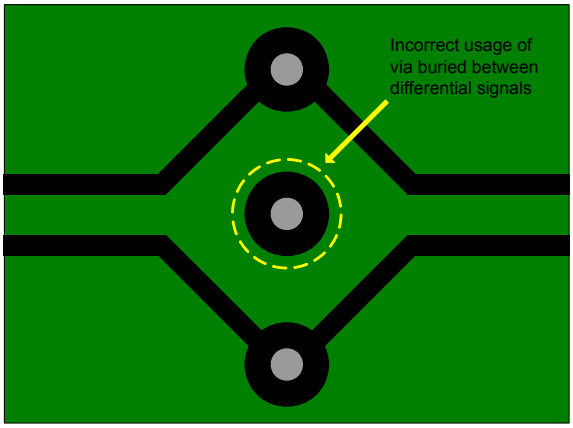


Figure 13-22. No Vias Between High-Speed Traces in the Same Differential Pair

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. Refer to Figure 13-23.

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 in order 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.

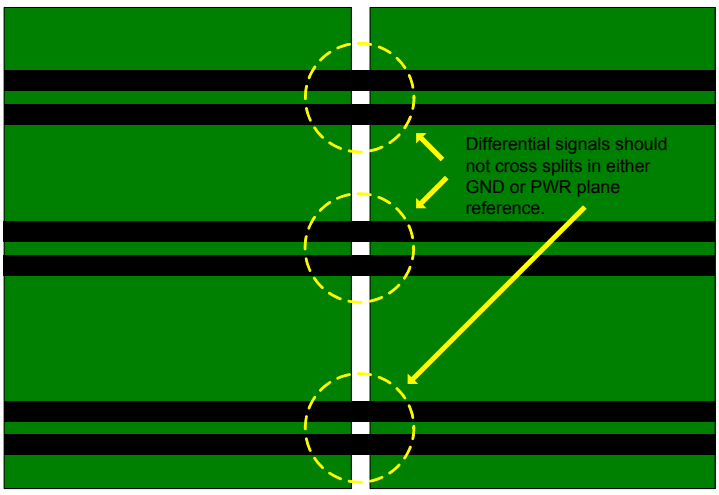


Figure 13-23. Do Not Cross Plane Splits

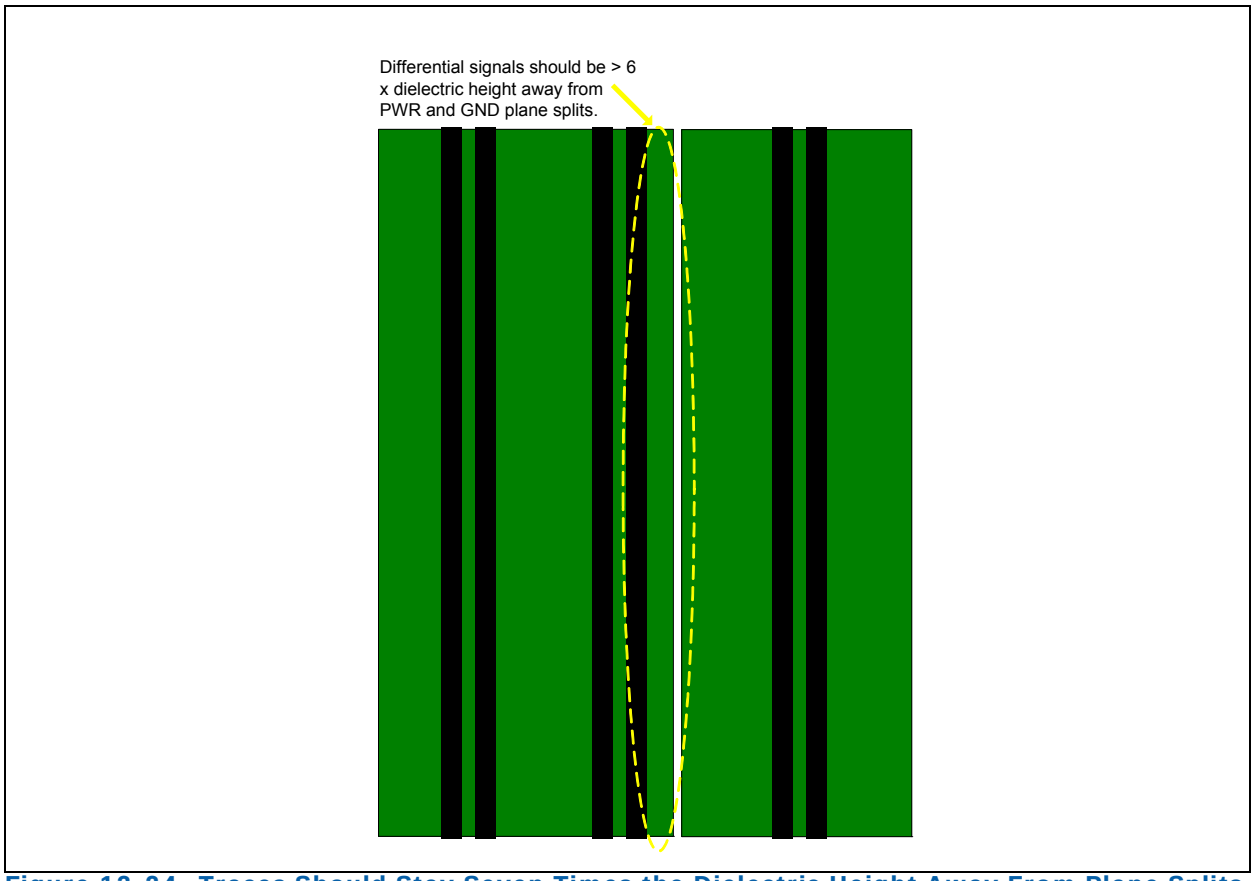


Figure 13-24. Traces Should Stay Seven Times the Dielectric Height Away From Plane Splits Or Voids

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, then 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. Refer to Section 12.0 for the PHY in regards to actual placement requirements of the capacitors.

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, then 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, then 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 13-25 and Figure 13-26.

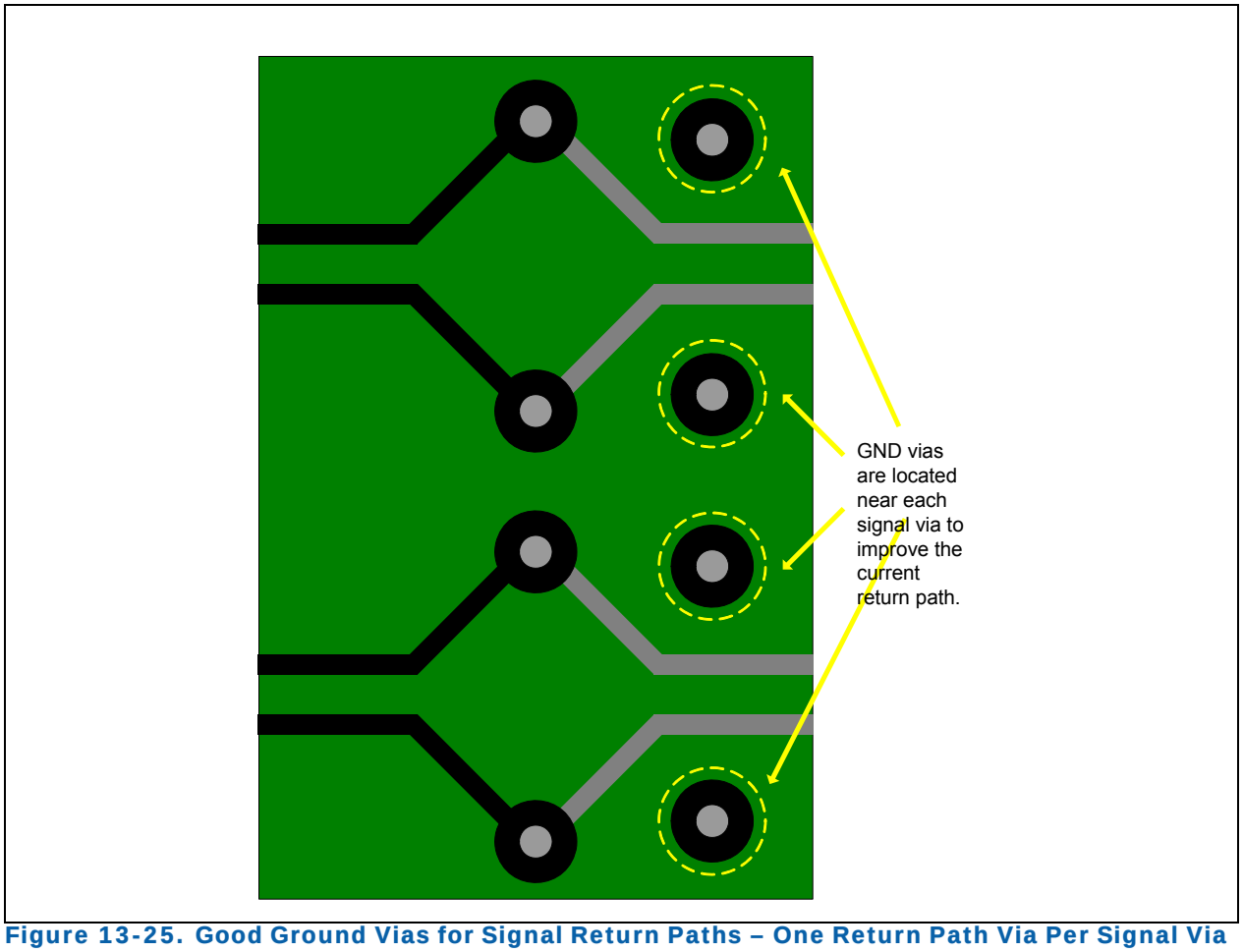


Figure 13-25. Good Ground Vias for Signal Return Paths – One Return Path Via Per Signal Via

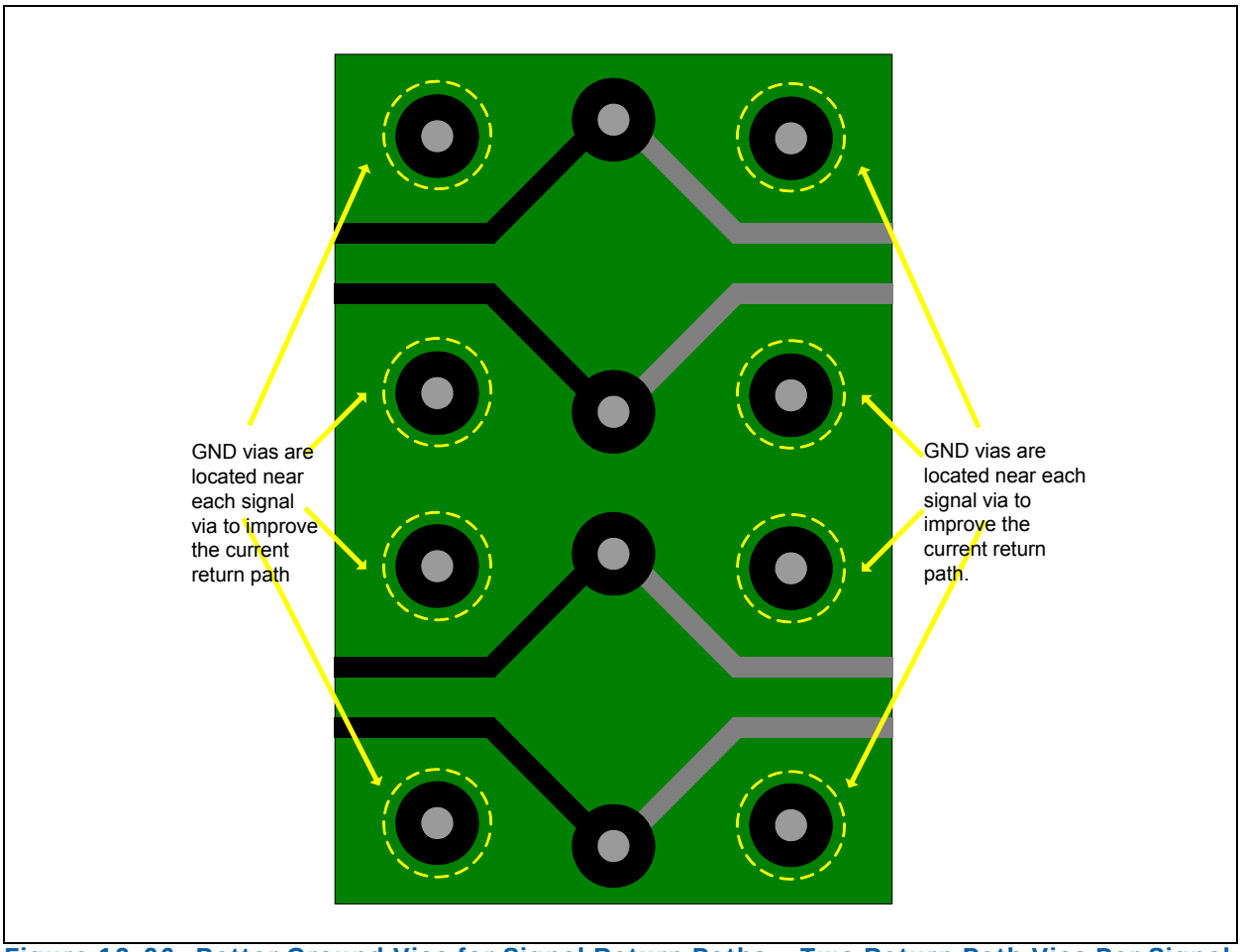


Figure 13-26. Better Ground Vias for Signal Return Paths – Two Return Path Vias Per Signal Via (Less Reflection)

If the PCB fabrication process permits it, it's best to remove signal via pads on unconnected metal layers. See Figure 13-27 and Figure 13-28.

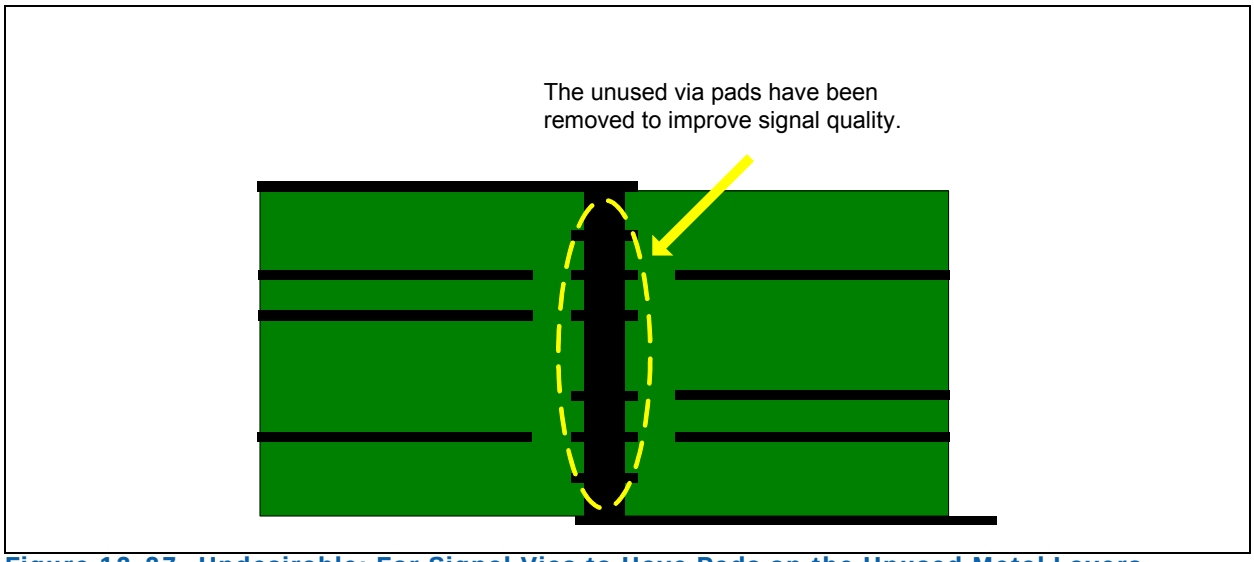


Figure 13-27. Undesirable: For Signal Vias to Have Pads on the Unused Metal Layers

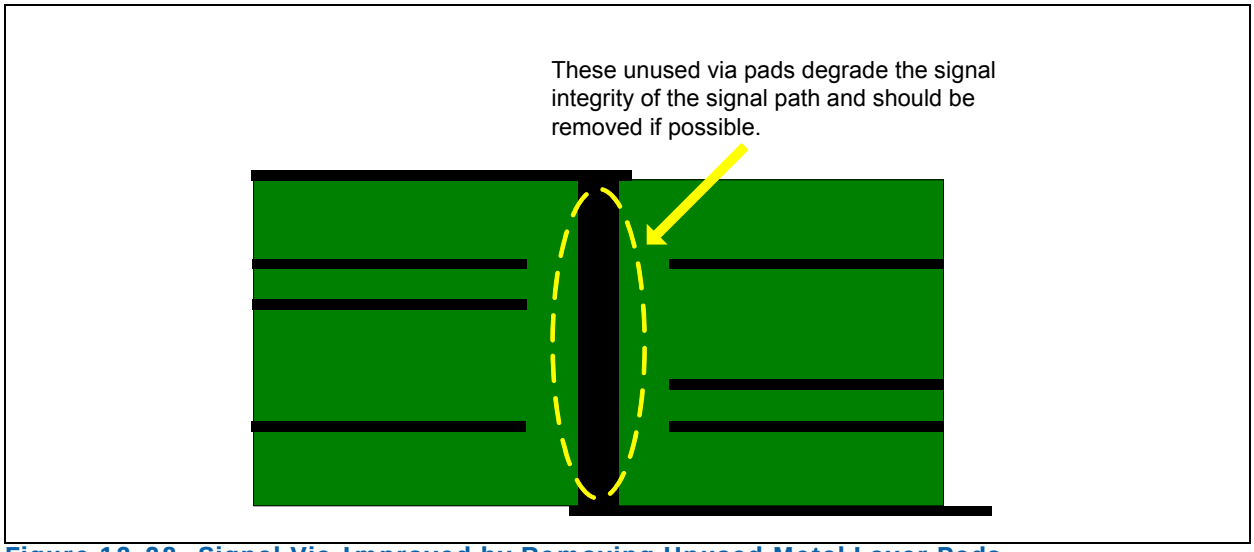


Figure 13-28. Signal Via Improved by Removing Unused Metal Layer Pads

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 13-29. Clearance between the pad and the surrounding metal should be >= 10 mils.

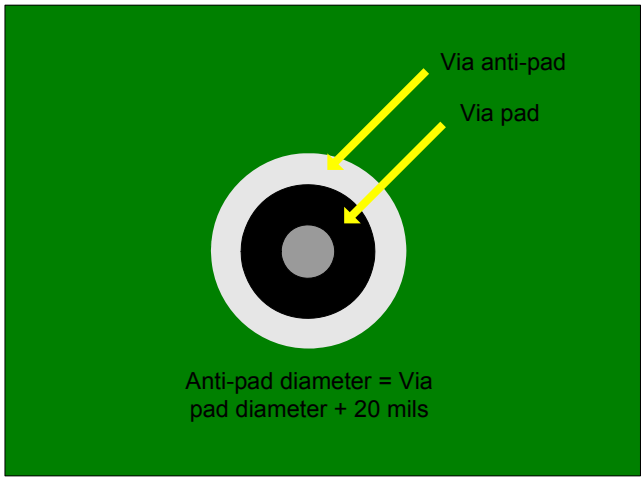


Figure 13-29. Increase Anti-Pad Diameter To Reduce Shunt Capacitance

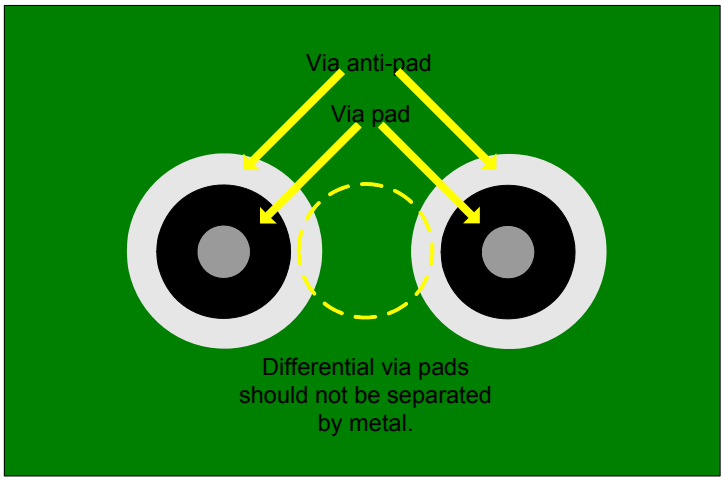


Figure 13-30. Differential Signal Via Pads Should Not Be Separated By Metal

Each time differential signal vias pass through a plane layer, within each differential pair, the anti-pads should overlap. See Figure 13-31, Figure 13-32 and Figure 13-32.

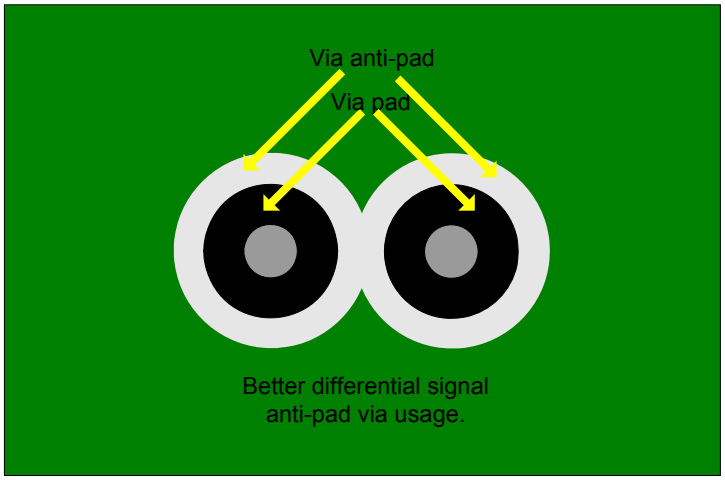


Figure 13-31. Better Differential Signals Via Anti-pads

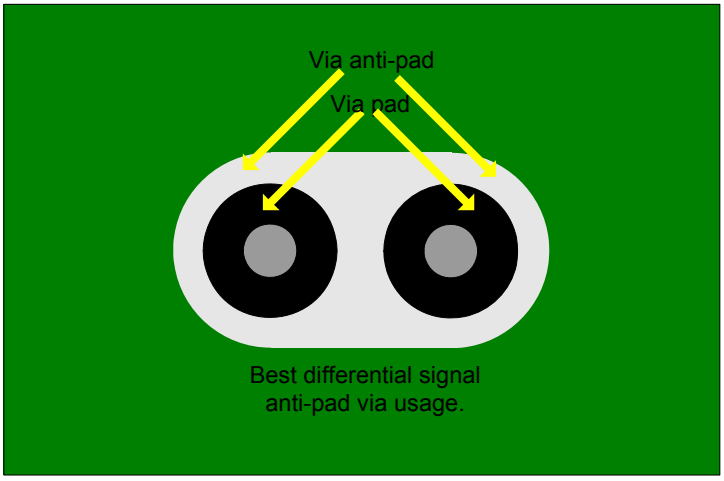


Figure 13-32. Best Differential Signals Via Anti-Pads

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:

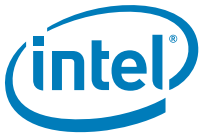
- Planned 10GBASE-T signal trace routing on the PCB complies with the interconnect characteristics recommended in IEEE 802.3an.
- With sufficient advance notice, Intel engineers can provide assistance under these conditions:
  - Trace routing should be optimized prior to the next steps – request a layout 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, then 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)

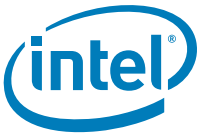
Table 13-5 lists the BOM materials for all X540 LAN on Motherboard (LOM) designs.

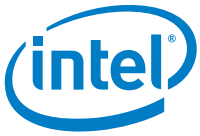
Table 13-5. X540 LOMs

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-038 G17-188T-038 (Light-pipe) G12-1JJT-038 JT4-1108HL 1840497-1 RJTGE1G4172J JFM5801J-710G-4F	2
RJ45	Lotes Bel-Fuse	ABA-JKM-002-Y02/ABA-JKM-002-Y03 SS60300-010 (No LED)	
2.5V Flash	Winbond Numonyx Atmel	W25Q16CL M25PX16 AT25DF161A-SHF-T (F-2.5V)	1
3.3V Flash	Atmel Winbond Macronix	AT25DF161A-SH-T W25Q16CV MX25L1605A	1
5th CH Filter – Inductor 33 nH	Coilcraft Morata	0402 - HP-33NXGLU 0402 - L0075S0080LQW15A_00	8
5th CH Filter – Cap. 3 pF	Johanson Dielectrics, Inc.	0402 - 500R07S3R0BV4T	4
5th CH Filter – Cap. 6.8 pF	Murata	0402 - GRM1555C1H6R8CZ01D	2
Power Supplies FB	TDK Corporation	MPZ2012S101A	3



Note: This page intentionally left blank.





14.0 Thermal Design Recommendations

14.1 Introduction

This section can be used as an aid to designing a thermal solution for systems implementing the the X540.

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 X540 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. Heatsinks and higher airflow might be required if case temperatures exceed those listed in Table 14-2.

Information in this section includes:

- Packaging Terminology — provides definitions for terminology used in this section.
- Thermal Specifications — provides the X540 case temperature specifications and where to find power requirements. This sub-section also discusses thermal packaging techniques.
- Thermal Attributes — provides the X540 thermal characteristic data, package mechanical attributes, and package thermal characteristic data. Use this sub-section to determine your thermal solution requirements.
- Thermal Enhancements — discusses the use of heatsinks, heatsink attach methods, interfacing, and reliability.
- Measurements for Thermal Specifications — provides instructions for measuring the X540 case temperature with and without a heatsink.
- Thermal Diode — provides information how to monitor the X540 die temperature for thermal management or characterization.
- Conclusion — provides a summary.
- Heatsink and Attach Suppliers — provides manufacturer contact information.
- PCB Guidelines — includes general PCB recommendations.

14.2 Intended Audience

The intended audience for this section is system design engineers using the X540. 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 X540 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 X540.

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:

- Flip Chip Ball Grid Array (FCBGA): 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 X540).
- 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).
- θJA (Theta JA): Thermal resistance junction-to-ambient, °C/W.
- ΨJT (Psi JT): Junction-to-top (of package) thermal characterization parameter, °C/W. ΨJT 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. ΨJT 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

To ensure proper operation of the X540, 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 θJA values should be used as reference only and can vary by system environment. ΨJT values also can vary by system environment, and are listed in Table 14-1 as the maximum value for the X540 simulations.

Analysis indicates that real applications are unlikely to cause the X540 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 X540 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	?$_{JA}$ (°C/W)	?$_{JT}$ (°C/W)
25 mm FCBGA without IHS[1]	16.5[2]	NA
25 mm FCBGA without IHS –HS (11.43 mm height)[3]	8.91[4]	0.23[5]

1. Integrated Heat Spreader. The X540 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 with high profile 11.43mm.
4. Natural Convection (Still Air), Heat sink attached.
5. Psi_JT is given as maximum value for a worst-case X540 scenario and might vary to a lesser value in some scenarios.

Table 14-2 The X540 Absolute Thermal Maximum Rating (°C)

Application	Estimated TDP (W)[1]	Tcase Max hs[2] (°C)[3]
X540	12.5 W @ 110 Tj max	107

1. Power value listed in Table 14-2 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 X540 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

14.8.1 Designing for Thermal Performance

Sub-sections Heatsink and Attach Suppliers and PCB Guidelines describes the PCB and system design recommendations required to achieve the proper X540 thermal performance.

14.8.2 Typical System Definition

A system with the following attributes was used to generate thermal characteristics data:

- A heatsink 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 X540 thermal performance.

14.8.3 Package Mechanical Attributes

The X540 is packaged in a 25 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 heatsink combination for the X540 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 X540 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* or detailed). Contact your local Intel sales representative for the X540 thermal models.

Table 14-3 X540 Expected Tcase and Tj (°C) for 11.43 mm High Heat Sink at 12.5 W (JEDEC Card)

	Air Flow [LFM]								
Ambient	0	50	100	150	200	250	300	350	400
45	144.5	131.3	114.5	102.7	92.65	86.97	82.96	80.5	77.84
50	149.8	136.3	119.8	107.7	97.63	91.98	87.97	85.06	82.85
55	154.4	141.2	124.5	112.2	102.7	96.94	92.98	90.06	87.88
60	159.7	146.2	129.4	117.2	107.7	102	97.98	95.07	92.89
65	164.6	151.1	134.4	122.2	112.7	107	103	100.1	97.9
70	169.5	156.2	139.8	127.7	117.7	112	108	105.1	102.9
75	174.5	161.2	144.4	132.3	122.7	117	113	110.1	107.9
80	179.9	166.1	149.4	137.3	127.7	122	118	115.1	112.9
85	184.9	171.1	154.4	142.3	132.7	127	123	120.1	117.9

Note: The red blocked value(s) indicate airflow/ambient combinations that exceed the allowable junction temperature for the X540 at 12.5 W.

14.9 Thermal Enhancements

One method frequently used to improve thermal performance is to increase the device surface area by attaching a metallic heatsink to the component top. Increasing the surface area of the heatsink reduces the thermal resistance from the heatsink to the air, which increases heat transfer.

14.9.1 Clearances

To be effective, a heatsink 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 heatsink used with the X540 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.

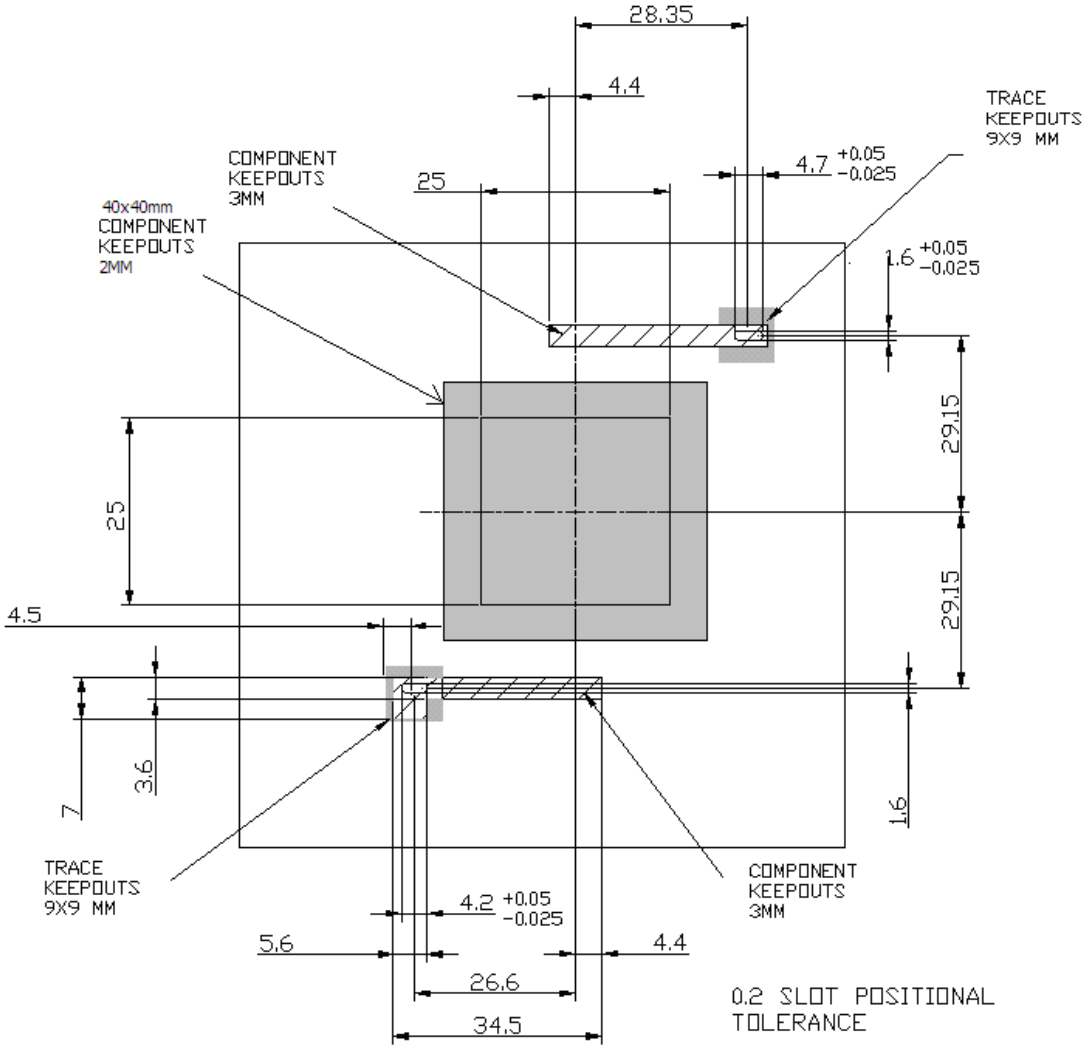

28.35
4.4
TRACE
KEEPOUTS
9X9 MM
COMPONENT
KEEPOUTS
3MM
25
4.7 +0.05 -0.025
40x40mm
COMPONENT
KEEPOUTS
2MM
1.6 +0.05 -0.025
29.15
25
29.15
4.5
3.6
7
1.6
TRACE
KEEPOUTS
9X9 MM
COMPONENT
KEEPOUTS
3MM
4.2 +0.05 -0.025
5.6
4.4
26.6
34.5
0.2 SLOT POSITIONAL
TOLERANCE

Figure 14-1 X540 Heatsink Keep Out Restrictions (Top Side Keep Out)

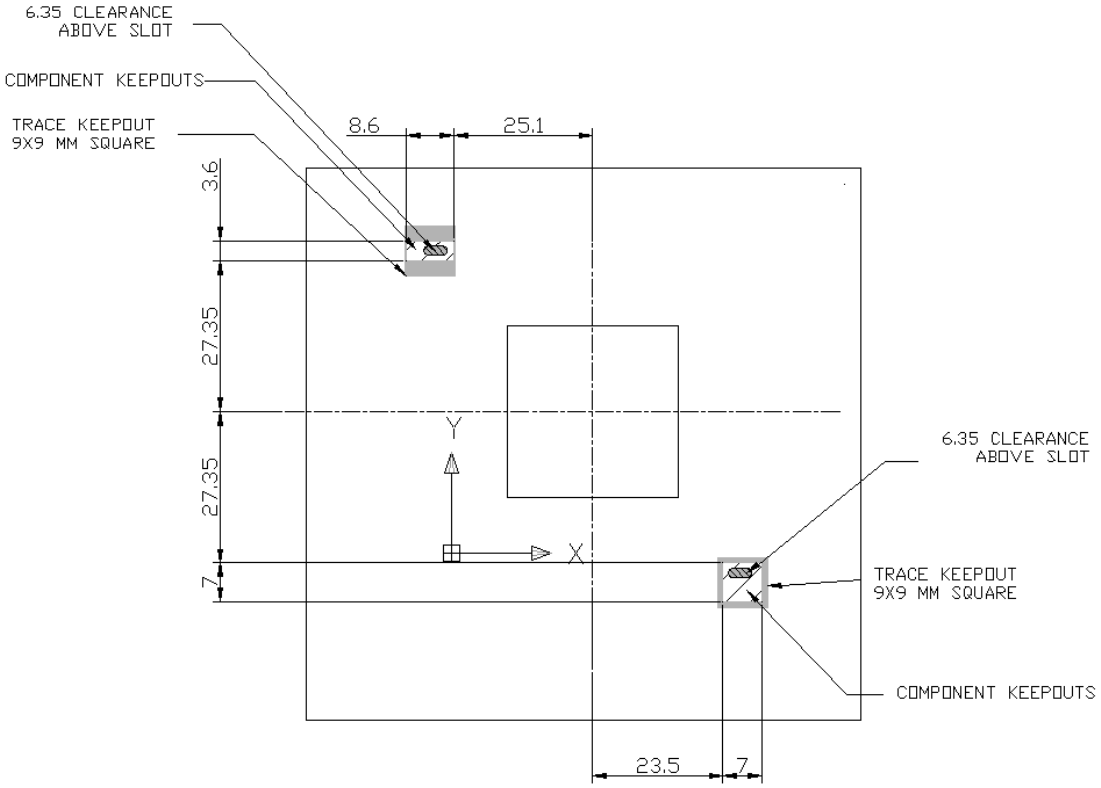


Figure 14-2 X540 Heatsink Keep Out Restrictions (Bottom Side Keep Out)

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 Table 14-3 at the TDP. If, after implementing the recommended enhanced thermal solution, the case temperature continues to exceed allowable values, then 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 Heatsinks

If required, the following extruded heatsinks are the suggested X540 thermal solutions (Figure 14-3). Also, see Figure 14-1and Figure 14-2 for heat sink information.

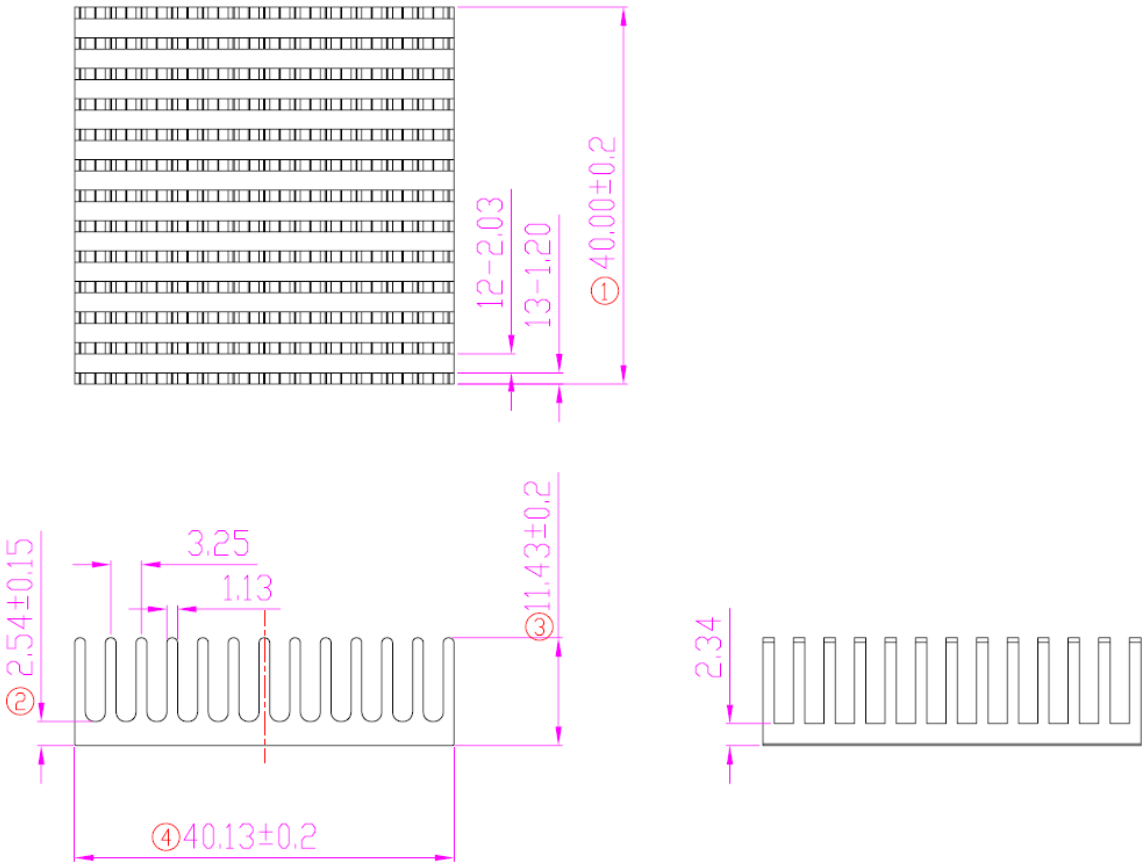


Figure 14-3 X540 Extruded Heatsinks (11.43 mm Height Passive Heat Sink (AAVID Thermalloy Profile# 72845)

14.9.4 Attaching the Extruded Heatsink

The extruded heatsink can be attached using clips 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 heatsink 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.

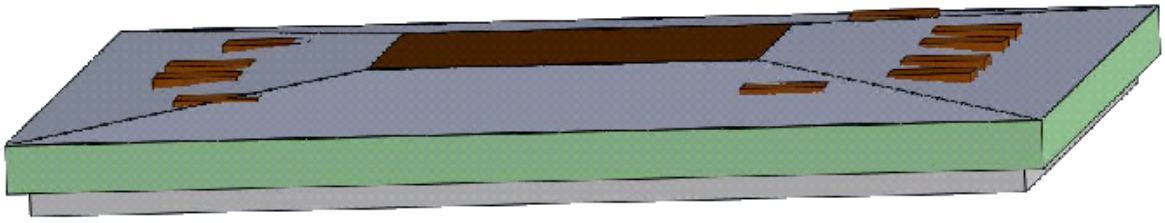


Figure 14-4 Die-Side Capacitors Exposed to Heat Sink Contact

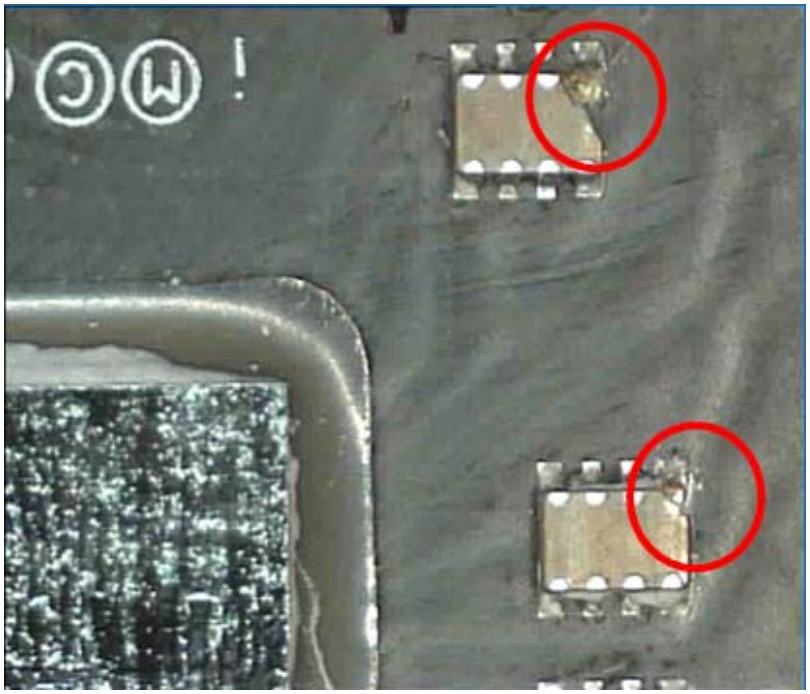


Figure 14-5 Damage Caused by Heat Sink Contact

14.9.4.4 Maximum Static Normal Load

The X540 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 heatsink 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 heatsink-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 heatsink 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 X540 heatsink design, it is important to understand the interface between the silicon die and the heatsink 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 heatsink 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 heatsink 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 heatsink 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 heatsink. 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 X540 case temperature are provided in Section 14.10.1.

14.10.1 Case Temperature Measurements

Maintain the X540 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 Heatsink)

The following approach is recommended to minimize measurement errors for attaching the thermocouple with no heatsink:

- 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.

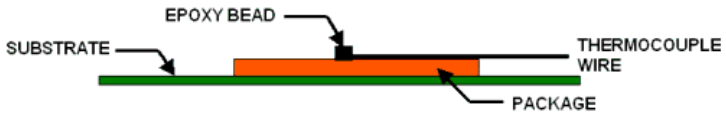


Figure 14-6 Technique for Measuring Tcase with 0° Angle Attachment and No Heatsink

14.10.1.2 Attaching the Thermocouple (Heatsink)

The following approach is recommended to minimize measurement errors for attaching the thermocouple with heatsink:

- 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 heatsinks.
- For testing purposes, a hole (no larger than 0.150 inches in diameter) must be drilled vertically through the center of the heatsink to route the thermocouple wires out.
- Ensure there is no contact between the thermocouple cement and heatsink base. Any contact affects the thermocouple reading.

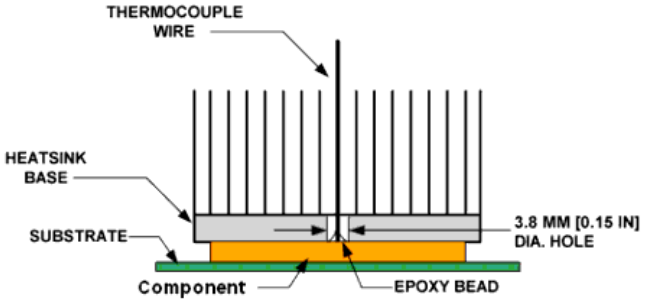


Figure 14-7 Technique for Measuring Tcase with 90° Angle Attachment

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 heatsinks, 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 heatsink 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 X540. Properly designed solutions provide adequate cooling to maintain the X540 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, heatsinks might be required if case temperatures exceed those listed in Table 14-2.

By maintaining the X540 case temperature at or below those recommended in this section, the X540 will function properly and reliably.

Use this section to understand the X540 thermal characteristics and compare them to your system environment. Measure the X540 case temperatures to determine the best thermal solution for your design.

14.12 Heatsink and Attach Suppliers

Table 14-5 Heatsink and Attach Suppliers

Part	Part Number	Supplier	Contact
Extruded Al Heat sink + Clip + PCM45 (TIM) Assembly	Generated specific to customer numbering scheme	Cooler Master	Eugene Lai Cooler Master USA, INC. (Fremont) Office: 510-770-8566 # 222 yuchin_lai@coolermaster.com
PCM45 Series	PCM45F	Honeywell	North America Technical Contact: Paula Knoll 1349 Moffett Park Dr. Sunnyvale, CA 94089 Cell: 1-858-705-1274 Business: 858-279-2956 paula.knoll@honeywell.com

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 X540 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.

15.0 Diagnostics

15.1 JTAG Test Mode Description

The X540 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 k ohm 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 k ohm 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 k ohm 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

Table 15-2. Main TAP Instructions Supported

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
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 X540 A0 is 0x11512013 (Intel's Vendor ID = 0x013, Device ID = 0x1512, Rev ID = 0x1) The ID code value for the X540 B0 is 0x21512013 (Intel's Vendor ID = 0x013, Device ID = 0x1512, Rev ID = 0x2)	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 shall select 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

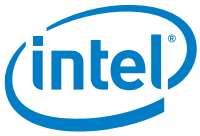
Loopback operations are supported by the X540 to assist with system and the X540 debug. Loopback operation can be used to test transmit and receive aspects of software drivers, as well as to verify electrical integrity of the connections between the X540 and the system (such as PCIe bus connections, etc.) See Section 3.6.4 for more details.

15.2.1 Tx->Rx MAC Loopback

This loopback is closed on the internal XGMII interface of the MAC core.

To configure the X540 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 in order 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 X540 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.

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

16.0 Packets Format

16.1 ARP Packet Formats

16.1.1 ARP Request Packet

Offset	# Of bytes	Field	Value (In Hex)	Action
0	6	Destination Address		Compare
6	6	Source Address		Stored
12	4	Possible VLAN Tag		Stored
12	8	Possible Length + LLC/SNAP Header		Stored
12	2	Type	0806	Compare
14	2	Hardware Type	0001	Compare
16	2	Protocol Type	0800	Compare
18	1	Hardware Size	06	Compare
19	1	Protocol Address Length	04	Compare
20	2	Operation	0001	Compare
22	6	Sender HW Address	-	Stored
28	4	Sender IP Address	-	Stored
32	6	Target HW Address	-	Ignore
38	4	Target IP Address	ARP IP address	Compare

16.1.2 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	4	Possible VLAN Tag	From ARP Request
12	8	Possible Length + LLC/SNAP Header	From ARP Request
12	2	Type	0x0806
14	2	Hardware Type	0x0001
16	2	Protocol Type	0x0800
18	1	Hardware Size	0x06
19	1	Protocol Address Length	0x04
20	2	Operation	0x0002
22	6	Sender Hardware Address	Programmed from NVM or BMC
28	4	Sender IP Address	Programmed from NVM or BMC
32	6	Target Hardware Address	ARP Request Sender Hardware Address
38	4	Target IP Address	ARP Request Sender IP Address

16.1.3 Gratuitous ARP Packet

Offset	# of bytes	Field	Value
0	6	Destination Address	Broadcast address.
6	6	Source Address	
12	2	Type	0x0806
14	2	Hardware Type	0x0001
16	2	Protocol Type	0x0800
18	1	Hardware Size	0x06

Offset	# of bytes	Field	Value
19	1	Protocol Address Length	0x04
20	2	Operation	0x0001
22	6	Sender Hardware Address	
28	4	Sender IP Address	
32	6	Target Hardware Address	
38	4	Target IP Address	

16.2 IP and TCP/UDP Headers for TSO

This section outlines the format and content for the IP, TCP and UDP headers. The X540 requires baseline information from the software device driver in order to construct the appropriate header information during the segmentation process.

Header fields that are modified by the X540 are highlighted in the figures that follow.

Note: The IP header is first shown in the traditional (like 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 ise fetched from memory in little endian format.

1 0 1 2 3 4 5 6 7	8 9 0 1 2 3 4 5	2 6 7 8 9 0 1 2 3	3 4 5 6 7 8 9 0 1
Version / IP Hdr Length	TYPE of service	Total length (**IP header + payload** length)	
Identification		Flags	Fragment Offset
Time to Live	Layer 4 Protocol ID	Header Checksum	
Source Address			
Destination Address			
Options			

Figure 16-1 IPv4 Header (Traditional Representation - most left byte first on the wire)

Byte3	Byte2	Byte1	Byte0
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Total length LSB - MSB		TYPE of service	Version / IP Hdr Length
Fragment Offset Low	RES / NF / MF / Fragment Offset High	Identification LSB - MSB	

Header Checksum LSB - MSB	Layer 4 Protocol ID	Time to Live
Source Address LSB MSB		
Destination Address LSB MSB		
Options		

Figure 16-2 IPv4 Header (Little Endian Order - Byte 0 first on the wire)

Identification is increased on each packet.

Flags Field Definitions:

The *Flags* field is as follows. Note that hardware does not evaluate or change these bits.

- MF - More Fragments
- NF - No Fragments
- Reserved

The X540 does TCP segmentation, not IP Fragmentation. IP fragmentation might occur in transit through a network's infrastructure.

1 0 1 2 3 4 5 6 7	8 9 0 1 2 3 4 5	2 6 7 8 9 0 1 2 3	3 4 5 6 7 8 9 0 1
Version	Priority	Flow Label	
Payload Length (**excluding the IP header length**)		Next Header Type	Hop Limit
MSB Source Address LSB			
MSB Destination Address LSB			
Extensions (if any)			

Figure 16-3 IPv6 Header (Traditional Representation - most left byte first on the wire)

Byte3 7 6 5 4 3 2 1 0	Byte2 7 6 5 4 3 2 1 0	Byte1 7 6 5 4 3 2 1 0	Byte0 7 6 5 4 3 2 1 0
Flow Label LSB MSB		Version	Priority
Hop Limit	Next Header Type	Payload Length (**excluding the IP header length**) LSB - MSB	
Source Address LSB MSB			

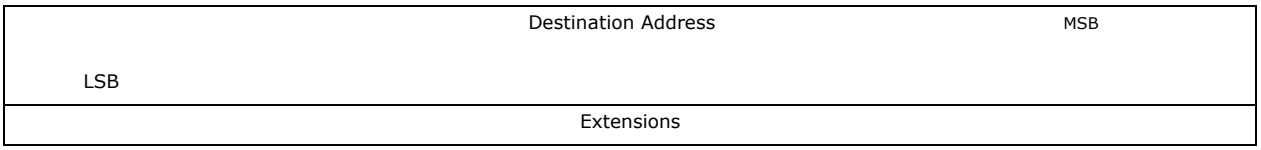


Figure 16-4 IPv6 Header (Little Endian Order - byte 0 first on the wire)

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 (such as 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.

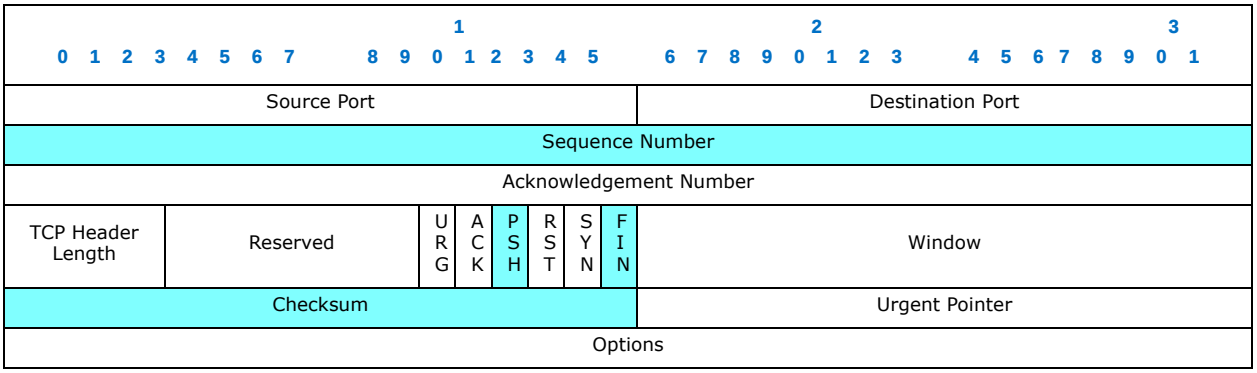


Figure 16-5 TCP Header (Traditional Representation)

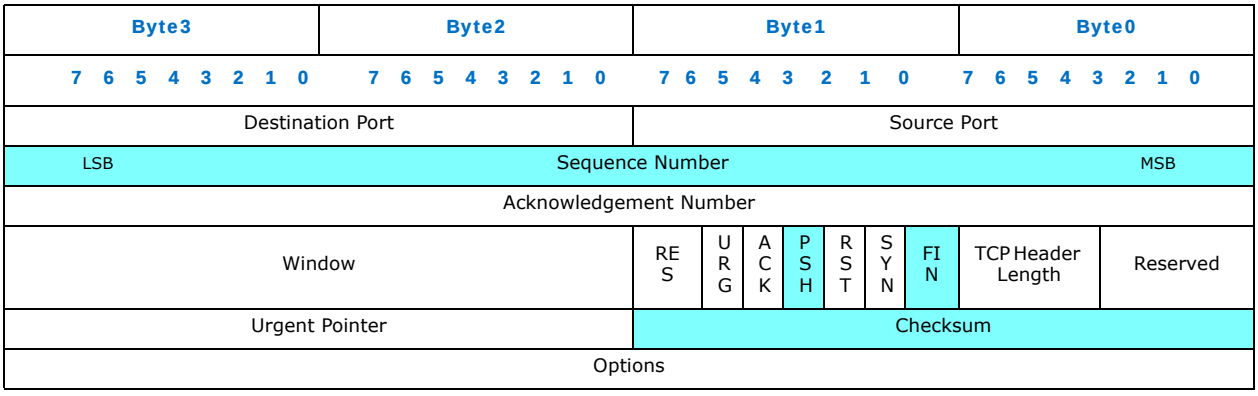


Figure 16-6 TCP Header (Little Endian)

The TCP header is always a multiple of 32-bit words. TCP options might 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 16-7). 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 difficult. 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 LSB of the 16-bit pseudo header sum, where the MSB of the 16-bit sum is the byte that corresponds to the first checksum byte out on the wire.

The *TCP 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 might be modified are defined as:

- PSH: receiver should pass this data to the application 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 16-7 TCP/UDP Pseudo Header Content for IPv4 (Traditional Representation)

IPv6 Source Address	
IPv6 Final Destination Address	
TCP/UDP Packet Length	
Zero	Next Header

Figure 16-8 TCP/UDP Pseudo Header Content for IPv6 (Traditional Representation)

Note: 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 (such as 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 (like TCP header plus TCP data). Some upper-layer protocols carry their own length information (such as 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, each time when 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 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 16-9 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. For example, the 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 0 1 2 3 4 5	6 7 8 9 0 1 2 3	4 5 6 7 8 9 0 1
1		2	3
Source Port		Destination Port	
Length		Checksum	

Figure 16-10 UDP Header (Traditional Representation)

Byte3								Byte2								Byte1								Byte0							
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 16-11 UDP Header (Little Endian Order)

The 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 previously discussed). 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.

16.3 Magic Packet

Magic Packets are broadcast frames, but could also be a multicast or unicast Ethernet MAC addresses. The X540 accepts this packet if it matches any of its pre-programmed Ethernet MAC addresses. A Magic Packet can be sent over a variety of connection-less 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 X540 expects the Ethernet MAC address stored in the RAL[0] and RAH[0] registers.

The X540 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.

16.4 Packet Types for Packet Split Filtering

The following packet types are supported by the packet split feature in the X540. This section describes the packets from a 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.

16.4.1 Type 1.1: Ethernet (VLAN/SNAP) IP Packets

16.4.1.1 Type 1.1: Ethernet, IP, Data

This type contains only the 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	S=(0/4)	Possible VLAN Tag	8100 ****	Compare	
12+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	4Xh	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.

Offset	# of Bytes	Field	Value	Action	Comment
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	

16.4.1.2 Type 1.2: Ethernet (SNAP/VLAN), IPv4, UDP

This type contains only the 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	S=(0/4)	Possible VLAN Tag	8100 ****	Check	
12+S	D=(0/8)	Possible Length + LLC/SNAP Header		Check	
12+D+S	2	Type	0800h	Compare	IP
IP Header					
14+D+S	1	Version/ HDR length	4Xh	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) 000h	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	

Offset	# of Bytes	Field	Value	Action	Comment
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.

16.4.1.3 Type 1.3: Ethernet (VLAN/SNAP) IPv4 TCP

This type contains only the 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	S=(0/4)	Possible VLAN Tag	8100 ****	Check	
12+S	D=(0/8)	Possible Length + LLC/SNAP Header		Check	
12+D+S	2	Type	0800h	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	4Xh	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

Offset	# of Bytes	Field	Value	Action	Comment
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.

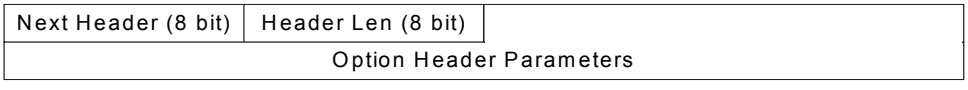
- N = (IP HDR length -5) * 4.
- F = (TCP header length - 5) * 4.

16.4.1.4 Type 1.4: Ethernet IPv4 and IPv6

16.4.1.4.1 IPv6 Header Options Processing

This type of processing looks at the next-header field and header length in order 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:



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).

16.4.1.4.2 IPv6 Next Header Values

When parsing an IPv6 header, the X540 does not parse all possible extension headers and if there is an extension header that is not supported by the X540 then 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.

16.4.1.4.3 Type 1.4.1: Ethernet, IPv4 and IPv6 Headers

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	S=(0/4)	Possible VLAN Tag	8100	Check	
12+S	D=(0/8)	Possible Length + LLC/SNAP Header		Check	
12+D+S	2	Type	0800h	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	4Xh	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	

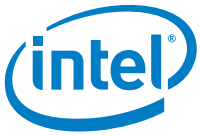
Offset	# of Bytes	Field	Value	Action	Comment
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x29	Compare	Ipv6
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
IPv6 Header					
34+D+S+N	1	Version/ Traffic Class	6Xh	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 in order for the packet to be parsed.

16.4.1.4.4 Type 1.4.2: Ethernet (VLAN/SNAP) IPv4, IPv6 and UDP Headers

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	S=(0/4)	Possible VLAN Tag	8100	Check	
12+S	D=(0/8)	Possible Length + LLC/ SNAP Header		Check	
12+D+S	2	Type	0800h	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	4Xh	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	6Xh	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	

Offset	# of Bytes	Field	Value	Action	Comment
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	

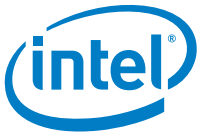
In this case the packet is split after (82+D+S+N+B) bytes.

N = (IP HDR length - 5) * 4.

16.4.1.4.5 Type 1.4.3: Ethernet (VLAN/SNAP) IPv4, Ipv6 and TCP Headers

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	S=(0/4)	Possible VLAN Tag	8100	Check	
12+S	D=(0/8)	Possible Length + LLC/ SNAP Header		Check	
12+D+S	2	Type	0800h	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	4Xh	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	



Offset	# of Bytes	Field	Value	Action	Comment
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	6Xh	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	
88+T	2	Window size	-	Ignore	

Offset	# of Bytes	Field	Value	Action	Comment
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

16.4.2 Type 2: Ethernet and IPv6

16.4.2.1 Type 2.1: Ethernet and 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 then the header is not split. The supported packet types for header split are programmed in the PSRTYPE register (per VF).

Offset	# of Bytes	Field	Value (hex)	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	S=(0/4)	Possible VLAN Tag	8100	Check	
12+S	D=(0/8)	Possible Length + LLC/SNAP Header		Check	
IPv6 Header					
12+D+S	2	Type	86DDh	Compare	IP
14+D+S	1	Version/ Traffic Class	6Xh	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 in order for the header to be split.

Offset	# of Bytes	Field	Value (hex)	Action	Comment
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).

16.4.2.1.1 Type 2.2: Ethernet (VLAN/SNAP) IPv6 and UDP Headers

This type contains only Ethernet header, IPv6 header, and UDP header.

Offset	# of Bytes	Field	Value (hex)	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	S=(0/4)	Possible VLAN Tag		Check	
12+S	D=(0/8)	Possible Length + LLC/SNAP Header		Check	
IPv6 Header					
12+D+S	2	Type	86DDh	Compare	IP
14+D+S	1	Version/ Traffic Class	6Xh	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	

Offset	# of Bytes	Field	Value (hex)	Action	Comment
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.

16.4.2.2 Type 2.3: Ethernet (VLAN/SNAP) IPv6 and TCP Headers

This type contains only Ethernet header, IPv6 header and UDP header.

Offset	# of Bytes	Field	Value (hex)	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	S=(0/4)	Possible VLAN Tag		Check	
12+S	D=(0/8)	Possible Length + LLC/SNAP Header		Check	
IPv6 Header					
12+D+S	2	Type	86DDh	Compare	IP
14+D+S	1	Version/ Traffic Class	6Xh	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	

Offset	# of Bytes	Field	Value (hex)	Action	Comment
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 = (TC header length - 5) * 4.
- The last 'next-header field of the last header of the IP section must be 0x11.

16.4.3 Type 3: Reserved

Type 3 used to be iSCSI packets (header split is not supported for iSCSI packets in the X540).

16.4.4 Type 4: NFS Packets

NFS headers can come in all the frames that contain UDP/TCP header. The NFS (and RPC headers) are extensions to these types of packets: 1.2, 1.3, 1.4.2, 1.4.3, 2.2, 2.3 that were presented in the previous sections. In this section only the NFS (and RPC) header is described and to its length should be added the length of the primary type of the packet.

The X540 starts looking within the UDP/TCP payload to check whether it contain an NFS header if either the source or destination port of the TCP/UDP equal to 0x801.

```
Destination port equal to 0x801 => NFS write request (as received by
the NFS server).

Source port equal to 0x801       => NFS read reply (as received by the
NFS client).
```

The VSZ/CSZ fields are 4 bytes long but there actual values are less than 2 words by definition, so hardware only checks the lower 2 bytes of these size fields.

RPC read requests are not described in this document since they contain only headers and no data -> no need to split.

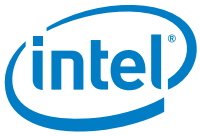
NFS over TCP is problematic - in fact, the RPC header might appear in the middle of the frame. It remains to be checked if software can support always putting RPC right next to the UDP/TCP header, but it doesn't have to.

16.4.4.1 Type 4.1: NFS Write Request

In all the write requests, the destination port of the TCP/UDP header must be 0x801.

16.4.4.1.1 Type 4.1.1: NFS Write Request (NFSv2)

Offset	# of Bytes	Field	Value (hex)	Action	Comment
RPC header					
0	D =(0/4)	Record Header	-	Ignore	If the previous header was a TCP header then this field contains 4 bytes.
0+D	4	Message type	0x00	Compare	
4+D	4	RPC version	0x02	Compare	
8+D	4	RPC program	0x18A63	Compare	
12+D	4	Program version	0x02	Compare	
16+D	4	Procedure	0x08	Compare	
20+D	4	Credentials Size (CSZ)	<400	Check	
24+D	B	Credentials Data	-	Ignore	B = (CSZ pad 4)
24+D+B	4	Verifier Flavor	-	Ignore	
28+D+B	4	Verifier Size (VSZ)	<400	Check	
32+D+B	F	Verifier Data		Ignore	F = (VSZ pad 4)
NFS Header					



Offset	# of Bytes	Field	Value (hex)	Action	Comment
32+D+B+F	32	handle		Ignore	
64+D+B+F	4	begin offset		Ignore	
68+D+B+F	4	Offset		Ignore	
72+D+B+F	4	Total count		Ignore	
76+D+B_F	4	Data len		Ignore	

In this case, the packet is split after (80+D+B+F) bytes should be added to the UDP/TCP type that was already parsed.

16.4.4.1.2 Type 4.1.2: NFS Write Request (NFSv3)

Offset	# of Bytes	Field	Value (hex)	Action	Comment
RPC Header					
0	D =(0/4)	Record Header	-	Ignore	If the previous header was a TCP header than this field contains 4 bytes.
0+D	4	Message type	0x00	Compare	
4+D	4	RPC version	0x02	Compare	
8+D	4	RPC program	0x18A63	Compare	
12+D	4	Program version	0x03	Compare	
16+D	4	Procedure	0x07	Compare	
20+D	4	Credentials Size (CSZ)	<400	Check	
24+D	B	Credentials Data	-	Ignore	B = (CSZ padded to 4)
24+D+B	4	Verifier Flavor	-	Ignore	
28+D+B	4	Verifier Size (VSZ)	<400	Check	
32+D+B	F	Verifier Data		Ignore	F = (VSZ padded to 4)
NFS Header					
32+D+B+F	4	Fhandle_size	<64	Check	
36+D+B+F	S	fhandle		Ignore	S = (Fhandle_size padded to 4)
36+D+B+F+S	8	Offset		Ignore	

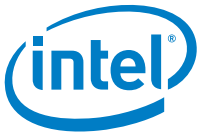
Offset	# of Bytes	Field	Value (hex)	Action	Comment
44+D+B+F+S	4	Count		Ignore	
48+D+B+F+S	4	Stable_how		Ignore	
52+D+B+F+S	4	Data len		Ignore	

In this case the packet is split after (56+D+B+F+S) bytes should be added to the UDP/TCP type that was already parsed.

16.4.4.1.3 Type 4.1.3: NFS Write Request (NFSv4)

Offset	# of Bytes	Field	Value (hex)	Action	Comment
RPC Header					
0	D =(0/4)	Record Header	-	Ignore	If the previous header was a TCP header than this field contains 4 bytes.
0+D	4	Message type	0x00	Compare	
4+D	4	RPC version	0x02	Compare	
8+D	4	RPC program	0x18A63	Compare	
12+D	4	Program version	0x04	Compare	
16+D	4	Procedure	0x26	Compare	
20+D	4	Credentials Size (CSZ)	<400	Check	
24+D	B	Credentials Data	-	Ignore	B = (CSZ pad 4)
24+D+B	4	Verifier Flavor	-	Ignore	
28+D+B	4	Verifier Size (VSZ)	<400	Check	
32+D+B	F	Verifier Data		Ignore	F = (VSZ pad 4)
NFS Header					
32+D+B+F	8	State ID		Ignore	
40+D+B+F	8	Offset		Ignore	
48+D+B+F	4	Stable_how		Ignore	
52+D+B+F	4	Data len		Ignore	

In this case the packet is split after (100+D+F) bytes should be added to the UDP/TCP type that was already parsed.



16.4.4.1.4 Type 4.2.1: NFS Read Response (NFSv3)

Offset	# of Bytes	Field	Value (hex)	Action	Comment
RPC Header					
0	D =(0/4)	Record Header	-	Ignore	If the previous header was a TCP header than this field contains 4 bytes.
0+D	4	XID		Ignore	
4+D	4	Message type	0x01	Compare	
8+D	4	Reply status	0x00	Ignore	'0' means OK and only if this value is '0' is there additional data.
12+D	4	Verifier Flavor	-	Ignore	
16+D	4	Verifier Size (VSZ)	<400	Check	
20+D	F	Verifier Data	-	Ignore	F = (VSZ pad 4)
20+D+F	4	Accept status	0x00	Ignore	'0' means OK
NFS Header					
24+D+F	4	Status	0x00	Ignore	
28+D+F	68	Attributes	-	Ignore	
96+D+F	4	Data len	-	Ignore	

In this case the packet is split after (40+D+F+S) bytes should be added to the UDP/TCP type that was already parsed.

16.4.4.1.5 Type 4.2.1: NFS Read Response (NFSv4)

Offset	# of Bytes	Field	Value (hex)	Action	Comment
RPC Header					
0	D =(0/4)	Record Header	-	Ignore	If the previous header was a TCP header than this field contains 4 bytes.
0+D	4	XID	-	Ignore	
4+D	4	Message type	0x01	Compare	

Offset	# of Bytes	Field	Value (hex)	Action	Comment
8+D	4	Reply status	0x00	Ignore	'0' means OK and only if this value is '0' is there additional data.
12+D	4	Verifier Flavor	-	Ignore	
16+D	4	Verifier Size (VSZ)	<400	Check	
20+D	F	Verifier Data	-	Ignore	F = (VSZ pad 4)
20+D+F	4	Accept status	0x00	Ignore	'0' means OK
NFS header					
24+D+F	4	Status	0x00	Ignore	
	4	Attr_follow	-	Check	
28+D+F	S	Attributes	-	Ignore	Attr_flow=1? S=84: S=0
28+D+F+S	4	Count	-	Ignore	
32+D+F+S	4	Eof	-	Ignore	
36+D+F+S	4	Data len	-	Ignore	

In this case the packet is split after (36+D+F) bytes should be added to the UDP/TCP type that was already parsed.

16.5 IPSec Formats Run Over the Wire

This section describes the IPSec packet encapsulation formats run over the wire by IPsec packets that affect the offload in either Tx or Rx direction.

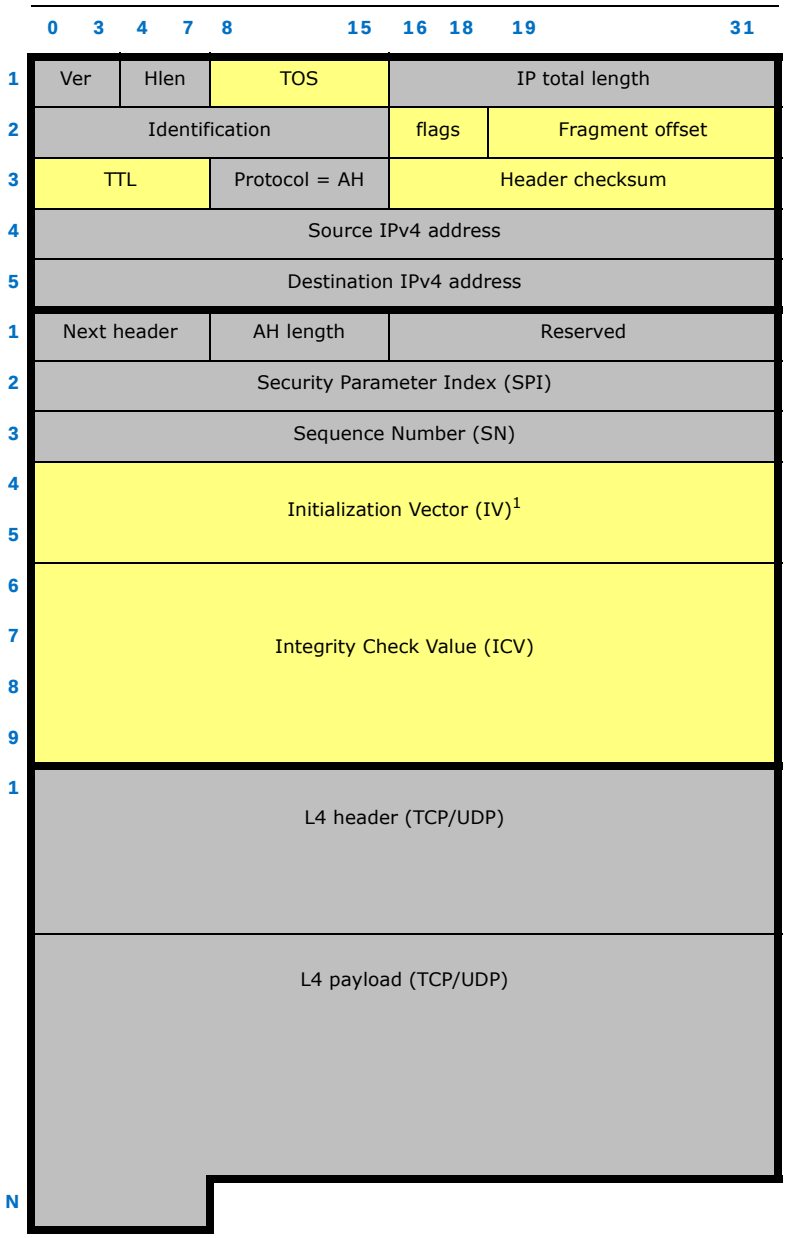
The following legend is valid for the figures Figure 16-12 through Figure 16-17 of this section.

.

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 switching between the source and the destination and must thus be 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.

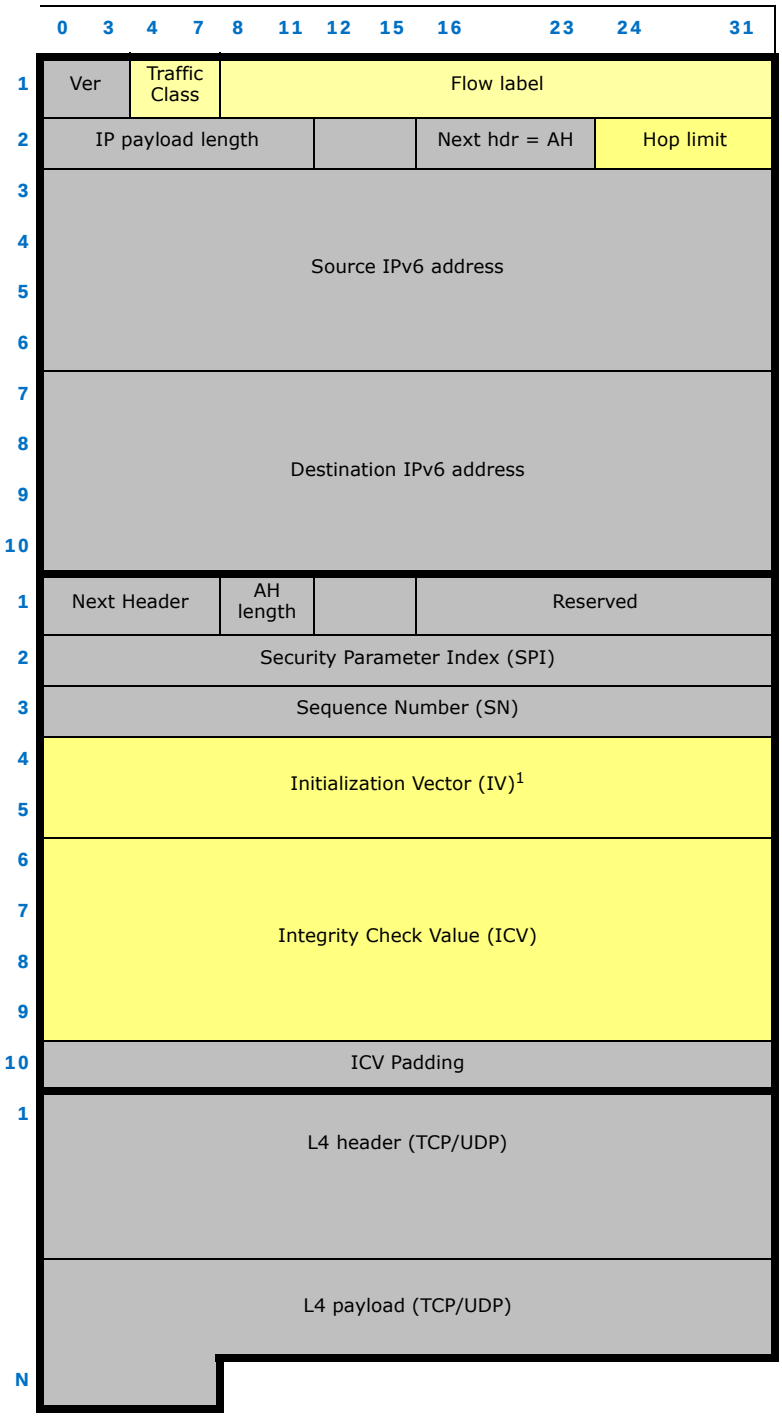
16.5.1 AH Formats

- 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 (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 (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, such as 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 operating system 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 zero by the sender (and the receiver) when the SA is established, such as the first packet sent using a given SA has a sequence number of one.
  - 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 means 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 software device 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 16-12 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 16-13 AH Packet Over IPv6

16.5.2 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 (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 (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 operating system 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 zero by the sender (and the receiver) when the SA is established, such as the first packet sent using a given SA has a sequence number of one.
  - 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 means that these bytes are sent over the wire. A remote IPsec implementation might 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.

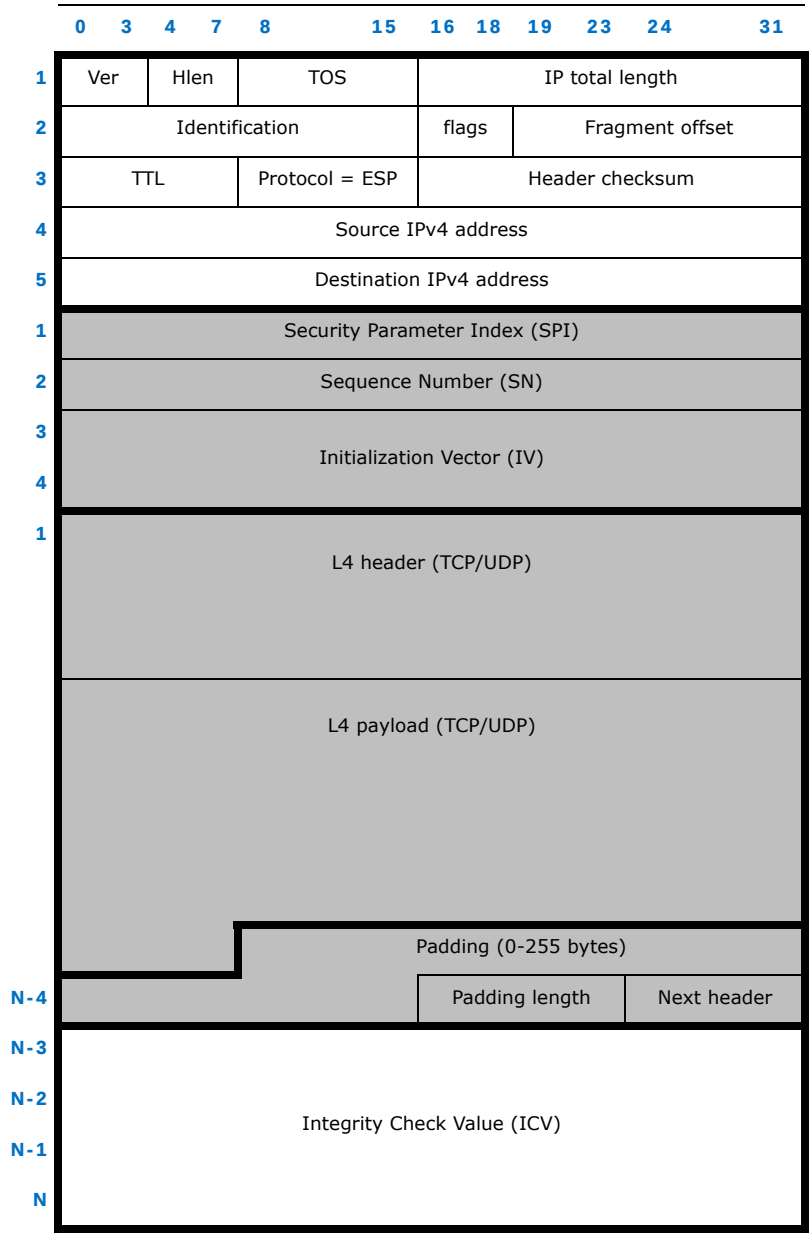


Figure 16-14 Authenticated Only ESP Packet Over IPv4

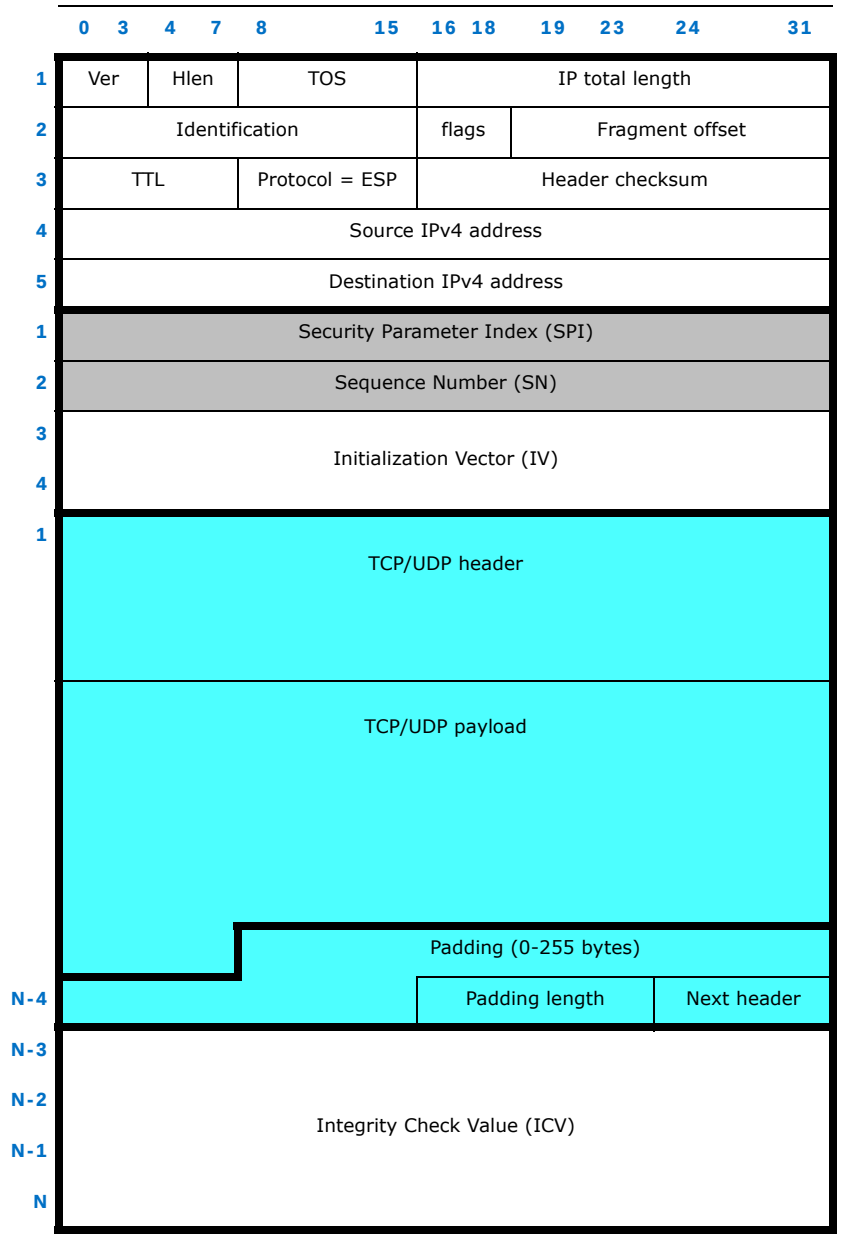


Figure 16-15 Authenticated and Encrypted ESP Packet Over IPv4

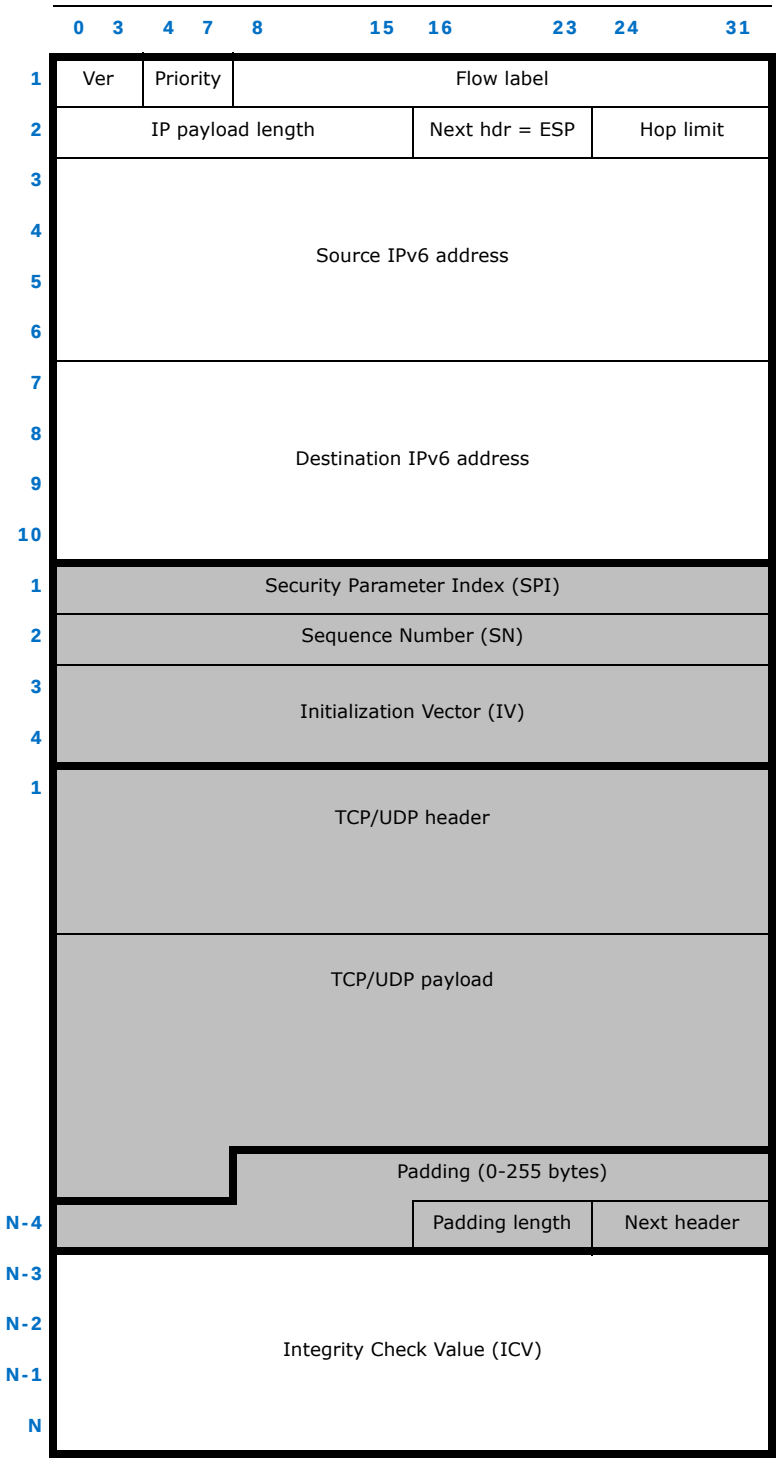


Figure 16-16 Authenticated Only ESP Packet Over IPv6

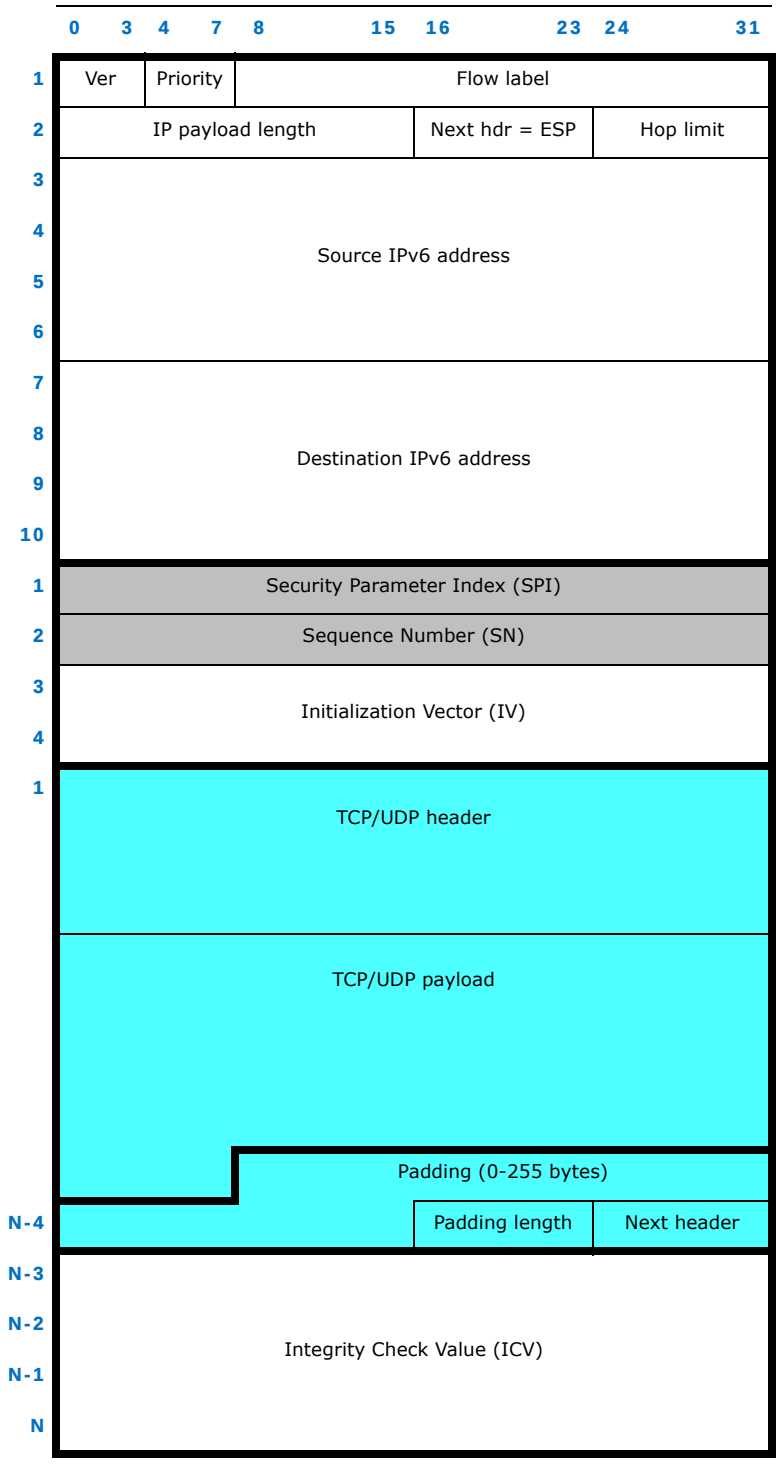


Figure 16-17 Authenticated and Encrypted ESP Packet Over IPv6

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 plain-text input fields. Refer to Section 7.12.7.

16.6 FCoE Framing

Related Standards

- FC-FS-2 - FRAMING AND SIGNALING-2 (FC-FS-2) Rev 1.00
- FCoE - Fibre Channel over Ethernet Draft Presented at the T11 on May 2007

16.6.1 FCoE Frame Format

FCoE packets encapsulate FC frames as shown in Figure 16-18 and Figure 16-19. Maximum expected FCoE frame size is 2164 bytes. This size does not include FC extension headers (not expected in FCoE), without optional MACsec 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 MSB is first on the wire and the LSb 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.

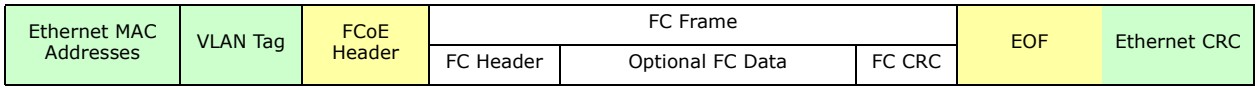
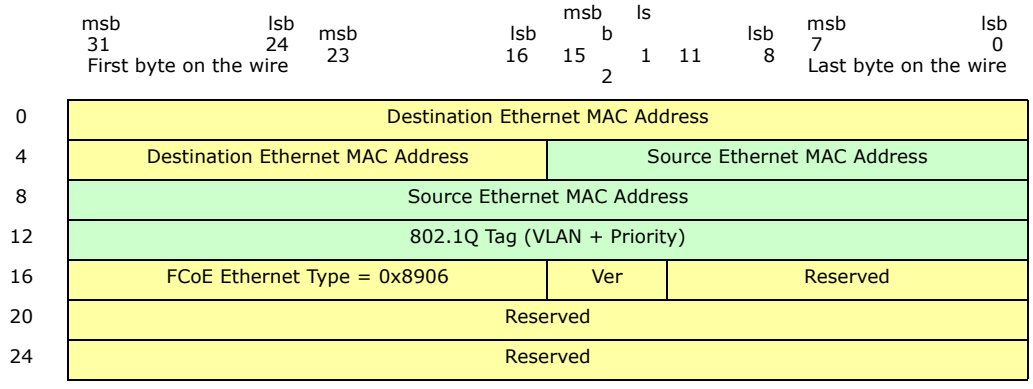


Figure 16-18 Ethernet Encapsulation to FC Frames



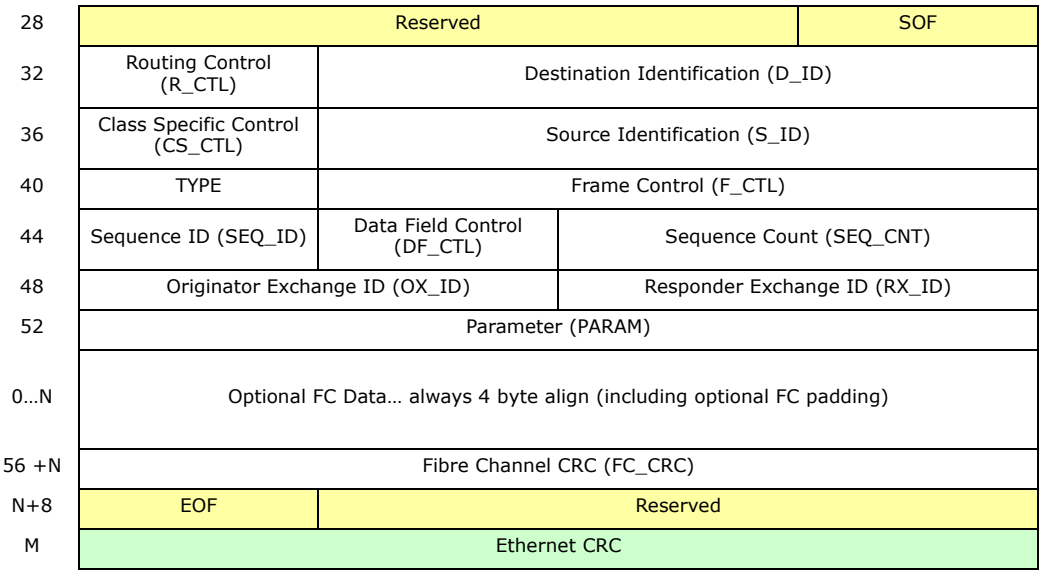


Figure 16-19 FCoE Packet Structure

16.6.1.1 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.

16.6.1.2 802.1Q Tag

802.1Q tagging is mandatory for FCoE usage. FCoE assumes DCB functionality that provides priority-based FC and QCN. These functions require 802.1Q tag presence to define packet priority.

16.6.1.3 FCoE Header

The FCoE header is composed of a new FCoE Ethernet type, FCoE version tag and the start-off-frame tag.

FCoE Ethernet Type Equals 0x8906

Version (Ver)A 4-bit field that indicates the FCoE protocol version number. The X540 supports FCoE version as defined by FCRXCTRL.FCOEVER.

Start-of-Frame (SOF) The FCoE SOF is a subset of the FC-FS-2 SOF codes as listed in the Table 16-1:

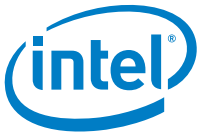


Table 16-1 E_SOF Mapping

FC SOF	FCoE SOF Code	FC Traffic Class	Comment
SOFf	0x28	F	Fabric SOF. 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.

16.6.1.4 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 EOF maps the FC-FS-2 EOF codes as listed in Table 16-2:

Table 16-2 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.

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:
$X_{32}+X_{26}+X_{23}+X_{22}+X_{16}+X_{12}+X_{11}+X_{10}+X_8+X_7+X_5+X_4+X_2+X+1$

16.6.2 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 16-20 while relevant fields are detailed in this section.

SOF	Extended Header	FC Header	Optional Headers	FC Payload (FC Data & optional padding)	FC CRC	EOF

Figure 16-20 FC Frame Format

16.6.2.1 FC SOF and EOF

FC SOF delimiter and 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 listed in Table 16-1 and Table 16-2.

16.6.2.2 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 entire FC frame. Hardware adds this field if the FCoE bit is set in the transmit context descriptor. The FC CRC offload is described in more detail in Section 7.13.3.2.

16.6.2.3 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 hardware as part of Large Send Offload (LSO). These fields are indicated in this section as Dynamic while fields that are not modified by 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 16.6.2.3.1. Section 7.13.2.7 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 7.13.2.7.1 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 the 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 zero. 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 zero or be increased by one from the last used SEQ_CNT.

Originator Exchange ID (OX_ID)

The OX_ID is a 2-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 0xFFFF. A value of 0xFFFF indicates that the OX_ID is unassigned and that the originator is not enforcing uniqueness via the OX_ID. The X540 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 2-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 0xFFFF.

Parameter (PARAM)

The *Parameter* field is a *Dynamic* field that is based on frame type. For data frames with the relative offset present bit set to 1b, 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.

16.6.2.3.1 Frame Control (F_CTL)

The Frame Control (F_CTL) is a 3-byte field that contains control information relating to the frame content. If an error in bit usage is detected, software initiates a reject frame (P_RJT) in response with an appropriate reason code (FCoE software device driver responsibility). The F_CTL format is listed in Table 16-3.

When a bit(s) is designated as *Static* it is provided by the FCoE software device driver as part of the large send header. 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 software device driver as part of the large send header. Hardware might change it in some of the packets in the large send as described in Section 7.13.2.7. Exact setting of these bit(s) is listed in Table 16-3.

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 16-3 F_CTL Format

Control Field	Bit	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	0b = Word 1, bits 31-24 = CS_CTL. 1b = Word 1, bits 31-24 = priority. Defines the meaning of the CS_CTL byte in the FC header to be either CS_CTL or priority indication.
Sequence Initiative	16	Dynamic	0b = Hold sequence initiative. 1b = Transfer sequence initiative. This bit is used to transfer initiative from a sender to a recipient. This bit is meaningful only on the last frame of a sequence (when bit 19 - *End Sequence* is set).
X_ID reassigned	15	Static	Obsolete
Invalidate X_ID	14	Static	Obsolete
ACK Form	13:12	Static	00b = No assistance provided 10b = Reserved. 01b = Ack_1 Required 11b = Ack_0 Required. ACK form is meaningful on all Class 1, Class 2, or Class 6 Data frames of a sequence and on all connect request frames. ACK_Form is not meaningful on Class 1, Class 2, or Class 6 Link_Control frames, or any Class 3 frames.
Data Compression	11	Static	Obsolete
Data Encryption	10	Static	Obsolete
Retransmitted Sequence	9	Static	0b = Original sequence transmission. 1b = Sequence retransmission. Meaningful in Class 1 and 6 and only.

Control Field	Bit	Type	Description
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 *End Sequence* is set to 1b and the *Sequence Initiative* bit is cleared 0b.
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-trams,ot 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 Seq with immediate re-transmit.
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).

16.6.2.4 FC Extended Headers

Note: The following section is provided as a background on FC and is not required for hardware implementation or software implementation since in FCoE frames, extended headers are not expected. The following quote is from the FCoE spec Rev 0.7 "No Extended Headers are relevant for the operations of a DCB NIC".

The diagrams in Figure 16-21, Figure 16-22 and Table 16-4 show the FCoE frame structure with extended headers and lists the existing headers. Extended headers are identified by the R_CTL value as shown in the diagrams. The X540 does not support these extended headers since they are not expected in FCoE usage.

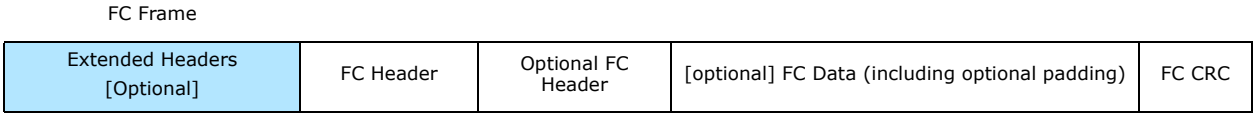


Figure 16-21 FC Frame Format with Extended Headers

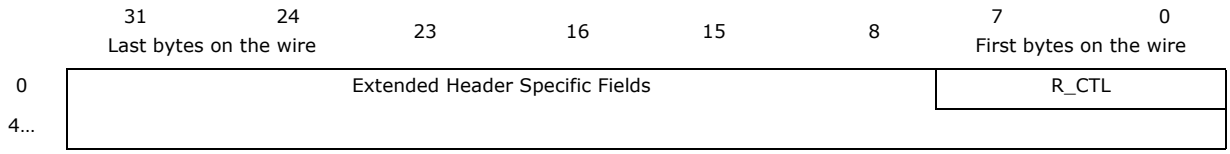


Figure 16-22 Extended Header Format

Table 16-4 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	-

16.6.2.5 FC Optional Headers

Note: Most of the following section is provided as a background on FC and is not required for hardware implementation. Readers can skip the detailed explanation of the optional headers provided and refer to the tables and figures that follow the text.

FCoE frames might include FC optional headers. These headers (if they exist) would always show in the first frame in a sequence. While FC implementation might include the FC optional headers only on the first frame in a sequence. In large send functionality, the FC optional headers might show only on the first frame as shown in Figure 7-49 and Table 16-5 (in Section 7.13.2.7.1).

The following diagrams Table 16-5, Figure 16-23 and Figure 16-24 show the FCoE frame structure with optional headers and lists the optional headers. The optional headers (that are present) are always ordered as shown in Figure 16-23 and Figure 16-24. Their presence is indicated in the *Data Field Control* (DF_CTL) field in the FC header as listed in Table 16-5

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 16-5 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 bytes
bits 1:0	Device Header	0, 16, 32, or 64 bytes

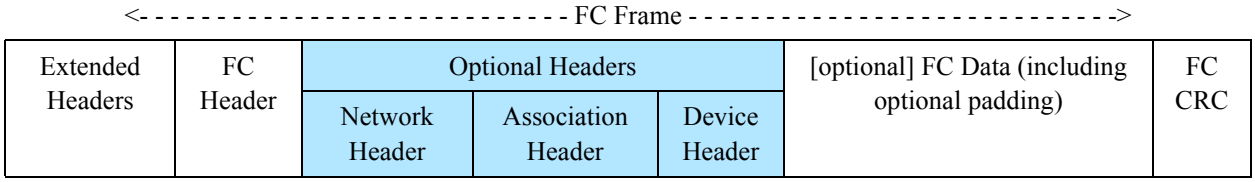


Figure 16-23 FC Frame Format With Optional Headers (Without ESP Header)

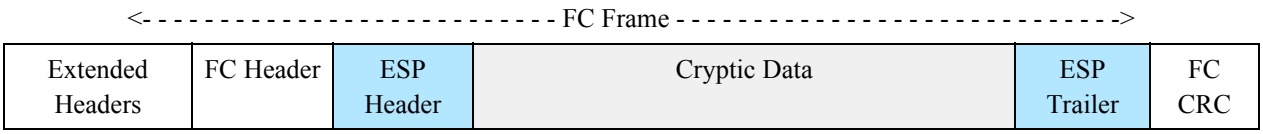


Figure 16-24 FC Frame format with Optional Headers (With ESP Header)

ESP Header

This is the first optional header that covers the entire 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, 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 1b. Hardware does not support LSO when ESP optional header is used. When present, the ESP header and trailer are present in all frames of the exchange.

Network Header

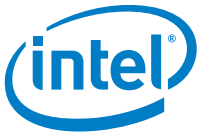
The network header, if used, must be present only in the first data frame of a sequence. A bridge or a gateway node that interfaces to an external network might use the network header. The network header, is an optional header 16 bytes long within the FC data field content. Its presence is indicated by bit 5 in the DF_CTL field being set to 1b. The network header might be used for routing between FCoE networks of different fabric address spaces, or FCoE and non-FCoE networks. The network header contains name identifiers for network destination addresses and network source addresses.

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 1b. The association header might 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 X540 does not use the association for any filtering purposes but rather uses the OX_ID.

Device Header

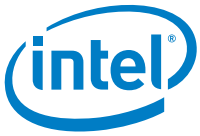
The device header, if present, must be present either in the first data frame or in all data frames of a sequence. If LSO is used then the device header, if present, is present only in the first frame of the same large send. The device Header, if present, must be 16, 32, or 64 bytes in size as defined by bits 1:0 in the DF_CTL field. The contents of the device header are controlled at a level above FC-2. Upper ULP might use a device header, requiring the device header to be supported. The device header might 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 might reject the frame with the reason code of TYPE not supported.



17.0 Glossary and Acronyms

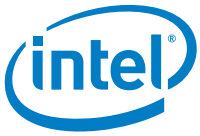
Term	Definition
1’s complement	A system known as ones' complement can be used to represent negative numbers in a binary system. The ones' complement form of a negative binary number is the bitwise NOT applied to it.
2’s complement	A system of two's-complement arithmetic represents negative integers by counting backwards and wrapping around. Any number whose left-most bit is 1 is considered negative.
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.
AAD	Additional Authentication Data input, which is authenticated data that must be left un-encrypted.
ACK	Acknowledgement
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.
AN	Auto negotiation
AN	Association Number

Term	Definition
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.
ARP	Address Resolution Protocol
b/w or BW	Bandwidth
backbone	Aa 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
BKM	Best Known Method
BMC	Baseboard Management Controller
BME	Bus Master Enable
BT	Byte Time.
BYTE alignment	Implies that the physical addresses can be odd or even. Examples: 0FECBD9A1h, 02345ADC6h.
BWG	Bandwidth Group.
CA	Secure Connectivity Association (CA): A security relationship, established and maintained by key agreement protocols, that comprises a fully connected subset of the service access points in stations attached to a single LAN that are to be supported by MACsec.
CAM	Content Addressable Memory
Ciphertext	Encrypted data, whose length is exactly that of the plaintext.
CFI	Canonical Form Indicator
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.
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.



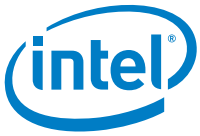
Term	Definition
CPID	Congestion Point Identifier –which should include the congestion point Ethernet MAC Address, as well as a local identifier for the local congestion entity, usually a queue in the switch.
CRC	Cyclic Redundancy Check A cyclic redundancy check (CRC) is a type of function that takes 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
CTS	Cisco Trusted Security
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.
D3Hot	In D3 the LAN Controller only responds to PCI configuration accesses and does not generate master cycles.
D3Cold	Power Off If Vcc is removed from the device al of its PCI functions transition immediately to D3 cold. When Power is restored a PCI Reset must be asserted.
Dr	Internal Power management state when minimal function is provided (WoL, Manageability)
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
DCB	Data Center Ethernet.
DCX	DCB Configuration Exchange protocol
DDP	Direct Data placement
DFT	Testability.
DFX	Design for *

Term	Definition
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.
DSP	Digital Signal Processor
DTE	Data Terminal Equipment
DUT	Device Under Test
DWORD (Double-Word) alignment	Implies that the physical addresses may only be aligned on 4-byte boundaries; i.e., the last nibble of the address may only end in 0, 4, 8, or Ch. For example, 0FECBD9A8h.
EAPOL	Extensible Authentication Protocol over LAN
EAS	External Architecture Specification.
ECC	Error Correction Coding
ECRC	End to End CRC
EDB	End Data Bit
ECC	Error Correction Coding
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
E-SOF	FCoE Start of Frame



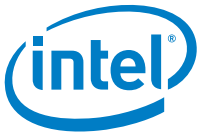
Term	Definition
ESP	IP Encapsulating Security Payload — An IPsec header providing encryption and authentication capabilities defined in RFC 4303. The Encapsulating Security Payload (ESP) extension header provides origin authenticity, integrity, and confidentiality protection of a packet. ESP also supports 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.
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.
FCoE	Fiber Channel over Ethernet
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).
FCS	Frame Check Sequence of Ethernet frames
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”.
FC Frame	Fiber Channel Frames are the smallest units sent between the initiator and the target. The FC-FS-2 spec define 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.
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.
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

Term	Definition
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 multithreading technology on the Pentium 4 microarchitecture. 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
ICV	128-bits Integrity Check Value (referred also as authentication tag). used for MACsec header and signature
IDS	intrusion detection systems
IFCS	Insert Frame Check Sequence of Ethernet frames
IFS	Inter Frame Spacing
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



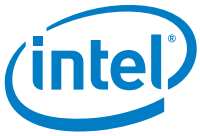
Term	Definition
IP tunneling	IP tunneling is the process of embedding one IP packet inside of another, for the purpose of simulating a physical connection between two remote networks across an intermediate network. IP tunnels are often used in conjunction with IPSec protocol to create a VPN between two or more remote networks across a "hostile" network such as the Internet.
IPC	Inter Processor Communication.
IP — CPMP	Carrier Performance Measurement Plan
IPG	Inter Packet Gap.
IP Sec	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 doesn't 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 spec terminology) i.e. control and access the off loading engine (SecY in 802.1AE spec 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 (10GBASE-T) 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
LSC	Link Status Change
LS	Least significant / Lowest order (for example: LS bit = Least significant bit)
LSO	Large Send Offload same as TSO
LSP	Link Strict Priority
LTSSM	Link Training and Status State Machine Defined in the PCIe specs.
MAC	Media Access Control.
MACsec, 802.1AE	Is a MAC level encryption/authentication scheme defined in IEEE 802.1AE that uses symmetric cryptography. The 802.1AE defines an AES-GCM 128 bit key as a mandatory cipher suite which can be processed by the LAN controller.
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.



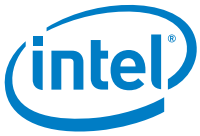
Term	Definition
Mod / Modulo	In computing, the modulo operation finds the remainder of division of one number by another.
MPA	Marker PDU Aligned Framing for TCP
MPDU	MACSEC Protocol Data Unit including SecTag, User Data and ICV
MRQC	Multiple Receive Queues Command register
MS	Most significant / Highest 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 Acknowledgement
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
NEXT	Near End Crosstalk
NIC	Network Interface Controller.
NFTS	Number of Fast Training Signals
NFS	Network File Server
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
Core	Network Interface Registers

Term	Definition
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
PBA	Pending Bit Array (in MSI-X context)
PBA	Printed Board Assembly (in NVM or board context)
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 (PN) in a MACsec context: Monotonically increasing value used to uniquely identify a MACsec frame in the sequence.
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 D3Hot	A Power down state with the PCI continuing to receive a proper power supply.
Power State D3Cold	A Power down state with the PCI also in a power down state.
Power State Dr	Device state when PCIe reset is asserted.
Power State Sx	Lan Connected Device: SMBus Active and PCI Powered down.
PPM	Parts Per Million
PRBS	Pseudo-Random Binary Sequence



Term	Definition
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.
Receive latency	measured from packet reception from the wire and until the descriptor is updated on PCIe.
RDMA	Remote Direct Memory Access
RDMAP	Remote Direct Memory Access Protocol
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
Rx, RX	Receive
SA	Security Association (in MACSec)
SA	Source Address (in Ethernet frame context)
SAC	Single Address Cycle (SAC) messages
SAK	Security Associations Key

Term	Definition
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.
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.
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.
SCL signal	SM Bus Clock
SCTP	Stream Control Transmission Protocol
SDA signal	SM Bus Data
SDP	Software-Definable Pins
SecY	802.1AE spec 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.
SNMP	Standard Network Management Protocol
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



Term	Definition
SoL	Serial Over LAN Serial Over LAN is 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.
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 will be 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
SPXB interface	PCI Express Backbone
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
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.
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

Term	Definition
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).
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.
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

Term	Definition
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.